DSAC - Differentiable RANSAC for Camera Localization

Eric Brachmann[1], Alexander Krull[1], Sebastian Nowozin[2]
Jamie Shotton[2], Frank Michel[1], Stefan Gumhold[1], Carsten Rother[1]
[1] TU Dresden, [2] Microsoft

Abstract

RANSAC is an important algorithm in robust optimization and a central building block for many computer vision applications. In recent years, traditionally hand-crafted pipelines have been replaced by deep learning pipelines, which can be trained in an end-to-end fashion. However, RANSAC has so far not been used as part of such deep learning pipelines, because its hypothesis selection procedure is non-differentiable. In this work, we present two different ways to overcome this limitation. The most promising approach is inspired by reinforcement learning, namely to replace the deterministic hypothesis selection by a probabilistic selection for which we can derive the expected loss w.r.t. to all learnable parameters. We call this approach DSAC, the differentiable counterpart of RANSAC. We apply DSAC to the problem of camera localization, where deep learning has so far failed to improve on traditional approaches. We demonstrate that by directly minimizing the expected loss of the output camera poses, robustly estimated by RANSAC, we achieve an increase in accuracy. In the future, any deep learning pipeline can use DSAC as a robust optimization component[1].

1. Introduction

Introduced in 1981, the random sample consensus (RANSAC) algorithm [11] remains the most important algorithm for robust estimation. It is easy to implement, it can be applied to a wide range of problems and it is able to handle data with a substantial percentage of outliers, *i.e*. data points that are not explained by the data model. RANSAC and variants thereof [39, 28, 7] have, for many years, been important tools in computer vision, including multi-view geometry [16], object retrieval [29], pose estimation [36, 4] and simultaneous localization and mapping (SLAM) [27]. Solutions to these diverse tasks often involve a common strategy: Local predictions (*e.g*. feature matches) induce a global model (*e.g*. a homography). In this schema, RANSAC provides robustness to erroneous local predictions.

Recently, deep learning has been shown to be highly successful at image recognition tasks [37, 17, 13, 31], and, increasingly, in other domains including geometry [10, 19, 20, 9]. Part of this recent success is the ability to perform end-to-end training, *i.e*. propagating gradients back through an entire pipeline to allow the direct optimization of a task-specific loss function, examples include [41, 1, 38].

In this work, we are interested in learning components of a computer vision pipeline that follows the principle: predict locally, fit globally. As explained earlier, RANSAC is an integral component of this wide-spread strategy. We ask the question, whether we can train such a pipeline end-to-end. More specifically, we want to learn parameters of a convolutional neural network (CNN) such that models, fit robustly to its predictions via RANSAC, minimize a task specific loss function.

RANSAC works by first creating multiple model hypotheses from small, random subsets of data points. Then it scores each hypothesis by determining its consensus with all data points. Finally, RANSAC selects the hypothesis with the highest consensus as the final output. Unfortunately, this hypothesis selection is non-differentiable, meaning that it cannot directly be used in an end-to-end-trained deep learning pipeline.

A common approach within the deep learning community is to soften non-differentiable operators, *e.g*. argmax in LIFT [41] or visual word assignment in NetVLAD [1]. In the case of RANSAC, the non-differentiable operator is the argmax operator which selects the highest scoring hypothesis. Similar to [41], we might substitute the argmax for a soft argmax, which is a weighted average of arguments [6]. We indeed explore this direction but argue that this substitution changes the underlying principle of RANSAC. Instead of learning how to select a good hypothesis, the pipeline learns a (robust) average of hypotheses. We show experimentally that this approach learns to focus on a narrow selection of hypotheses and is prone to overfitting.

Alternatively, we aim to preserve the hard hypothesis selection but treat it as a probabilistic process. We call this

[1] We will make our source code publicly available on the DSAC project website.

approach DSAC – Differentiable SAmple Consensus – our new, differentiable counterpart to RANSAC. DSAC allows us to differentiate the *expected* loss of the pipeline w.r.t. to all learnable parameters. This technique is well known in reinforcement learning, for stochastic computation problems like policy gradient approaches [34].

To demonstrate the principle, we choose the problem of camera localization: From a single RGB image in a known static scene, we estimate the 6D camera pose (3D translation and 3D rotation) relative to the scene. We demonstrate an end-to-end trainable solution for this problem, building on the scene coordinate regression forest (SCoRF) approach [36, 40, 5]. The original SCoRF approach uses a regression forest to predict the 3D location of each pixel in an observed image in terms of 'scene coordinates'. A hypothesize-verify-refine RANSAC loop then randomly select scene coordinates of four pixel locations to generate an initial set of camera pose hypotheses, which is then iteratively pruned and refined until a single high-quality pose estimate remains. In contrast to previous SCoRF approaches, we adopt two CNNs for predicting scene coordinates and for scoring hypotheses. More importantly, the key novelty of this work is to replace RANSAC by our new, differentiable DSAC.

Our contributions are in short:

- We present and discuss two alternative ways of making RANSAC differentiable, by soft argmax and probabilistic selection. We call our new RANSAC version, with the latter option, DSAC (Differentiable SAmple Consensus).
- We put both options into a new end-to-end trainable camera localization pipeline. It contains two separate CNNs, linked by our new RANSAC, motivated by previous work [36, 23].
- We validate experimentally that the option of probabilistic selection is superior, *i.e*. less sensitive to overfitting, for our application. We conjecture that the advantage of probabilistic selection is allowing hard decisions and, at the same time, keeping broad distributions over possible decisions.
- We exceed the state-of-the-art results on camera localization by 3.3%.

1.1. Related Work

Over the last decades, researchers have proposed many variants of the original RANSAC algorithm [11]. Most works focus on either or both of two aspects: speed [8, 28, 7], or quality of the final estimate [39, 8]. For detailed information about RANSAC variants we refer the reader to [30]. To the best of our knowledge, this work is the first to introduce a differentiable variant of RANSAC for the purpose of end-to-end learning.

