CONSAC: Robust Multi-Model Fitting by Conditional Sample Consensus

Florian Kluger[1], Eric Brachmann[2], Hanno Ackermann[1], Carsten Rother[2], Michael Ying Yang[3], Bodo Rosenhahn[1]
[1]Leibniz University Hannover, [2]Heidelberg University, [3]University of Twente

Abstract

We present a robust estimator for fitting multiple parametric models of the same form to noisy measurements. Applications include finding multiple vanishing points in man-made scenes, fitting planes to architectural imagery, or estimating multiple rigid motions within the same sequence. In contrast to previous works, which resorted to hand-crafted search strategies for multiple model detection, we learn the search strategy from data. A neural network conditioned on previously detected models guides a RANSAC estimator to different subsets of all measurements, thereby finding model instances one after another. We train our method supervised as well as self-supervised. For supervised training of the search strategy, we contribute a new dataset for vanishing point estimation. Leveraging this dataset, the proposed algorithm is superior with respect to other robust estimators as well as to designated vanishing point estimation algorithms. For self-supervised learning of the search, we evaluate the proposed algorithm on multi-homography estimation and demonstrate an accuracy that is superior to state-of-the-art methods.

1. Introduction

Describing 3D scenes by low-dimensional parametric models, oftentimes building upon simplifying assumptions, has become fundamental to reconstructing and understanding the world around us. Examples include: i) fitting 3D-planes to an architectural scene, which relates to finding multiple homographies in two views; ii) tracking rigid objects in two consecutive images, which relates to fitting multiple fundamental matrices; iii) identifying the dominant directions in a man-made environment, which relates to finding multiple vanishing points. Once such parametric models are discovered from images, they can ultimately be used in a multitude of applications and high-level vision tasks. Examples include the automatic creation of 3D models [1, 24, 48, 61], autonomous navigation [39, 47, 20, 30] or augmented reality [10, 11, 2, 45].

Model-fitting has generally been realised as a two-step procedure. Firstly, an error-prone, low-level process to extract data points which shall adhere to a model is executed. For example, one could match 2D feature points between pairs of images as a basis for homography estimation [21], in order to determine the 3D plane where the 3D points live on. Secondly, a robust estimator that fits model parameters to *inlier* data points is used, while at the same time identifying erroneous data points as so-called *outliers* [19]. Some outliers can be efficiently removed by pre-processing, e.g. based on the descriptor distances in feature matching [34].

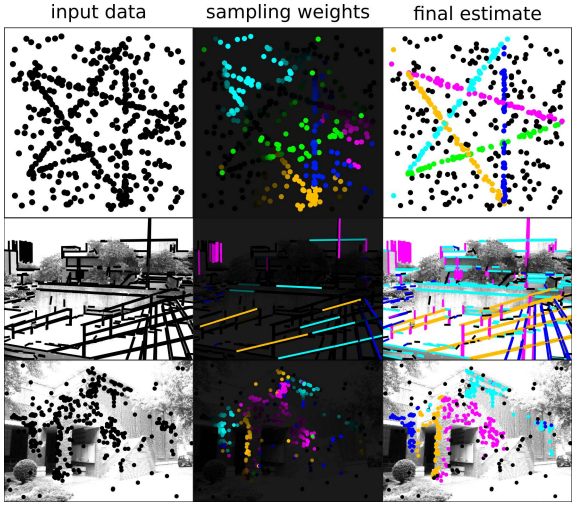

Figure 1: **CONSAC applications:** line fitting (top), vanishing point estimation (middle) and homography estimation (bottom) for multiple instances. Colour hues in column two and three indicate different instances, brightness in column two varies by sampling weight.

While the case of fitting a *single* parametric model to data has received considerable attention in the literature, we focus on the scenario of fitting *multiple* models of the same form to data. This is of high practical relevance, as motivated in the example above. There, multiple 3D planes represented by multiple homographies are fitted. However, when multiple models are present in the data, estimation becomes more challenging. Inliers of one model constitute outliers of all other models. Naturally, outlier filters fail in removing such *pseudo-outliers*.

Early approaches to multi-model fitting work *sequentially*: They apply a robust estimator like RANSAC repeatedly, removing the data points associated with the currently predicted model in each iteration [59]. Modern, state-of-the-art methods solve multi-model fitting *simultaneously* instead, by using clustering or optimisation techniques to assign data points to models or an outlier class [6, 7, 8, 42, 3, 26, 54, 35, 36, 37, 38, 13]. In our work, we revisit the idea of sequential processing, but combine it with recent advances in learning robust estimators [66, 46, 12]. Sequential processing easily lends itself to conditional sampling approaches, and with this we are able to achieve state-of-the-art results despite supposedly being conceptually inferior to simultaneous approaches.

The main inspiration of our work stems from the work of Brachmann and Rother [12], where they train a neural network to enhance the sample efficiency of a RANSAC estimator for *single model estimation*. In contrast, we investigate *multi-model fitting* by letting the neural network update sampling weights conditioned on models it has already found. This allows the neural network to not only suppress outliers, but also inliers of all but the current model of interest. Since our new RANSAC variant samples model hypotheses based on conditional probabilities, we name it *Conditional Sample Consensus* or CONSAC, in short. CONSAC, as illustrated by Fig. 1, proves to be powerful and achieves top performance for several applications.

Machine learning has been applied in the past to fitting of a single parametric model, by directly predicting model parameters from images [27, 18], replacing a robust estimator [66, 46, 52] or enhancing a robust estimator [12]. However, to the best of our knowledge, CONSAC is the first application of machine learning to robust fitting of *multiple models*.

One limiting factor of applying machine learning to multi-model fitting is the lack of suitable datasets. Previous works either evaluate on synthetic toy data [54] or few hand-labeled, real examples [63, 56, 17]. The most comprehensive and widely used dataset, AdelaideRMF [63] for homography and fundamental matrix estimation, does not provide training data. Furthermore, the test set consists of merely 38 labeled image pairs, re-used in various publications since 2011 with the danger of steering the design of new methods towards overfitting to these few examples.

We collected a new dataset for multi-model fitting, vanishing point (VP) estimation in this case, which we call *NYU-VP*[1]. Each image is annotated with up to eight vanishing points, and we provide pre-extracted line segments which act as data points for a robust estimator. Due to its size, our dataset is the first to allow for supervised learning of a multi-model fitting task. We observe that robust estimators which work well for AdelaideRMF [63], do not necessarily achieve good results for our new dataset. CONSAC not only exceeds the accuracy of these alternative robust estimators for vanishing point estimation. It also surpasses designated vanishing point estimation algorithms, which have access to the full RGB image instead of only pre-extracted line segments, on two datasets.

Furthermore, we demonstrate that CONSAC can be trained self-supervised for the task of multi-homography estimation, *i.e*. where no ground truth labelling is available. This allows us to compare CONSAC to previous robust estimators on the AdelaideRMF [63] dataset despite the lack of training data. Here, we also achieve a new state-of-the-art in terms of accuracy.

To summarise, our **main contributions** are as follows:

- CONSAC, the first learning-based method for robust *multi-model fitting*. It is based on a neural network that sequentially updates the conditional sampling probabilities for the hypothesis selection process.
- A new dataset, which we term *NYU-VP*, for vanishing point estimation. It is the first dataset to provide sufficient training data for supervised learning of a multi-model fitting task. In addition, we present *YUD+*, an extension to the York Urban Dataset [17] (YUD) with extra vanishing point labels.
- We achieve state-of-the-art results for vanishing point estimation for our new NYU-VP and YUD+ datasets. We exceed the accuracy of competing robust estimators as well as designated VP estimation algorithms.
- We achieve state-of-the-art results for multi-model homography estimation on the AdelaideRMF [63] dataset, while training CONSAC self-supervised with an external corpus of data.

2. Related Work

2.1. Multi-Model Fitting

Robust model fitting is a key problem in Computer Vision, which has been studied extensively in the past. RANSAC [19] is arguably the most commonly implemented approach. It samples minimal sets of observations to generate model hypotheses, computes the consensus sets for all hypotheses, *i.e*. observations which are consistent with a hypothesis and thus inliers, and selects the hypothesis with the largest consensus. While effective in the single-instance case, RANSAC cannot estimate multiple model instances apparent in the data. Sequential RANSAC [59] fits multiple models sequentially by applying RANSAC, removing inliers of the selected hypothesis, and repeating until a stopping criterion is reached. PEARL [26] instead fits multiple models simultaneously by optimising an energy-based functional, initialised via a stochastic sampling such as RANSAC. Several approaches based on fundamentally the same paradigm have been proposed subsequently [6, 7, 8, 42, 3]. Multi-X [6] is a generalisa-

[1]Code and datasets: https://github.com/fkluger/consac

tion to multi-class problems – *i.e.* cases where models of multiple types may fit the data – with improved efficiency, while Progressive-X [7] interleaves sampling and optimisation in order to guide hypothesis generation using intermediate estimates. Another group of methods utilises preference analysis [68] which assumes that observations explainable by the same model instance have similar distributions of residuals w.r.t. model hypotheses [54, 35, 36, 37, 38, 13]. T-Linkage [35] clusters observations by their preference sets agglomeratively, with MCT [38] being its multi-class generalisation, while RPA [36] uses spectral clustering instead. In order to better deal with intersecting models, RansaCov [37] formulates multi-model fitting as a set coverage problem. Common to all of these multi-model fitting approaches is that they mostly focus on the analysis and selection of sampled hypotheses, with little attention to the sampling process itself. Several works propose improved sampling schemes to increase the likelihood of generating accurate model hypotheses from all-inlier minimal sets [12, 5, 40, 15, 55] in the single-instance case. Notably, Brachmann and Rother [12] train a neural network to enhance the sample efficiency of RANSAC by assigning sampling weights to each data point, effectively suppressing outliers. Few works, such as the conditional sampling based on residual sorting by Chin et al. [14], or the guided hyperedge sampling of Purkait et al. [43], consider the case of multiple instances. In contrast to these handcrafted methods, we present the first learning-based conditional sampling approach.

2.2. Vanishing Point Estimation

While vanishing point (VP) estimation is part of a broader spectrum of multi-model fitting problems, a variety of algorithms specifically designed to tackle this task has emerged in the past [4, 9, 29, 32, 50, 53, 58, 62, 65, 67]. While most approaches proceed similarly to other multi-model fitting methods, they usually exploit additional, domain-specific knowledge. Zhai et al. [67] condition VP estimates on a horizon line, which they predict from the RGB image via a convolutional neural network (CNN). Kluger et al. [29] employ a CNN which predicts initial VP estimates, and refine them using a task-specific expectation maximisation [16] algorithm. Simon et al. [50] condition the VPs on the horizon line as well. General purpose robust fitting methods, such as CONSAC, do not rely on such domain-specific constraints. Incidentally, these works on VP estimation conduct evaluation using a metric which is based on the horizon line instead of the VPs themselves. As there can only be one horizon line per scene, this simplifies evaluation in presence of ambiguities w.r.t. the number of VPs, but ultimately conceals differences in performance regarding the task these methods have been designed for. By comparison, we conduct evaluation on the VPs themselves.

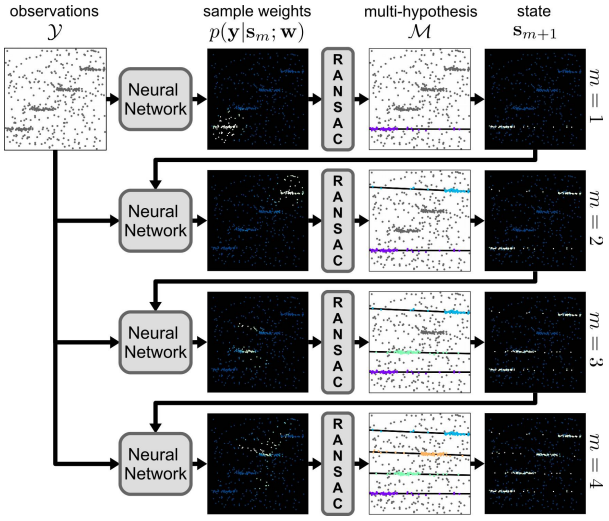

Figure 2: **Multi-Hypothesis Generation:** a neural network predicts sampling weights p for all observations conditioned on a state $\mathbf{s}$. A RANSAC-like sampling process uses these weights to select a model hypothesis and appends it to the current multi-instance hypothesis $\mathcal{M}$. The state $\mathbf{s}$ is updated based on $\mathcal{M}$ and fed into the neural network repeatedly.

3. Method

Given a set of noisy observations $\mathbf{y} \in \mathcal{Y}$ contaminated by outliers, we seek to fit M instances of a geometric model $\mathbf{h}$ apparent in the data. We denote the set of all model instances as $\mathcal{M} = \{\mathbf{h}_1, \ldots, \mathbf{h}_M\}$. CONSAC estimates $\mathcal{M}$ via three nested loops, cf. Fig. 2.

1. We generate a single model instance $\hat{\mathbf{h}}$ via RANSAC-based [19] sampling, guided by a neural network. This level corresponds to one row of Fig. 2.
2. We repeat single model instance generation while conditionally updating sampling weights. Multiple single model hypotheses compound to a *multi-hypothesis* $\mathcal{M}$. This level corresponds to the entirety of Fig. 2.
3. We repeat steps 1 and 2 to sample multiple multi-hypotheses $\mathcal{M}$ independently. We choose the best multi-hypothesis as the final multi-model estimate $\hat{\mathcal{M}}$.

We discuss these conceptional levels more formally below.

Single Model Instance Sampling We estimate parameters of a single model, e.g. one VP, from a minimal set of C observations, e.g. two line segments, using a minimal solver $f_{\mathbf{S}}$. As in RANSAC, we compute a hypothesis pool $\mathcal{H} = \{\mathbf{h}_1, \ldots, \mathbf{h}_S\}$ via random sampling of S minimal sets. We choose the best hypothesis $\hat{\mathbf{h}}$ based on a *single-instance* scoring function $g_{\mathbf{s}}$. Typically, $g_{\mathbf{s}}$ is realised as inlier counting via a residual function $r(\mathbf{y}, \mathbf{h})$ and a threshold τ.

Multi-Hypothesis Generation We repeat single model instance sampling M times to generate a full multi-

hypothesis $\mathcal{M}$, e.g. a complete set of vanishing points for an image. Particularly, we select M model instances $\hat{\mathbf{h}}_m$ from their respective hypothesis pools $\mathcal{H}_m$. Applied sequentially, previously chosen hypotheses can be factored into the scoring function g_s when selecting $\hat{\mathbf{h}}_m$:

$$\hat{\mathbf{h}}_m = \underset{\mathbf{h}\in\mathcal{H}_m}{\arg\max}\, g_\mathsf{s}(\mathbf{h}, \mathcal{Y}, \hat{\mathbf{h}}_{1:(m-1)})\,. \tag{1}$$

Multi-Hypothesis Sampling We repeat the previous process P times to generate a pool of multi-hypotheses $\mathcal{P} = \{\mathcal{M}_1, \dots \mathcal{M}_P\}$. We select the best multi-hypothesis according to a multi-instance scoring function g_m:

$$\hat{\mathcal{M}} = \underset{\mathcal{M}\in\mathcal{P}}{\arg\max}\, g_\mathsf{m}(\mathcal{M}, \mathcal{Y})\,, \tag{2}$$

where g_m measures the joint inlier count of all hypotheses in $\mathcal{M}$, and where the m in g_m stands for *multi-instance*.