In the following, we review previous work on differentiable algorithms and solutions for the problem of camera localization.

Differentiable Algorithms. The success of deep learning began with systems in which a CNN processes an image in one forward pass to directly predict the desired output, *e.g*. class probabilities [22], a semantic segmentation [25] or depth values and normals [10]. Given a sufficient amount of training data, CNNs can autonomously discover useful strategies for solving a task at hand, *e.g*. hierarchical part-structures for object recognition [42].

However, for many computer vision tasks, useful strategies have been known for a long time. Recently, researchers started to revisit and encode such strategies explicitly in deep learning pipelines. This can reduce the necessary amount of training data compared to CNNs with an unconstrained architecture [35]. Yi *et al*. [41] introduced a stack of CNNs that remodels the established sparse feature pipeline of detection, orientation estimation and description, originally proposed in [26]. Arandjelovic *et al*. [1] mapped the Vector of Locally Aggregated Descriptors (VLAD) [2] to a CNN architecture for place recognition. Thewlis *et al*. [38] substituted the recursive decoding of Deep Matching [32] with reverse convolutions for end-to-end trainable dense image matching.

Similar in spirit to these works, we show how to train an established, RANSAC-based computer vision pipeline in an end-to-end fashion. Instead of substituting hard assignments by soft counterparts as in [41, 1], we enable end-to-end learning by turning the hard selection into a probabilistic process. Thus, we are able to calculate gradients to minimize the expectation of the task loss function [34].

Camera Localization. The SCoRF camera localization pipeline [36], already discussed in the introduction, has been extended in several works. Guzman-Rivera *et al*. [14] trained a random forest to predict diverse scene coordinates to resolve scene ambiguities. Valentin *et al*. [40] trained the random forest to predict multi-model distributions of scene coordinates for increased pose accuracy. Brachmann *et al*. [5] addressed camera localization from an RGB image instead of RGB-D, utilizing the increased predictive power of an auto-context random forest. None of these works support end-to-end learning.

In a system similar to SCoRF but for the task of object pose estimation, Krull *et al*. [23] trained a CNN to measure hypothesis consensus by comparing rendered and observed images. In this work, we adopt the idea of a CNN measuring hypothesis consensus, but learn it jointly with the scene coordinate regressor and in an end-to-end fashion.

Kendall *et al*. [20] demonstrated that a single CNN is able to directly regress the 6D camera pose given an RGB image, but its accuracy on indoor scenes is inferior to a RGB-based SCoRF pipeline [5].

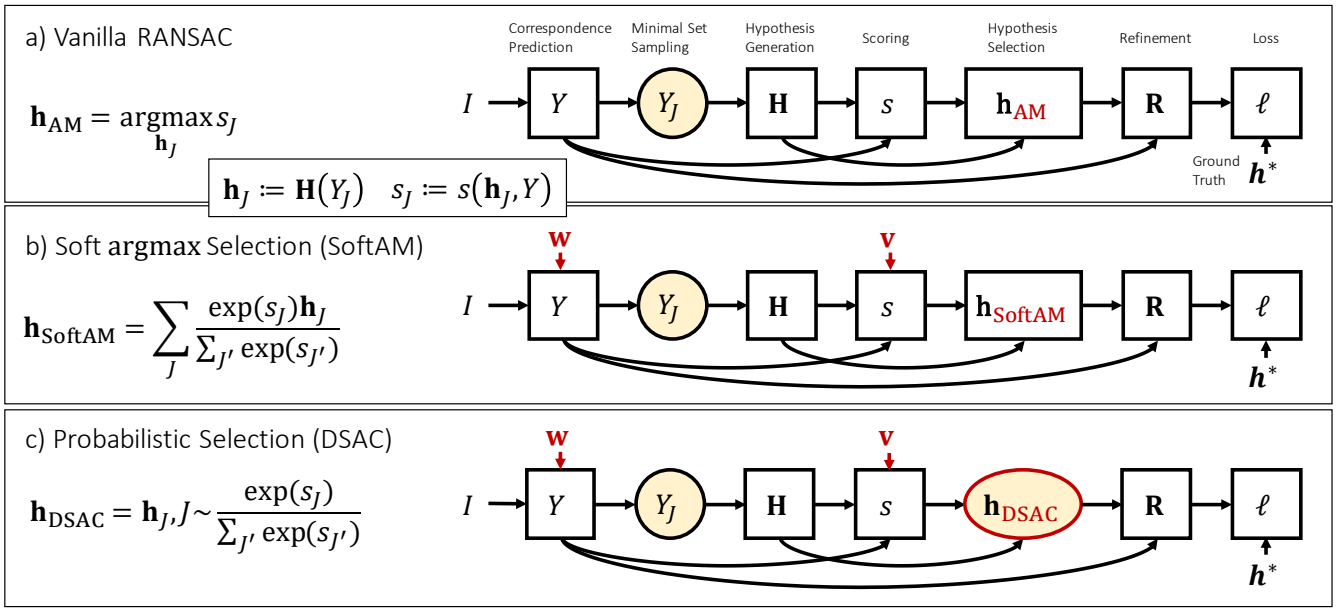

Figure 1. **Stochastic Computation Graphs [34].** A graphical representation of three RANSAC variants investigated in this work. The variants differ in the way they select the final model hypothesis: **a)** non-differentiable, vanilla RANSAC with hard, deterministic argmax selection; **b)** differentiable RANSAC with deterministic, soft argmax selection; **c)** differentiable RANSAC with hard, probabilistic selection (named DSAC). Nodes shown as boxes represent deterministic functions, while circular nodes with yellow background represent probabilistic functions. Arrows indicate dependency in computation. All differences between a), b) and c) are marked in red.

2. Method

2.1. Background

As a preface to explaining our method, we first briefly review the standard RANSAC algorithm for model fitting, and how it can be applied to the camera localization problem using discriminative scene coordinate regression.

Many problems in computer vision involve fitting a model to a set of data points, which in practice usually include outliers due to sensor noise and other factors. The RANSAC algorithm was specifically designed to be able to fit models robustly in the presence of noise [11]. Dozens of variations of RANSAC exist [39, 8, 28, 7]. We consider a general, basic variant here but the new principles presented in this work can be applied to many RANSAC variants, such as to locally-refined preemptive RANSAC [36].

A basic RANSAC implementation consists of four steps: (i) generate a set of model hypotheses by sampling minimal subsets of the data; (ii) score hypotheses based on some measure of consensus, *e.g.* by counting inliers; (iii) select the best scoring hypothesis; (iv) refine the selected hypothesis using additional data points, *e.g.* the full set of inliers. Step (iv) is optional, though in practice important for high accuracy.

We introduce our notation below using the example application of camera localization. We consider an RGB image I consisting of pixels indexed by i. We wish to estimate the parameters $\tilde{\mathbf{h}}$ of a model that explains I. In the camera localization problem this is the 6D camera pose, *i.e.* the 3D rotation and 3D translation of the camera relative to the scene's coordinate frame. Following [36], we do not fit model $\tilde{\mathbf{h}}$ directly to image data I, but instead make use of intermediate, noisy 2D-3D correspondences predicted for each pixel: $Y(I) = \{\mathbf{y}(I,i) | \forall i\}$, where $\mathbf{y}(I,i)$ is the 'scene coordinate' of pixel i, *i.e.* a discriminative prediction for where the point imaged at pixel i lives in the 3D scene coordinate frame. We will use $\mathbf{y}_i$ as shorthand for $\mathbf{y}(I,i)$. $Y(I)$ denotes the complete set of scene coordinate predictions for image I, and we write Y for $Y(I)$. To estimate $\tilde{\mathbf{h}}$ from Y we apply RANSAC as follows:

1. **Generate a pool of hypotheses.** Each hypothesis is generated from a subset of correspondences. This subset contains the minimal number of correspondences to compute a unique solution. We call this a minimal set Y_J with correspondence indices $J = \{j_1, ..., j_n\}$, where n is the minimal set size. To create the set, we uniformly sample n correspondence indices: $j_m \in [1, \ldots, |Y|]$ to get $Y_J := \{\mathbf{y}_{j_1}, ..., \mathbf{y}_{j_n}\}$. We assume a function $\mathbf{H}$ which generates a model hypothesis as $\mathbf{h}_J = \mathbf{H}(Y_J)$ from the minimal set Y_J. In our application, $\mathbf{H}$ is the perspective-n-point (PNP) algorithm [12], and $n = 4$.
2. **Score hypotheses.** Scalar function $s(\mathbf{h}_J, Y)$ measures the consensus / quality of hypothesis $\mathbf{h}_J$, *e.g.* by count-

ing inlier correspondences. To define an inlier in our application, we first define the reprojection error of scene coordinate $\mathbf{y}_i$:

$$e_i = \|\mathbf{p}_i - C\mathbf{h}_J\mathbf{y}_i\|, \tag{1}$$

where $\mathbf{p}_i$ is the 2D location of pixel i and C is the camera projection matrix. We call $\mathbf{y}_i$ an inlier if $e_i < \tau$, where τ is the inlier threshold. In this work, instead of counting inliers, we to aim to learn $s(\mathbf{h}_J, Y)$ to directly regress the hypothesis score from reprojection errors e_i, as we will explain shortly.

3. **Select best hypothesis.** We take

$$\mathbf{h}_{\text{AM}} = \underset{\mathbf{h}_J}{\text{argmax}}\, s(\mathbf{h}_J, Y)\ . \tag{2}$$

4. **Refine hypothesis.** $\mathbf{h}_{\text{AM}}$ is refined using function $\mathbf{R}(\mathbf{h}_{\text{AM}}, Y)$. Refinement may use all correspondences Y. A common approach is to select a set of inliers from Y and recalculate function $\mathbf{H}$ on this set. The refined pose is the output of the algorithm $\tilde{\mathbf{h}}_{\text{AM}} = \mathbf{R}(\mathbf{h}_{\text{AM}}, Y)$.

2.2. Learning in a RANSAC Pipeline

The system of Shotton *et al*. [36] had a single learned component, namely the regression forest that made the predictions $\mathbf{y}(I, i)$. Krull *et al*. [23] extended the approach to also learn the scoring function $s(\mathbf{h}_J, Y)$ as a generalization of the simpler inlier counting scheme of [36]. However, these have thus far been learned separately.

Our work instead aims to learn both, the scene coordinate predictions and the scoring function, and to do so jointly in an end-to-end fashion within a RANSAC framework. Making the parameterizations explicit, we have $\mathbf{y}(I, i; \mathbf{w})$ and $s(\mathbf{h}_J, Y; \mathbf{v})$. We aim to learn parameters $\mathbf{w}$ and $\mathbf{v}$, where $\mathbf{w}$ affects the quality of poses that we generate, and $\mathbf{v}$ affects the selection process which should choose a good hypothesis. We write $Y^{\mathbf{w}}$ to reflect that scene coordinate predictions depend on parameters $\mathbf{w}$. Similarly, we write $\mathbf{h}_{\text{AM}}^{\mathbf{w},\mathbf{v}}$ to reflect that the chosen hypothesis depends on $\mathbf{w}$ and $\mathbf{v}$.