3.1. Conditional Sampling

RANSAC samples minimal sets uniformly from $\mathcal{Y}$. For large amounts of outliers in $\mathcal{Y}$, the number of samples S required to sample an outlier-free minimal set with reasonable probability grows exponentially large. Brachmann and Rother [12] instead sample observations according to a categorical distribution $\mathbf{y} \sim p(\mathbf{y}; \mathbf{w})$ parametrised by a neural network $\mathbf{w}$. The neural network biases sampling towards outlier-free minimal sets which generate accurate hypotheses $\hat{\mathbf{h}}$. While this approach is effective in the presence of outliers, it is not suitable for dealing with pseudo-outliers posed by multiple model instances. Sequential RANSAC [59] conditions the sampling on previously selected hypotheses, *i.e.* $\mathbf{y} \sim p(\mathbf{y}|\{\hat{\mathbf{h}}_1, \dots, \hat{\mathbf{h}}_{m-1}\})$, by removing observations already deemed as inliers from $\mathcal{Y}$ after each hypothesis selection. While being able to reduce pseudo-outliers for subsequent instances, this approach can neither deal with pseudo-outliers in the first sampling step, nor with gross outliers in general. Instead, we parametrise the conditional distribution by a neural network $\mathbf{w}$ conditioned on a state $\mathbf{s}$: $\mathbf{y} \sim p(\mathbf{y}|\mathbf{s}; \mathbf{w})$.

The state vector $\mathbf{s}_m$ at instance sampling step m encodes information about previously sampled hypotheses in a meaningful way. We use the inlier scores of all observations w.r.t. all previously selected hypotheses as the state $\mathbf{s}_m$. We define the state entry $s_{m,i}$ of observation $\mathbf{y}_i$ as:

$$s_{m,i} = \max_{j\in[1,m)} g_\mathsf{y}(\mathbf{y}_i, \hat{\mathbf{h}}_j)\,, \tag{3}$$

with g_y gauging if $\mathbf{y}$ is an inlier of model $\mathbf{h}$. See the last column of Fig. 2 for a visualisation of the state. We sample multi-instance hypothesis pools independently:

$$p(\mathcal{P}; \mathbf{w}) = \prod_{i=1}^{P} p(\mathcal{M}_i; \mathbf{w})\,, \tag{4}$$

while conditioning multi-hypotheses on the state $\mathbf{s}$:

$$\begin{aligned} p(\mathcal{M}; \mathbf{w}) &= \prod_{m=1}^{M} p(\mathcal{H}_m|\mathbf{s}_m; \mathbf{w})\,, \\ \text{with}\quad p(\mathcal{H}|\mathbf{s}; \mathbf{w}) &= \prod_{s=1}^{S} p(\mathbf{h}_s|\mathbf{s}; \mathbf{w})\,, \\ \text{with}\quad p(\mathbf{h}|\mathbf{s}; \mathbf{w}) &= \prod_{c=1}^{C} p(\mathbf{y}_c|\mathbf{s}; \mathbf{w})\,. \end{aligned} \tag{5}$$

Note that we do not update state $\mathbf{s}$ while sampling single instance hypotheses pools $\mathcal{H}$, but only within sampling of multi-hypotheses $\mathcal{M}$. We provide details of scoring functions g_y, g_m and g_s in the appendix.

3.2. Neural Network Training

Neural network parameters $\mathbf{w}$ shall be optimised in order to increase chances of sampling outlier- and pseudo-outlier-free minimal sets which result in accurate, complete and duplicate-free multi-instance estimates $\hat{\mathcal{M}}$. As in [12], we minimise the expectation of a task loss $\ell(\hat{\mathcal{M}})$ which measures the quality of an estimate:

$$\mathcal{L}(\mathbf{w}) = \mathbb{E}_{\mathcal{P}\sim p(\mathcal{P};\mathbf{w})}\left[\ell(\hat{\mathcal{M}})\right]\,. \tag{6}$$

In order to update the network parameters $\mathbf{w}$, we approximate the gradients of the expected task loss:

$$\frac{\partial}{\partial\mathbf{w}}\mathcal{L}(\mathbf{w}) = \mathbb{E}_{\mathcal{P}}\left[\ell(\hat{\mathcal{M}})\frac{\partial}{\partial\mathbf{w}}\log p(\mathcal{P}; \mathbf{w})\right]\,, \tag{7}$$

by drawing K samples $\mathcal{P}_k \sim p(\mathcal{M}; \mathbf{w})$:

$$\frac{\partial}{\partial\mathbf{w}}\mathcal{L}(\mathbf{w}) \approx \frac{1}{K}\sum_{k=1}^{K}\left[\ell(\hat{\mathcal{M}}_k)\frac{\partial}{\partial\mathbf{w}}\log p(\mathcal{P}_k; \mathbf{w})\right]\,. \tag{8}$$

As we can infer from Eq. 7, neither the loss ℓ, nor the sampling procedure for $\hat{\mathcal{M}}$ need be differentiable. As in [12], we subtract the mean loss from ℓ to reduce variance.

3.2.1 Supervised Training

If ground truth models $\mathcal{M}^\mathsf{gt} = \{\mathbf{h}_1^\mathsf{gt}, \dots, \mathbf{h}_G^\mathsf{gt}\}$ are available, we can utilise a task-specific loss $\ell_\mathsf{s}(\hat{\mathbf{h}}, \mathbf{h}^\mathsf{gt})$ measuring the error between a single ground truth model $\mathbf{m}$ and an estimate $\hat{\mathbf{h}}$. For example, ℓ_s may measure the angle between an estimated and a true vanishing direction. First, however, we need to find an assigment between $\mathcal{M}^\mathsf{gt}$ and $\hat{\mathcal{M}}$. We compute a cost matrix $\mathbf{C}$, with $C_{ij} = \ell_\mathsf{s}(\hat{\mathbf{h}}_i, \mathbf{h}_j^\mathsf{gt})$, and define the multi-instance loss as the minimal cost of an assignment obtained via the Hungarian method [31] f_H: $\ell(\hat{\mathcal{M}}, \mathcal{M}^\mathsf{gt}) = f_\mathsf{H}(\mathbf{C}_{1:\min(M,G)})$. Note that we only consider *at most* G model estimates $\hat{\mathbf{h}}$ which have been selected first, regardless of how many estimates M were generated, *i.e.* this loss encourages early selection of good model hypotheses, but does not penalise bad hypotheses later on.

3.2.2 Self-supervised Training

In absence of ground-truth labels, we can train CONSAC in a self-supervised fashion by replacing the task loss with another quality measure. We aim to maximise the average joint inlier counts of the selected model hypotheses:

$$g_{\mathsf{ci}}(\hat{\mathbf{h}}_m, \mathcal{Y}) = \frac{1}{|\mathcal{Y}|} \sum_{i=1}^{|\mathcal{Y}|} \max_{j \in [1,m]} g_{\mathsf{i}}(\mathbf{y}_i, \hat{\mathbf{h}}_j)\,. \tag{9}$$

We then define our self-supervised loss as:

$$\ell_{\mathsf{self}}(\hat{\mathcal{M}}) = -\frac{1}{M} \sum_{m=1}^{M} g_{\mathsf{ci}}(\hat{\mathbf{h}}_m, \mathcal{Y})\,. \tag{10}$$

Eq. 9 monotonically increases w.r.t. m, and has its minimum when the models in $\hat{\mathcal{M}}$ induce the largest possible minimally overlapping inlier sets descending in size.

Inlier Masking Regularisation For self-supervised training, we found it empirically beneficial to add a weighted regularisation term $\kappa \cdot \ell_{\mathsf{im}}$ penalising large sampling weights for observations $\mathbf{y}$ which have already been recognised as inliers: $\ell_{\mathsf{im}}(\tilde{p}_{m,i}) = \max(0,\, \tilde{p}_{m,i} + s_{m,i} - 1)$, with $s_{m,i}$ being the inlier score as per Eq. 3 for observation $\mathbf{y}_i$ at instance sampling step m, and $\tilde{p}_{m,i}$ being its normalised sampling weight:

$$\tilde{p}_{m,i} = \frac{p(\mathbf{y}_i|\mathbf{s}_m; \mathbf{w})}{\max_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{y}|\mathbf{s}_m; \mathbf{w})}\,. \tag{11}$$

3.3. Post-Processing at Test Time

Expectation Maximisation In order refine the selected model parameters $\hat{\mathcal{M}}$, we implement a simple EM [16] algorithm. Given the posterior distribution:

$$p(\mathbf{h}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{h})p(\mathbf{h})}{p(\mathbf{y})}, \text{ with } p(\mathbf{y}) = \sum_{m=1}^{M} p(\mathbf{y}|\mathbf{h}_m)\,, \tag{12}$$

and likelihood $p(\mathbf{y}|\mathbf{h}) = \sigma^{-1}\phi(r(\mathbf{y}, \mathbf{h})\sigma^{-1})$ modelled by a normal distribution, we optimise model parameters $\mathcal{M}^*$ such that $\mathcal{M}^* = \arg\max_{\mathcal{M}} p(\mathcal{Y})$ with:

$$p(\mathcal{Y}) = \prod_{i=1}^{|\mathcal{Y}|} \sum_{m=1}^{M} p(\mathbf{y}_i|\mathbf{h}_m) p(\mathbf{h}_m)\,, \tag{13}$$

using fixed σ and $p(\mathbf{h}) = 1$ for all $\mathbf{h}$.

Instance Ranking In order to asses the significance of each selected model instance $\hat{\mathbf{h}}$, we compute a permutation $\boldsymbol{\pi}$ greedily sorting $\hat{\mathcal{M}}$ by joint inlier count, *i.e.*:

$$\pi_m = \arg\max_{q} \sum_{i=1}^{|\mathcal{Y}|} \max_{j \in \boldsymbol{\pi}_{1:m-1} \cup \{q\}} g_{\mathsf{i}}(\mathbf{y}_i, \hat{\mathbf{h}}_j)\,. \tag{14}$$

Such an ordering is useful in applications where the true number of instances present in the data may be ambiguous, and less significant instances may or may not be of interest. Small objects in a scene, for example, may elicit their own vanishing points, which may appear spurious for some applications, but could be of interest for others.

Instance Selection In some scenarios, the number of instances M needs to be determined as well but is not known beforehand, e.g. for uniquely assigning observations to model instances. For such cases, we consider the subset of instances $\hat{\mathcal{M}}_{1:q}$ up to the q-th model instance $\hat{\mathbf{h}}_q$ which increases the joint inlier count by at least Θ. Note that the inlier threshold θ for calculating the joint inlier count at this point may be chosen differently from the inlier threshold τ during hypothesis sampling. For example, in our experiments for homography estimation, we use a $\theta > \tau$ in order to strike a balance between under- and oversegmentation.

4. Multi-Model Fitting Datasets

Robust multi-model fitting algorithms can be applied to various tasks. While earlier works mostly focused on synthetic problems, such as fitting lines to point sets artificially perturbed by noise and outliers [54], real-world datasets for other tasks have been used since. The AdelaideRMF [63] dataset contains 38 image pairs with precomputed SIFT [34] feature point correspondences, which are clustered either via homographies (same plane) or fundamental matrices (same motion). Hopkins155 [56] consists of 155 image sequences with on average 30 frames each. Feature point correspondences are given as well, also clustered via their respective motions. For vanishing point estimation, the York Urban Dataset (YUD) [17] contains 102 images with three orthogonal ground truth vanishing directions each. All these datasets have in common that they are very limited in size, with no or just a small portion of the data reserved for training or validation. As a result, they are easily susceptible to parameter overfitting and ill-suited for contemporary machine learning approaches.

NYU Vanishing Point Dataset We therefore introduce the *NYU-VP* dataset. Based on the NYU Depth V2 [49] (NYU-D) dataset, it contains ground truth vanishing point labels for 1449 indoor scenes, *i.e.* it is more than ten times larger than the previously largest dataset in its category; see Tab. 1 for a comparison. To obtain each VP, we manually annotated at least two corresponding line segments. While most scenes show three VPs, it ranges between one and eight. In addition, we provide line segments extracted from the images with LSD [60], which we used in our experiments. Examples are shown in Fig. 3.

Figure 3: Examples from our newly presented NYU-VP dataset with two (left), three (middle) and five (right) vanishing points. **Top:** Original RGB image. **Middle:** Manually labelled line segments used to generate ground truth VPs. **Bottom:** Automatically extracted line segments.

task	dataset	train+val	test	instances
H	Adelaide [63]	0	19	1–6
F	Adelaide [63]	0	19	1–4
	Hopkins [56]	0	155	2–3
VP	YUD [17]	25	77	3
	YUD+ (ours)	25	77	3–8
	NYU-VP (ours)	1224	225	1–8

Table 1: Comparison of datasets for different applications of multi-model fitting: vanishing point (VP), homography (H) and fundamental matrix (F) fitting. We compare the numbers of combined training and validation scenes, test scenes, and model instances per scene.

YUD+ Each scene of the original York Urban Dataset (YUD) [17] is labelled with exactly three VPs corresponding to orthogonal directions consistent with the Manhattan-world assumption. Almost a third of all scenes, however, contain up to five additional significant yet unlabelled VPs. We labelled these VPs in order to allow for a better evaluation of VP estimators which do not restrict themselves to Manhattan-world scenes. This extended dataset, which we call *YUD+*, will be made available together with the automatically extracted line segments used in our experiments.

5. Experiments

For conditional sampling weight prediction, we implement a neural network based on the architecture of [12, 66]. We provide implementation and training details, as well as more detailed experimental results, in the appendix.

5.1. Line Fitting

We apply CONSAC to the task of fitting multiple lines to a set of noisy points with outliers. For training, we generated a synthetic dataset: each scene consists of randomly placed lines with points uniformly sampled along them and perturbed by Gaussian noise, and uniformly sampled outliers. After training CONSAC on this dataset in a supervised fashion, we applied it to the synthetic dataset of [54]. Fig. 4 shows how CONSAC sequentially focuses on different parts of the scene, depending on which model hypotheses have already been chosen, in order to increase the likelihood of sampling outlier-free non-redundant hypotheses. Notably, the network learns to focus on junctions rather than individual lines for selecting the first instances. The RANSAC-based single-instance hypothesis sampling makes sure that CONSAC still selects an individual line.