We would like to find parameters $\mathbf{w}$ and $\mathbf{v}$ such that the loss ℓ of the final, refined hypotheses over a training set of images $\mathcal{I}$ is minimized, *i.e*.

$$\tilde{\mathbf{w}}, \tilde{\mathbf{v}} = \underset{\mathbf{w},\mathbf{v}}{\text{argmin}} \sum_{I \in \mathcal{I}} \ell(\mathbf{R}(\mathbf{h}_{\text{AM}}^{\mathbf{w},\mathbf{v}}, Y^{\mathbf{w}}), \mathbf{h}^*), \tag{3}$$

where $\mathbf{h}^*$ are ground truth model parameters for I. To allow end-to-end learning, we need to differentiate w.r.t. $\mathbf{w}$ and $\mathbf{v}$. We assume a differentiable loss ℓ and differentiable refinement $\mathbf{R}$.

One might consider differentiating $\mathbf{h}_{\text{AM}}^{\mathbf{w},\mathbf{v}}$ w.r.t. to $\mathbf{w}$ via the minimal set Y_J of the single selected hypothesis of Eq. 2. But learning a RANSAC pipeline in this fashion fails because the selection process itself depends on $\mathbf{w}$ and $\mathbf{v}$, which is not represented in the gradients of the selected hypothesis.[2] Parameters $\mathbf{v}$ influence the selection directly via the scoring function $s(\mathbf{h}, Y; \mathbf{v})$, and parameters $\mathbf{w}$ influence the quality of competing hypotheses $\mathbf{h}$, though neither influence the initial uniform sampling of minimal sets Y_J.

We next present two approaches to learn parameters $\mathbf{w}$ and $\mathbf{v}$ – soft argmax selection (Sec. 2.2.1) and probabilistic selection (Sec. 2.2.2) – that do model the dependency of the selection process on the parameters.

2.2.1 Soft argmax Selection (SoftAM)

To solve the problem of non-differentiability, one can relax the argmax operator of Eq. 2 and substitute it for a soft argmax operator [6]. The soft argmax turns the hypothesis selection into a weighted average of hypotheses:

$$\mathbf{h}_{\text{SoftAM}}^{\mathbf{w},\mathbf{v}} = \sum_J P(J|\mathbf{v}, \mathbf{w})\mathbf{h}_J^{\mathbf{w}} \tag{4}$$

which averages over candidate hypotheses $\mathbf{h}_J^{\mathbf{w}}$ with

$$P(J|\mathbf{v}, \mathbf{w}) = \frac{\exp(s(\mathbf{h}_J^{\mathbf{w}}, Y^{\mathbf{w}}; \mathbf{v}))}{\sum_{J'} \exp(s(\mathbf{h}_{J'}^{\mathbf{w}} Y^{\mathbf{w}}; \mathbf{v}))}\ . \tag{5}$$

In this variant, scoring function $s(\mathbf{h}_J^{\mathbf{w}}, Y^{\mathbf{w}}; \mathbf{v})$ has to predict weights that lead to a robust average of hypotheses (*i.e*. model parameters). This means that model parameters corrupted by outliers should receive sufficiently small weights, such that they do not affect the accuracy of $\mathbf{h}_{\text{SoftAM}}^{\mathbf{w},\mathbf{v}}$.

Substituting $\mathbf{h}_{\text{AM}}^{\mathbf{w},\mathbf{v}}$ for $\mathbf{h}_{\text{SoftAM}}^{\mathbf{w},\mathbf{v}}$ in Eq. 3 allows us to calculate gradients to learn parameters $\mathbf{w}$ and $\mathbf{v}$. We refer the reader to the supplementary materials for details.

By utilizing the soft argmax operator, we diverge from the RANSAC principle of making one hard decision for a hypothesis. Soft argmax hypothesis selection bears similarity with an independent strain within the field of robust optimization, namely robust averaging, see *e.g*. the work of Hartley *et al*. [15]. While we explore soft argmax selection in the experimental evaluation, we introduce an alternative in the next section, that preserves the hard hypothesis selection, and is empirically superior for our task.

2.2.2 Probabilistic Selection (DSAC)

We substitute the deterministic selection of the highest scoring model hypothesis in Eq. 2 by a probabilistic selection, *i.e*. we chose a hypothesis probabilistically according to:

$$\mathbf{h}_{\text{DSAC}}^{\mathbf{w},\mathbf{v}} = \mathbf{h}_J^{\mathbf{w}}, \text{ with } J \sim P(J|\mathbf{v}, \mathbf{w}), \tag{6}$$

where $P(J|\mathbf{v}, \mathbf{w})$ is the softmax distribution of scores predicted by $s(\mathbf{h}_J^{\mathbf{w}}, Y^{\mathbf{w}}; \mathbf{v})$ (see Eq. 5).

[2] We observed in early experiments that the training loss immediately increases without recovering.

The inspiration for this approach comes from policy gradient approaches in reinforcement learning that involve the minimization of a loss function defined over a stochastic process [34]. Similarly, we are able to learn parameters $\mathbf{w}$ and $\mathbf{v}$ that minimize the expectation of loss of the stochastic process defined in Eq. 6:

$$\tilde{\mathbf{w}}, \tilde{\mathbf{v}} = \underset{\mathbf{w},\mathbf{v}}{\operatorname{argmin}} \sum_{I \in \mathcal{I}} \mathbb{E}_{J \sim P(J|\mathbf{v},\mathbf{w})} \left[\ell(\mathbf{R}(\mathbf{h}_J^{\mathbf{w}}, Y^{\mathbf{w}})) \right]. \quad (7)$$

As shown in [34], we can calculate the derivative w.r.t. parameters $\mathbf{w}$ as follows (similarly for parameters $\mathbf{v}$):

$$\frac{\partial}{\partial \mathbf{w}} \mathbb{E}_{J \sim P(J|\mathbf{v},\mathbf{w})} \left[\ell(\cdot) \right] = \mathbb{E}_{J \sim P(J|\mathbf{v},\mathbf{w})} \left[\ell(\cdot) \frac{\partial}{\partial \mathbf{w}} \log P(J|\mathbf{v},\mathbf{w}) + \frac{\partial}{\partial \mathbf{w}} \ell(\cdot) \right], \quad (8)$$

i.e. the derivative of the expectation is an expectation over derivatives of the loss and the log probabilities of model hypotheses. We inlcude further steps of the derivation of Eq. 8 in the supplementary materials.