5.2. Vanishing Point Estimation

A vanishing point $\mathbf{v} \propto \mathbf{K}\mathbf{d}$ arises as the projection of a direction vector $\mathbf{d}$ in 3D onto an image plane using camera parameters $\mathbf{K}$. Parallel lines, *i.e.* with the same direction $\mathbf{d}$, hence converge in $\mathbf{v}$ after projection. If $\mathbf{v}$ is known, the corresponding direction $\mathbf{d}$ can be inferred via inversion: $\mathbf{d} \propto \mathbf{K}^{-1}\mathbf{v}$. VPs therefore provide information about the 3D structure of a scene from a single image. While two corresponding lines are sufficient to estimate a VP, real-world scenes generally contain multiple VP instances. We apply CONSAC to the task of VP detection and evaluate it on our new NYU-VP and YUD+ datasets, as well as on YUD [17]. We compare against several other robust estimators, and also against task-specific state-of-the art VP detectors. We train CONSAC on the training set of NYU-VP in a supervised fashion and evaluate on the test sets of NYU-VP, YUD+ and YUD using the same parameters. YUD and YUD+ were neither used for training nor parameter tuning. Notably, NYU-VP only depicts indoor scenes, while YUD also contains outdoor scenes.

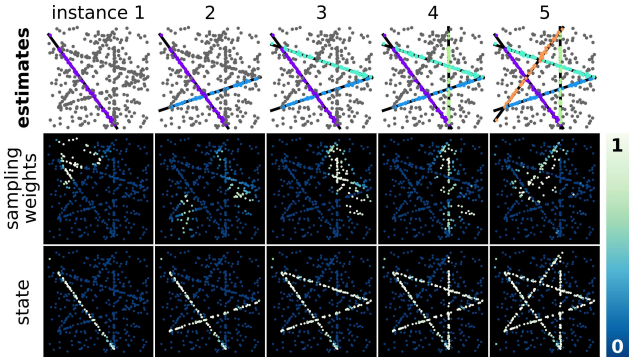

Figure 4: **Line fitting** result for the *star5* scene from [54]. We show the generation of the multi-hypothesis $\hat{\mathcal{M}}$ eventually selected by CONSAC. **Top:** Original points with estimated line instances at each instance selection step. **Middle:** Sampling weights at each instance step. **Bottom:** State $\mathbf{s}$ generated from the selected model instances.

5.2.1 Evaluation Protocol

We compute the error $e(\hat{\mathbf{h}}, \mathbf{h}^{\text{gt}})$ between two particular VP instances via the angle between their corresponding directions in 3D. Let $\mathbf{C}$ be the cost matrix with $C_{ij} = e(\hat{\mathbf{h}}_i, \mathbf{h}^{\text{gt}}_j)$. We can find a matching between ground truth $\mathcal{M}^{\text{gt}}$ and estimates $\hat{\mathcal{M}}$ by applying the Hungarian method on $\mathbf{C}$ and consider the errors of the matched VP pairs. For $N > M$ however, this would benefit methods with a tendency to oversegment, as a larger number of estimated VPs generally increases the likelihood of finding a good match to a ground truth VP. On the other hand, we argue that strictly penalising oversegmentation w.r.t. the ground truth is unreasonable, as smaller or more fine-grained structures which may have been missed during labelling may still be present in the data. We therefore assume that the methods also provide a permutation $\boldsymbol{\pi}$ (cf. Sec. 3.3) which ranks the estimated VPs by their significance, and evaluate using at most N most significant estimates. After matching, we generate the recall curve for all VPs of the test set and calculate the area under the curve (AUC) up to an error of $10°$. We report the average AUC and its standard deviation over five runs.

5.2.2 Robust Estimators

We compare against T-Linkage [35], MCT [38], Multi-X [6], RPA [36], RansaCov [37] and Sequential RANSAC [59]. We used our own implementation of T-Linkage and Sequential RANSAC, while adapting the code provided by the authors to VP detection for the other methods. All methods including CONSAC get the same line segments (geometric information only) as input, use the same residual metric and the same inlier threshold, and obtain the permutation $\boldsymbol{\pi}$ as described in Sec. 3.3. As Tab. 2 shows, CONSAC outperforms its competitors on all three datasets by a large margin. Although CONSAC was only trained on indoor scenes (NYU-VP) it also performs well on outdoor scenes (YUD/YUD+). Perhaps surprisingly, Sequential RANSAC also performs favourably, thus defying the commonly held notion that this greedy approach does not work well. Fig. 5 shows a qualitative result for CONSAC.

5.2.3 Task-Specific Methods

In addition to general-purpose robust estimators, we evaluate the state-of-the-art task-specific VP detectors of Zhai et al. [67], Kluger et al. [29] and Simon et al. [50]. Unlike the robust estimators, these methods may use additional information, such as the original RGB image, or enforce additional geometrical constraints. The method of Kluger et al. provides a score for each VP, which we used to generate the permutation $\boldsymbol{\pi}$. For Zhai et al. and Simon et al., we resorted to the more lenient naïve evaluation metric instead. Despite this, CONSAC performs superior to all task-specific methods on NYU-VP and YUD+, and slightly worse on YUD.

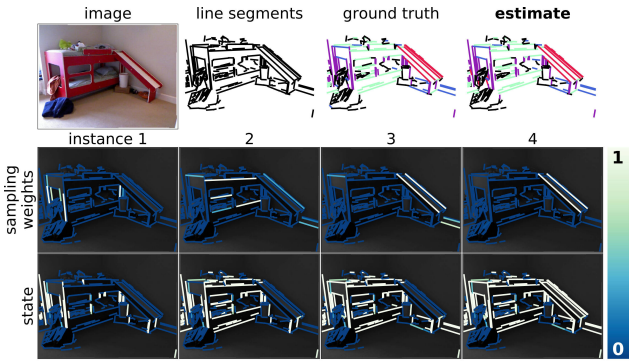

Figure 5: **VP fitting** result for a scene from the NYU-VP test set. **Top:** Original image, extracted line segments, assignment to ground truth VPs, and assignment to VPs predicted by CONSAC (average error: 2.2°). **Middle:** Sampling weights of line segments at each instance step. **Bottom:** State $\mathbf{s}$ generated from the selected model instances.

	NYU-VP		YUD+		YUD [17]	
	avg.	std.	avg.	std.	avg.	std.
robust estimators (on pre-extracted line segments)						
CONSAC	**65.0**	0.46	**77.1**	0.24	83.9	0.24
T-Linkage [35]	57.8	0.07	72.6	0.67	79.2	0.93
Seq. RANSAC	53.6	0.40	69.1	0.57	76.2	0.75
MCT [38]	47.0	0.67	62.7	1.28	67.7	0.59
Multi-X [6]	41.3	1.00	50.6	0.80	55.3	1.00
RPA [36]	39.4	0.65	48.5	1.14	52.5	1.35
RansaCov [37]	7.9	0.62	13.4	1.76	13.9	1.49
task-specific methods (full information)						
Zhai [67]†	63.0	0.25	72.1	0.50	84.2	0.69
Simon [50]†	62.1	0.67	73.6	0.77	85.1	0.74
Kluger [29]	61.7	—*	74.7	—*	**85.9**	—*

Table 2: **VP estimation:** Average AUC values (avg., in %, higher is better) and their standard deviations (std.) over five runs for vanishing point estimation on our new NYU-VP and YUD+ datasets as well as on YUD [17]. * Not applicable. † Naïve evaluation metric.

5.3. Two-view Plane Segmentation

Given feature point correspondences from two images showing different views of the same scene, we estimate multiple homographies $\mathbf{H}$ conforming to different 3D planes in the scene. As no sufficiently large labelled datasets exist for this task, we train our approach self-supervised (CONSAC-S) using SIFT feature correspondences extracted from the structure-from-motion scenes of [24, 51, 64] also used by [12]. Evaluation is performed on the AdelaideRMF [63] homography estimation dataset and adheres to the protocol used by [7], *i.e.* we re-

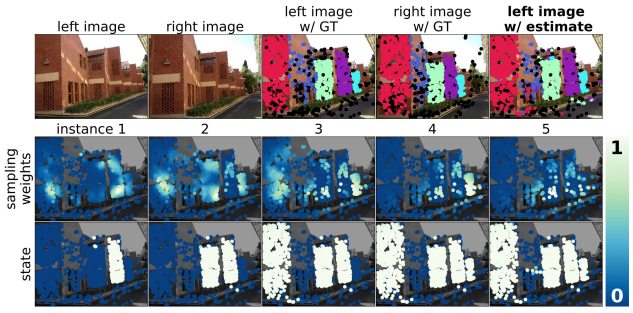

Figure 6: **Homography fitting** result for the AdelaideRMF *unihouse* scene. **Top:** Left and right image, feature points with ground truth labels, and feature points with labels predicted by CONSAC-S (ME: 8.4%). **Middle:** Sampling weights of feature points at each instance step. **Bottom:** State $\mathbf{s}$ generated from the selected model instances.

	AdelaideRMF-H [63]	
	avg.	std.
CONSAC-S	**5.21**	6.46
Progressive-X [7]*	6.86	**5.91**
Multi-X [6]*	8.71	8.13
Sequential RANSAC	11.14	10.54
PEARL [26]*	15.14	6.75
MCT [38]†	16.21	10.76
RPA [36]*	23.54	13.42
T-Linkage [35]*	54.79	22.17
RansaCov [37]*	66.88	18.44

Table 3: **Homography estimation:** Average misclassification errors (avg., in %, lower is better) and their standard deviations (std.) over five runs for homography fitting on the AdelaideRMF [63] dataset. * Results taken from [7]. † Results computed using code provided by the authors.

port the average misclassification error (ME) and its standard deviation over all scenes for five runs using identical parameters. We compare against the robust estimators Progressive-X [7], Multi-X [6], PEARL [26], MCT [38], RPA [36], T-Linkage [35], RansaCov [37] and Sequential RANSAC [59].

5.3.1 Results

As the authors of [38] used a different evaluation protocol, we recomputed results for MCT using the code provided by the authors. For Sequential RANSAC, we used our own implementation. Other results were carried over from [7] and are shown in Tab. 3. CONSAC-S outperforms state-of-the-art Progressive-X, yielding a significantly lower average ME with a marginally higher standard deviation. Notably, Sequential RANSAC performs favourably on this task as well. Fig. 6 shows a qualitative result for CONSAC-S.

	NYU-VP		Adelaide	
	avg.	std.	avg.	std.
with EM refinement				
CONSAC	65.01	0.46	—	—
CONSAC-S	63.44	0.40	5.21	6.46
without EM refinement				
CONSAC	62.90	0.52	—	—
CONSAC-S	61.83	0.58	6.17	7.79
CONSAC-S w/o IMR	59.94	0.47	8.14	11.79
CONSAC-S only IMR	29.31	0.37	21.12	13.45
CONSAC(-S) uncond.	48.36	0.29	9.17	11.50

Table 4: **Ablation study:** We compute mean AUC (NYU-VP), mean ME (AdelaideRMF [63]) and standard deviations for variations of CONSAC. See Sec. 5.4 for details.

5.4. Ablation Study

We perform ablation experiments in order to highlight the effectiveness of several methodological choices. As Tab. 4 shows, CONSAC with EM refinement consistently performs best on both vanishing point and homography estimation. If we disable EM refinement, accuracy drops measurably, yet remains on par with state-of-the-art (cf. Tab. 2 and Tab. 3). On NYU-VP we can observe that the self-supervised trained CONSAC-S achieves state-of-the-art performance, but is still surpassed by CONSAC trained in a supervised fashion. Training CONSAC-S without inlier masking regularisation (IMR, cf. Sec. 3.2.2) reduces accuracy measurably, while training only with IMR and disabling the self-supervised loss produces poor results. Switching to unconditional sampling for CONSAC (NYU-VP) or CONSAC-S (AdelaideRMF) comes with a significant drop in performance, and is akin to incorporating vanilla NG-RANSAC [12] into Sequential RANSAC.

6. Conclusion

We have presented CONSAC, the first learning-based robust estimator for detecting multiple parametric models in the presence of noise and outliers. A neural network learns to guide model hypothesis selection to different subsets of the data, finding model instances sequentially. We have applied CONSAC to vanishing point estimation, and multi-homography estimation, achieving state-of-the-art accuracy for both tasks. We contribute a new dataset for vanishing point estimation which facilitates supervised learning of multi-model estimators, other than CONSAC, in the future.

Acknowledgements This work was supported by the DFG grant *COVMAP* (RO 4804/2-1 and RO 2497/12-2) and has received funding from the European Research Council (ERC) under the European Union Horizon 2020 programme (grant No. 647769).

Appendix

This appendix contains additional implementation details (Sec. A) which may be helpful for reproducing our results. Sec. B provides additional details about the datasets presented and used in our paper. In Sec. C, we show additional details complementing our experiments shown in the paper.

A. Implementation Details

In Alg. 1, we present the CONSAC algorithm in another form, in addition to the description in Sec. 3 of the main paper, for ease of understanding. A list of all user definable parameters and the settings we used in our experiments is given in Tab. 5.

Algorithm 1 CONSAC

Input: $\mathcal{Y}$ – set of observations, $\mathbf{w}$ – network parameters
Output: $\hat{\mathcal{M}}$ – multi-hypothesis
$\mathcal{P} \leftarrow \varnothing$
for $i \leftarrow 1$ **to** P **do**
  $\mathcal{M} \leftarrow \varnothing$
  $\mathbf{s} \leftarrow \mathbf{0}$
  for $m \leftarrow 1$ **to** M **do**
    $\mathcal{H} \leftarrow \varnothing$
    for $s \leftarrow 1$ **to** S **do**
      Sample a minimal set of observations $\{\mathbf{y}_1, \ldots, \mathbf{y}_C\}$ with $\mathbf{y} \sim p(\mathbf{y}|\mathbf{s}; \mathbf{w})$.
      $\mathbf{h} \leftarrow f_{\mathsf{s}}(\{\mathbf{y}_1, \ldots, \mathbf{y}_C\})$
      $\mathcal{H} \leftarrow \mathcal{H} \cup \{\mathbf{h}\}$
    end
    $\hat{\mathbf{h}} \leftarrow \arg\max_{\mathbf{h} \in \mathcal{H}} g_{\mathsf{s}}(\mathbf{h}, \mathcal{Y}, \mathcal{M})$
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{\hat{\mathbf{h}}\}$
    $\mathbf{s} \leftarrow \max_{\hat{\mathbf{h}} \in \mathcal{M}} g_{\mathsf{y}}(\mathcal{Y}, \hat{\mathbf{h}})$
  end
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{\mathcal{M}\}$
end
$\hat{\mathcal{M}} \leftarrow \arg\max_{\mathcal{M} \in \mathcal{P}} g_{\mathsf{m}}(\mathcal{M}, \mathcal{Y})$