We call this method of differentiating RANSAC, that preserves hard hypothesis selection, DSAC – Differentiable SAmple Consensus. See Fig. 1 for a schematic view of DSAC in comparison to the RANSAC variants introduced at the beginning of this section. While learning parameters with the vanilla RANSAC is not possible, as mentioned before, both new variants (SoftAM and DSAC) are sensible options which we evaluate in the experimental section.

3. Differentiable Camera Localization

We demonstrate the principles for differentiating RANSAC for the task of one-shot camera localization from an RGB image. Our pipeline is inspired by the state-of-the-art pipeline of Brachmann *et al*. [5], which is an extension of the original SCoRF pipeline [36] from RGB-D to RGB images. Brachmann *et al*. use an auto-context random forest to predict multi-modal scene coordinate distributions per image patch. After that, minimal sets of four scene coordinates are randomly sampled and the PNP algorithm [12] is applied to create a pool of camera pose hypotheses. A preemptive RANSAC schema iteratively refines, re-scores and rejects hypotheses until only one remains. The preemptive RANSAC scores hypotheses by counting inlier scene coordinates, *i.e*. scene coordinates $\mathbf{y}_i$ for which reprojection error $e_i < \tau$. In a last step, the final, remaining hypothesis is further optimized using the uncertainty of the scene coordinate distributions.

Our pipeline differs from Brachmann *et al*. [5] in the following aspects:

- Instead of a random forest, we use a CNN (called 'Coordinate CNN' below) to predict scene coordinates. For each 42x42 pixel image patch, it predicts a scene coordinate point estimate. We use a VGG style architecture with 13 layers and 33M parameters. To reduce test time we process only 40x40 patches per image.
- We score hypotheses using a second CNN (called 'Score CNN' below). We took inspiration from the work of Krull *et al*. [23] for the task of object pose estimation. Instead of learning a CNN to compare rendered and observed images as in [23], our Score CNN predicts hypothesis consensus based on reprojection errors. For each of the 40x40 scene coordinate predictions $\mathbf{y}_i$ we calculate the reprojection error e_i for hypothesis $\mathbf{h}_J$ (see Eq. 1). This results in a 40x40 reprojection error image, which we feed into the Score CNN, a VGG style architecture with 13 layers and 6M parameters.
- Instead of the preemptive RANSAC schema, we score hypotheses only once and select the final pose, either by applying the soft argmax operator (SoftAM), or by probabilistic selection according to the softmaxed scores (DSAC).
- Only the final pose is refined. We choose inlier object coordinate predictions (at most 100), *i.e*. scene coordinates $\mathbf{y}_i$ with reprojection error $e_i < \tau$, and solve PNP [24] again using this set. This is iterated multiple times. Since the Coordinate CNN predicts only point estimates we do no further pose optimization using uncertainty.

See Fig. 2 for an overview of our pipeline. Where applicable we use the parameter values reported by Brachmann *et al*. in [5], *e.g*. sampling 256 hypotheses, using 8 refinement steps and an inlier threshold of $\tau = 10$px.

4. Experiments

For comparability to other methods, we show results on the widely used 7-Scenes dataset [36]. The dataset consists of RGB-D images of 7 indoor environments where each frame is annotated with its 6D camera pose. A 3D model of each scene is also available. The data of each scene is comprised of multiple sequences (= independent camera paths) which are assigned either to test or training. The number of images per scene ranges from 1k to 7k for training resp. test. We omit the depth channels and estimate poses using RGB images only. See the supplementary materials for a discussion of the difficulty of the 7-Scenes dataset.

We measure accuracy by the percentage of images for which the pose error is below 5° and 5cm. For training, we use the following differentiable loss which is closely correlated with the task loss:

$$\ell_{\text{pose}}(\mathbf{h}, \mathbf{h}^*) = \max(\measuredangle(\boldsymbol{\theta}, \boldsymbol{\theta}^*), \|\mathbf{t} - \mathbf{t}^*\|), \quad (9)$$

where $\mathbf{h} = (\boldsymbol{\theta}, \mathbf{t})$, $\boldsymbol{\theta}$ denotes the axis-angle representation of the camera rotation, and $\mathbf{t}$ is the camera translation.

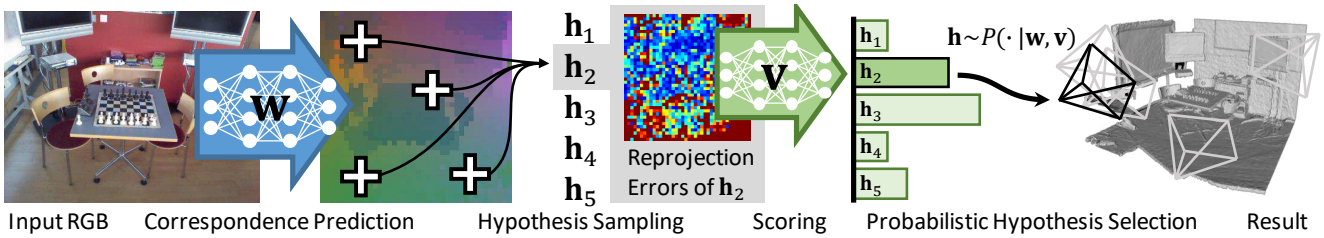

Figure 2. **Differentiable Camera Localization Pipeline.** Given an RGB image, we let a CNN with parameters $\mathbf{w}$ predict 2D-3D correspondences, so called scene coordinates [36]. From these, we sample minimal sets of four scene coordinates and create a pool of hypotheses $\mathbf{h}$. For each hypothesis, we create an image of reprojection errors which is scored by a second CNN with parameters $\mathbf{v}$. We select a hypothesis probabilistically according to the score distribution. The selected pose is also refined.