A.1. Neural Network

We use a neural network similar to PointNet [44] and based on [12, 66] for prediction of conditional sampling weights in CONSAC. Fig. 7 gives an overview of the architecture. Observations $\mathbf{y} \in \mathcal{Y}$, e.g. line segments or feature point correspondences, are stacked into a tensor of size $D \times |\mathcal{Y}| \times 1$. Note that the size of the tensor depends on the number of observations per scene. The dimensionality D of each observation $\mathbf{y}$ is application specific. The current state $\mathbf{s}$ contains a scalar value for each observation and is hence a tensor of size $1 \times |\mathcal{Y}| \times 1$. The input of the network is a concatenation of observations $\mathcal{Y}$ and state $\mathbf{s}$, *i.e.* a tensor of size $(D+1) \times |\mathcal{Y}| \times 1$. After a single convolutional layer (1×1, 128 channels) with ReLU [22] activation function, we apply six residual blocks [23]. Each residual block is composed of two series of convolutions (1×1, 128 channels), instance normalisation [57], batch normalisation [25] (optional) and ReLU activation. After another convolutional layer (1×1, 1 channel) with sigmoid activation, we normalise the outputs so that the sum of sampling weights equals one. Only using 1×1 convolutions, this network architecture is order invariant w.r.t. observations $\mathcal{Y}$. We implement the architecture using PyTorch [41] version 1.2.0.

			VP estimation	homography estimation
training	learning rate		10^{-4}	$2 \cdot 10^{-6}$
	batch size	B	16	1
	batch normalisation		yes	no
	epochs		400	100
	inlier threshold	τ	10^{-3}	10^{-4}
	IMR weight	κ	10^{-2}	10^{-2}
	observations per scene	$\lvert\mathcal{Y}\rvert$	256	256
	number of instances	M	3	6
	single-instance samples	S	2	2
	multi-instance samples	P	2	2
	sample count	K	4	8
test	inlier threshold	τ	10^{-3}	10^{-4}
	inlier thresh. (selection)	θ	—	$3 \cdot 10^{-3}$
	inlier cutoff (selection)	Θ	—	6
	observations per scene	$\lvert\mathcal{Y}\rvert$	variable	
	number of instances	M	6	6
	single-instance samples	S	32	100
	multi-instance samples	P	32	100
	EM iterations		10	10
	EM standard deviation	σ	10^{-8}	10^{-9}

Table 5: **User definable parameters** of CONSAC and the values we chose for our experiments on vanishing point estimation and homography estimation. We distinguish between values used during training and at test time. Mathematical symbols refer to the notation used either in the main paper or in this supplementary document.

A.1.1 Training Procedure

We train the neural network using the Adam [28] optimiser and utilise a cosine annealing learning rate schedule [33]. We clamp losses to a maximum absolute value of 0.3 in order to avoid divergence caused by large gradients resulting from large losses induced by poor hypothesis samples.

Number of Observations In order to keep the number of observations $|\mathcal{Y}|$ constant throughout a batch, we sample a fixed number of observations from all observations of

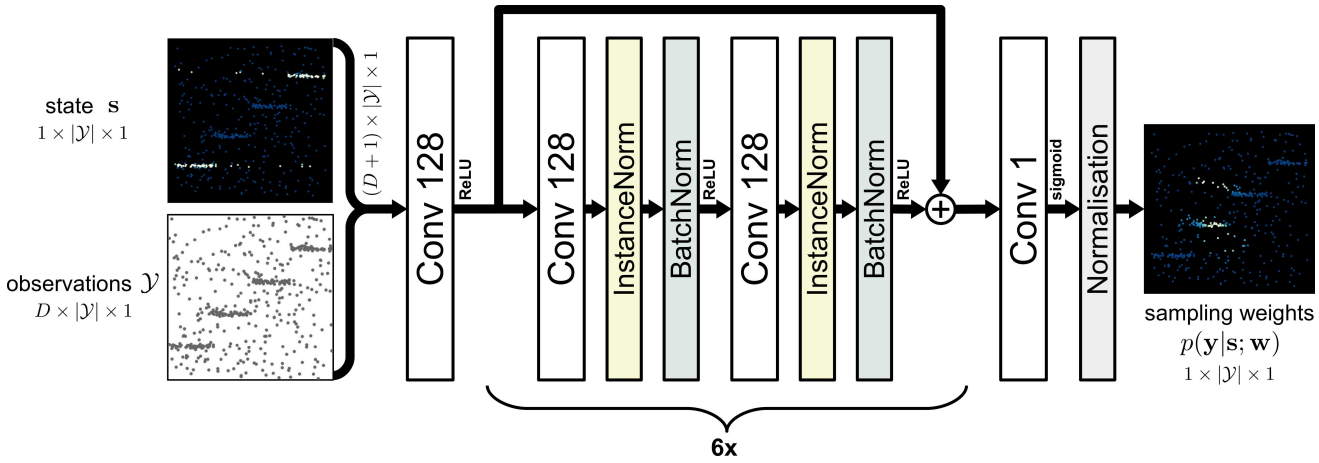

Figure 7: **CONSAC neural network architecture** used for all experiments. We stack observations $\mathcal{Y}$, e.g. line segments or point correspondences (*not* an image), and state $\mathbf{s}$ into a tensor of size $(D+1) \times |\mathcal{Y}| \times 1$, and feed it into the network. The network is composed of linear 1×1 convolutional layers interleaved with instance normalisation [57], batch normalisation [25] and ReLU [22] layers which are arranged as residual blocks [23]. Only using 1×1 convolutions, the network is order invariant w.r.t. observations $\mathcal{Y}$. The architecture is based on [12, 66].

a scene during training. At test time, all observations are used.

Pseudo Batches During training, we sample P multi-hypotheses $\mathcal{M}$, from which we select the best multi-hypothesis $\hat{\mathcal{M}}$ for each set of input observations $\mathcal{Y}$ within a batch of size B. To approximate the expectation of our training loss (see Sec. 3.2 of the main paper), we repeat this process K times, to generate K samples of *selected* multi-hypotheses $\hat{\mathcal{M}}$ for each $\mathcal{Y}$. We generate each multi-hypothesis $\mathcal{M}$ by *sequentially* sampling S single-instance hypotheses $\mathbf{h}$ and selecting the best one, conditioned on a state $\mathbf{s}$. The state $\mathbf{s}$ varies between these innermost sampling loops, since we compute $\mathbf{s}$ based on all previously selected single instance hypotheses $\hat{\mathbf{h}}$ of a multi-hypothesis $\mathcal{M}$. Because $\mathbf{s}$ is always fed into the network alongside observations $\mathcal{Y}$, we have to run $P \cdot K$ forward passes for each batch. We can, however, parallelise these passes by collating observations and states into a tensor of size $P \times K \times B \times (D+1) \times |\mathcal{Y}|$. We reshape this tensor so that it has size $B^* \times (D+1) \times |\mathcal{Y}|$ with an effective pseudo batch size $B^* = P \cdot K \cdot B$, in order to process all samples in parallel while using the same neural network weights for each pass within B^*. This means that sample sizes P and K are subject to both time and hardware memory constraints. We observe, however, that small sample sizes during training are sufficient in order to achieve good results using higher sample sizes at test time.