We measure angle $\measuredangle(\boldsymbol{\theta}, \boldsymbol{\theta}^*)$ between estimated and ground truth rotation in degree, and distance $\|\mathbf{t} - \mathbf{t}^*\|$ between estimated and ground truth translation in cm.

Since the dataset does not include a designated validation set, we separated multiple blocks of 100 consecutive frames from the training data to be used as validation data (in total 10% per scene). We fixed all learning parameters on the validation set (*e.g.* learning rate and total amount of parameter updates). Once all hyper parameters are fixed, we re-train on the full training set.

4.1. Componentwise Training

Our pipeline contains two trainable components, namely the Coordinate CNN and the Score CNN. First, we explain how to train both components using surrogate losses, *i.e.* train them not in an end-to-end fashion but separately. End-to-end training using differentiable RANSAC will be discussed in Sec. 4.2.

Scene Coordinate Regression. Similar to Brachmann *et al.* [5], we use the depth information of training images to generate scene coordinate ground truth. Alternatively, this ground truth can also be rendered using the available 3D models. We train the Coordinate CNN using the following surrogate loss: $\ell_{\text{coord}}(\mathbf{y}, \mathbf{y}^*) = \|\mathbf{y} - \mathbf{y}^*\|$, where $\mathbf{y}$ is the scene coordinate prediction and $\mathbf{y}^*$ is ground truth. We also experimented with other losses including L_2 (squared distance), Huber [18] and Tukey [3] which consistently performed worse on the validation set.

We trained with mini batches of 64 randomly sampled training patches. We used the Adam [21] optimizer with a learning rate of 10^{-4}. We cut the learning rate in half after each 50k updates, and train for a total of 300k updates.

Score Regression. We synthetically created data to train the Score CNN in the following way. By adding noise to the ground truth pose of training images, we generated poses above and below the pose error threshold of 5° and 5cm. Using the scene coordinate predictions of the trained Coordinate CNN, we compute reprojection error images of these poses. Poses with a large pose error w.r.t. the ground truth pose will lead to large reprojection errors, and we want the Score CNN to predict a small score. Poses close to ground truth will lead to small reprojection errors, and we want the Score CNN to predict a high score. More formally, the pose error $\ell_{\text{pose}}(\mathbf{h}, \mathbf{h}^*)$ of a hypothesis $\mathbf{h}$ should be negatively correlated with the score prediction $s(\mathbf{h}, Y; \mathbf{v})$. Thus, we train the Score CNN to minimize the following loss: $\ell_{\text{score}}(s, s^*) = |s - s^*|$, where: $s^* = -\beta \ell_{\text{pose}}(\mathbf{h}, \mathbf{h}^*)$. Parameter β controls the broadness of the score distribution after applying softmax. We use this distribution for weights in SoftAM (see Eq. 5) and to sample a hypothesis in DSAC (see Eq. 6). A value of $\beta = 10$ gave reasonable distributions on the validation set, *i.e.* poses close to ground truth had a high probability to be selected, and poses far away from ground truth had a low probability to be selected.

We trained the Score CNN with a batch size of 64 reprojection error images of randomly generated poses. We used Adam [21] for optimization with a learning rate of 10^{-4}. We train for a total of 2k updates.

Table 2. **Median pose errors** of the complete 7-Scenes dataset (17000 frames). Most accurate result marked **bold**.

	Brachmann *et al.* [5]	4.5cm, 2.0°
Ours, Trained Componentwise	RANSAC	4.2cm, 1.1°
	SoftAM	4.2cm, 1.1°
	DSAC	4.3cm, 1.1°
Ours, Trained End-To-End	SoftAM	4.5cm, 1.2°
	DSAC	**4.1cm, 1.1°**

Results. We report the accuracy of our pipeline, trained componentwise, in Table 1. We present the accuracy per scene and the average over scenes. Since scenes with few test frames like *Stairs* and *Heads* are overrepresented in the average, we additionally show accuracy on the dataset as a whole (denoted *Complete*, *i.e.* 17000 test frames).

We distinguish between *RANSAC*, *i.e.* non-differentiable argmax hypothesis selection, *SoftAM*, *i.e.* differentiable soft argmax hypothesis selection and *DSAC*, *i.e.* differentiable probabilistic hypothesis selection.

As can be seen in Table 1, RANSAC, SoftAM and DSAC

Table 1. Accuracy measured as the percentage of test images where the pose error is below 5cm and 5°. *Complete* denotes the combined set of frames (17000) of all scenes. Numbers in green denote improved accuracy after end-to-end training for SoftAM resp. DSAC compared to componentwise training. Similarly, red numbers denote decreased accuracy. **Bold** numbers indicate the best result for each scene.

			Ours				
	Sparse Features [36]	Brachmann *et al.* [5]	Trained Componentwise			Trained End-To-End	
			RANSAC	SoftAM	DSAC	SoftAM	DSAC
Chess	70.7%	94.9%	96.8%	96.8%	97.1%	97.3% +0.5%	**97.4%** +0.3%
Fire	49.9%	**73.5%**	71.8%	72.0%	71.4%	71.9% -0.1%	71.6% +0.2%
Heads	67.6%	48.1%	66.7%	67.3%	68.5%	**67.9%** +0.6%	67.0% -1.5%
Office	36.6%	53.2%	57.6%	58.5%	57.4%	47.8% -10.7%	**59.4%** +2.0%
Pumpkin	21.3%	54.5%	**59.0%**	58.7%	57.6%	57.0% -1.7%	58.3% +0.7%
Kitchen	29.8%	42.2%	40.1%	40.4%	38.6%	40.2% -0.2%	**42.7%** +4.1%
Stairs	9.2%	**20.1%**	12.8%	13.5%	13.7%	12.3% -1.2%	13.4% -0.3%
Average	40.7%	55.2%	57.8%	58.2%	57.7%	56.3% -1.4%	**58.5%** +0.8%
Complete	38.6%	55.2%	56.8%	57.2%	56.3%	54.4% -2.8%	**58.0%** +1.7%

achieve very similar results when trained componentwise. The probabilistic hypothesis selection of DSAC results in a slightly reduced accuracy of -0.5% on the complete dataset, compared to RANSAC.

We compare our pipeline to the sparse features baseline presented in [36] and the pipeline of Brachmann *et al.* [5], which is state-of-the-art on this dataset at the moment. All variants of our pipeline surpass, on average, the accuracy of both competitors. Note, conceptually the main advantage over Brachmann *et al.* [5] is the new scoring CNN. We also measured the median pose error of all frames in the dataset, see Table 2. Compared to Brachmann *et al.* [5] we are able to decrease both rotational and translational error. PoseNet [20] states median translational errors of around 40cm per scene, so it cannot compete in terms of accuracy.

4.2. End-to-End Training

In order to facilitate end-to-end learning as described in Sec. 2, some parts of the pipeline need to be differentiable which might not be immediately obvious. We already introduced the differentiable loss ℓ_{pose}. Furthermore, we need to derive the model function $\mathbf{H}(Y_J)$ and refinement $\mathbf{R}$ w.r.t. learnable parameters.

In our application, $\mathbf{H}(Y_J)$ is the PNP algorithm. Off-the-shelf implementations (*e.g.* [12, 24]) are fast enough for calculating the derivatives via central differences.

Refinement $\mathbf{R}$ involves determining inlier sets and resolving PNP in multiple iterations. This procedure in non-differentiable because of the hard inlier selection procedure. However, because the number of inliers is large (100 in our case), refined poses tend to vary smoothly with changes to the input scene coordinates. Hence, we treat the refinement procedure as a black box, and calculate derivatives via central differences, as well. For stability, we stop refinement early, in case less than 50 inliers have been found. Because of the large number of inputs and to keep central differences tractable, we subsample the scene coordinates for which gradients are calculated (we use 1%), and correct the gradient magnitude accordingly ($\times 100$).

Similar to *e.g.* [41] or [20], we found it important to have a good initialization when learning end-to-end. Learning from scratch quickly reached a local minimum. Hence, we initialize the Coordinate CNN and the Score CNN with componentwise training, see Sec. 4.1.

We found the same set of training hyperparameters to work well for the validation set for both, SoftAM and DSAC. We use the following learning rate schedule for the Coordinate CNN: $\alpha_t = 10^{-4}/(1 + 0.1t)$ where α_t is the learning rate at iteration t. For the Score CNN we use a fixed learning rate of 10^{-7}. Our end-to-end pipeline contains substantial stochasticity because of the sampling of minimal sets Y_J. Instead of the Adam procedure, which was unstable, we use stochastic gradient descent with momentum [33] of 0.9, and we clamp all gradients to the range of -0.1 to 0.1, before passing them to the Score CNN or the Coordinate CNN. We train for 10k updates.

Results. See Table 1 for results of both strategies. Compared to the initialization (trained componentwise), we observe a significant improvement for DSAC (+1.7% on the complete dataset, standard error of the mean $\pm 0.4\%$). DSAC improves accuracy for most scenes, with strongest effects for *Office* (+2.0%) and *Kitchen* (+4.1%). SoftAM significantly decreases accuracy compared to the componentwise initialization (-2.8% on the complete dataset). SoftAM overfits severely on the *Office* scene (-10.7%) and decreases accuracy for most other scenes.

The pipeline learned end-to-end with DSAC improves on the results of Brachmann *et al.* [5] by 3.3% (scene average) resp. 2.8% (complete set). DSAC also improves the median pose error, see Table 2.

4.3. Insights and Detailed Studies

Ablation Study. We study the effect of learning the Score CNN and the Coordinate CNN in an end-to-end fashion, in-

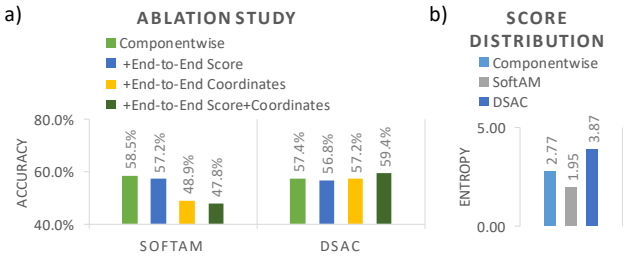

Figure 3. **(a)** Effect of end-to-end learning on pose accuracy w.r.t. individual components. **(b)** Effect of end-to-end training on the average entropy of the score distribution. We used the *Office* test set for both studies. Set text for details.

dividually. We use componentwise training as initialization for both CNNs. See Fig. 3 a) for results on the *Office* scene. For DSAC, training both components in an end-to-end fashion is important for best accuracy. For SoftAM, we see that the bad results on this scene are not due to overfitting on the Score CNN, but its way of learning the Coordinate CNN.

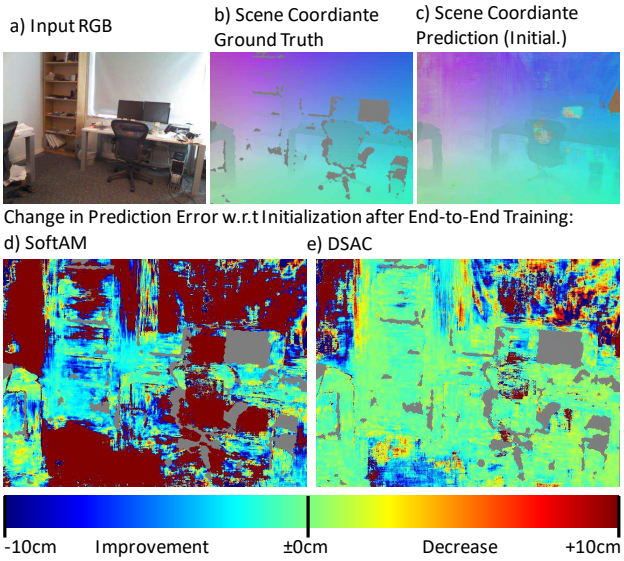

Figure 4. **Prediction quality.** We analyze scene coordinate prediction quality on an *Office* test image **(a)** with ground truth scene coordinates **(b)** (XYZ mapped to RGB). The prediction after componentwise training can be seen in **(c)**. We vizualize the relative change of prediction error w.r.t. componentwise training in **(d)** for SoftAM, resp. in **(e)** for DSAC. We observe an aggressive strategy of SoftAM which focuses large improvements on small areas (14% of predictions improve). DSAC shows small improvements but on large areas (38% of predictions improve). Note that DSAC achieves superior pose accuracy on this scene.

Analysis of Scene Coordinate Predictions. In the componentwise training, the Coordinate CNN learned to minimize the surrogate loss ℓ_{coord}, *i.e.* the distance $\|\mathbf{y}_i - \mathbf{y}_i^*\|$ of scene coordinate predictions $\mathbf{y}_i$ w.r.t. ground truth $\mathbf{y}_i^*$. In Fig. 4, we visualize how the prediction of the Coordinate CNN changes when trained in an end-to-end fashion, *i.e.* to minimize the loss ℓ_{pose}. Both end-to-end learning strategies, SoftAM and DSAC, increase the accuracy of scene coordinate predictions in some areas of the scene at the cost of decreasing the accuracy in other areas. We observe very extreme changes for the SoftAM strategy, *i.e.* the increase and decrease in scene coordinate accuracy is large in magnitude, and improvements are focused to small scene areas. The DSAC strategy leads to a much more cautious tradeoff, *i.e.* changes are smaller and widespread. Note that we use identical learning parameters for both strategies. We conclude that SoftAM tends to overfit due to overly aggressive changes in scene coordinate predictions.

Score Distribution Entropy. See Fig. 3 b) for an analysis of the effect of end-to-end learning on the average entropy of the softmax score distribution (see Eq. 5). We observe a clear reduction in entropy for the SoftAM strategy. The larger the pose error of a hypothesis is, the larger is also its influence on the pose average (see Eq. 4). SoftAM has to weigh down such poses aggressively for a good average. DSAC can allow for a broader distribution (see the increase in entropy) because poses which are unlikely to be chosen, do not affect the loss of poses which are likely to be chosen. This is an additional factor in the stability of DSAC.

Restoring the argmax **Selection.** After end-to-end training, one may restore the original RANSAC algorithm, *e.g.* selecting hypotheses w.r.t. scores via argmax. In this case, the average accuracy of DSAC stays at 58.5%, while the accuracy of SoftAM decreases further to an average of 55.8%.

Further Discussions. See the supplementary materials for a discussion of run time and the potential benefits of modeling multi-modal scene coordinate distributions.

5. Conclusion

We presented two strategies for differentiating the RANSAC algorithm: Using a soft argmax operator, and probabilistic selection. By experimental evaluation we conclude that probabilistic selection is superior and call this approach DSAC. We demonstrated the use of DSAC for learning a camera localization pipeline end-to-end. However, DSAC can be deployed in any deep learning pipeline where robust optimization is beneficial, for example learning structure from motion or SLAM end-to-end.

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

References

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

[2] R. Arandjelovic and A. Zisserman. All about VLAD. In *CVPR*, 2013.

[3] A. E. Beaton and J. W. Tukey. The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data. *Technometrics*, 1974.

[4] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6D object pose estimation using 3D object coordinates. In *ECCV*, 2014.

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

[7] O. Chum and J. Matas. Matching with PROSAC - Progressive sample consensus. In *CVPR*, 2005.

[8] O. Chum, J. Matas, and J. Kittler. Locally optimized RANSAC. In *DAGM*, 2003.

[9] D. DeTone, T. Malisiewicz, and A. Rabinovich. Deep image homography estimation. *CoRR*, 2016.

[10] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015.

[11] 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.

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

[13] R. Girshick. Fast R-CNN. In *ICCV*, 2015.

[14] 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.

[15] R. Hartley, K. Aftab, and J. Trumpf. L1 rotation averaging using the Weiszfeld algorithm. In *CVPR*, 2011.

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

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, 2015.

[18] P. J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 1964.

[19] A. Kanazawa, D. W. Jacobs, and M. Chandraker. WarpNet: Weakly supervised matching for single-view reconstruction. *CoRR*, 2016.

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

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, 2014.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[23] A. Krull, E. Brachmann, F. Michel, M. Y. Yang, S. Gumhold, and C. Rother. Learning analysis-by-synthesis for 6D pose estimation in RGB-D images. In *ICCV*, 2015.

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

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

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

[27] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: A versatile and accurate monocular SLAM system. *CoRR*, 2015.

[28] D. Nistér. Preemptive RANSAC for live structure and motion estimation. In *ICCV*, 2003.

[29] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.

[30] R. Raguram, J.-M. Frahm, and M. Pollefeys. A comparative analysis of RANSAC techniques leading to adaptive real-time random sample consensus. In *ECCV*, 2008.

[31] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, 2015.

[32] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. DeepMatching: Hierarchical deformable dense matching. *IJCV*, 2016.

[33] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 1988.

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

[35] S. Shalev-Shwartz and A. Shashua. On the sample complexity of end-to-end training vs. semantic abstraction training. *CoRR*, 2016.

[36] 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.

[37] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2014.

[38] J. Thewlis, S. Zheng, P. Torr, and A. Vedaldi. Fully-trainable deep matching. In *BMVC*, 2016.

[39] P. H. S. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *CVIU*, 2000.

[40] 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.

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

[42] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.