Inlier Masking Regularisation For self-supervised training, we multiply the inlier masking regularisation (IMR) term ℓ_{im} (cf. Sec. 3.2.2 in the main paper) with a factor κ in order to regulate its influence compared to the regular self-supervision loss ℓ_{self}, *i.e.*:

$$\ell = \ell_{\text{self}} + \kappa \cdot \ell_{\text{im}} \tag{15}$$

A.2. Scoring Functions

In order to gauge whether an observation $\mathbf{y}$ is an inlier of model instance $\mathbf{h}$, we utilise a soft inlier function adapted from [11]:

$$g_{\text{i}}(\mathbf{y}, \mathbf{h}) = 1 - \sigma(\beta r(\mathbf{y}, \mathbf{h}) - \beta\tau)\,, \tag{16}$$

with inlier threshold τ, softness parameter $\beta = 5\tau^{-1}$, a task-specific residual function $r(\mathbf{y}, \mathbf{h})$ (see Sec. A.3 for details), and using the sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}\,. \tag{17}$$

The multi-instance scoring function g_{m}, which we use to select the best muti-hypothesis, *i.e.* hypothesis of multiple model instances $\hat{\mathcal{M}} = \{\hat{\mathbf{h}}_1, \ldots, \hat{\mathbf{h}}_M\}$, from a pool of multi-instance hypotheses $\mathcal{P} = \{\mathcal{M}_1, \ldots, \mathcal{M}_P\}$, counts the joint inliers of all models in a multi-instance:

$$g_{\text{m}}(\mathcal{M}, \mathcal{Y}) = \sum_{\mathbf{y} \in \mathcal{Y}} \max_{\mathbf{h} \in \mathcal{M}} g_{\text{i}}(\mathbf{y}, \mathbf{h})\,. \tag{18}$$

The single instance scoring function g_{s}, which we use for selection of single model instances $\mathbf{h}$ given the set of previously selected model instances $\mathcal{M}$, is a special case of the multi-instance scoring function g_{m}:

$$g_{\text{s}}(\mathbf{h}, \mathcal{Y}, \mathcal{M}) = g_{\text{m}}(\mathcal{M} \cup \{\mathbf{h}\}, \mathcal{Y})\,. \tag{19}$$

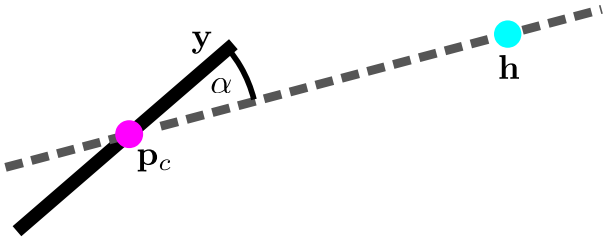

Figure 8: Visualisation of the angle α used for the vanishing point estimation residual function $r(\mathbf{y}, \mathbf{h})$.

A.3. Residual Functions

Line Fitting For the line fitting problem, each observation is a 2D point in homogeneous coordinates $\mathbf{y} = (x\, y\, 1)^\mathsf{T}$, and each model is a line in homogeneous coordinates $\mathbf{h} = \frac{1}{\|(n_1\, n_2)\|}\,(n_1\, n_2\, d)^\mathsf{T}$. We use the absolute point-to-line distance as the residual:

$$r(\mathbf{y}, \mathbf{h}) = |\mathbf{y}^\mathsf{T}\mathbf{h}|\,. \tag{20}$$

Vanishing Point Estimation Observations $\mathbf{y}$ are given by line segments with start point $\mathbf{p}_1 = (x_1\, y_1\, 1)^\mathsf{T}$ and end point $\mathbf{p}_2 = (x_2\, y_2\, 1)^\mathsf{T}$, and models are vanishing points $\mathbf{h} = (x\, y\, 1)^\mathsf{T}$. For each line segment $\mathbf{y}$, we compute the corresponding line $\mathbf{l}_\mathbf{y} = \mathbf{p}_1 \times \mathbf{p}_2$ and the centre point $\mathbf{p}_c = \frac{1}{2}(\mathbf{p}_1 + \mathbf{p}_2)$. As visualised by Fig. 8, we define the residual via the cosine of the angle α between $\mathbf{l}_\mathbf{y}$ and the constrained line $\mathbf{l}_c = \mathbf{h} \times \mathbf{p}_c$, *i.e.* the line connecting the vanishing point with the centre of the line segment:

$$r(\mathbf{y}, \mathbf{h}) = 1 - \cos\alpha = 1 - \frac{|\mathbf{l}_{\mathbf{y},1:2}^\mathsf{T}\mathbf{l}_{c,1:2}|}{\|\mathbf{l}_{\mathbf{y},1:2}\|\|\mathbf{l}_{c,1:2}\|}\,. \tag{21}$$

Homography Estimation Observations $\mathbf{y}$ are given by point correspondences $\mathbf{p}_1 = (x_1\, y_1\, 1)^\mathsf{T}$ and $\mathbf{p}_2 = (x_2\, y_2\, 1)^\mathsf{T}$, and models are plane homographies $\mathbf{h} = \mathbf{H}^{3\times3}$ which shall map $\mathbf{p}_1$ to $\mathbf{p}_2$. We compute the symmetric squared transfer error:

$$r(\mathbf{y}, \mathbf{h}) = \|\mathbf{p}_1 - \mathbf{p}_1'\|^2 + \|\mathbf{p}_2 - \mathbf{p}_2'\|^2\,, \tag{22}$$

with $\mathbf{p}_2' \propto \mathbf{H}\mathbf{p}_1$ and $\mathbf{p}_1' \propto \mathbf{H}^{-1}\mathbf{p}_2$.

B. Dataset Details and Analyses

B.1. Line Fitting

For training CONSAC on the line fitting problem, we generated a synthetic dataset of 10000 scenes. Each scene consists of four lines placed at random within a $\{0, 1\} \times \{0, 1\}$ square. For each line, we randomly define a line segment with a length of $30 - 100\%$ of the maximum length of the line within the square. Then, we randomly sample $40 - 100$ points along the line segment and perturb them by Gaussian noise $\mathcal{N} \sim (0, \sigma^2)$, with $\sigma \in (0.007, 0.008)$ sampled uniformly. Finally, we add $40 - 60\%$ outliers via random uniform sampling. Fig. 9 shows a few examples from this dataset.

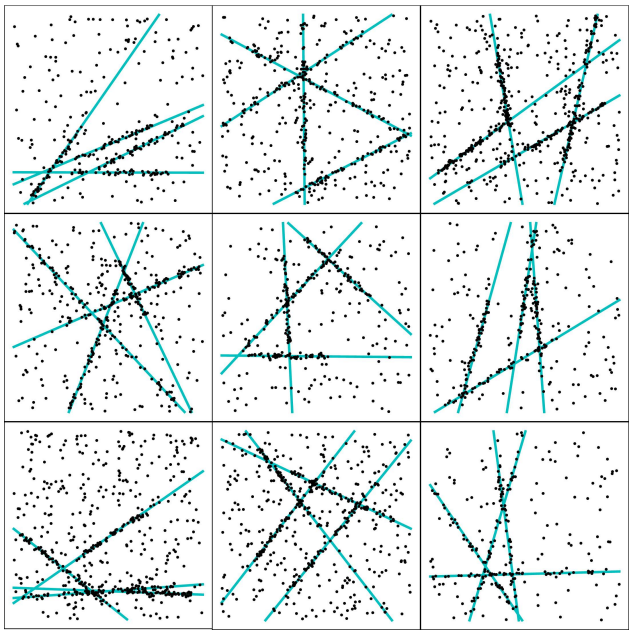

Figure 9: **Line fitting:** we show examples from the synthetic dataset we used to train CONSAC on the line fitting problem. Each scene consists of four lines placed at random, with points sampled along them, perturbed by Gaussian noise and outliers. Cyan = ground truth lines.

Figure 10: **Line fitting:** we use the synthetic *stair4* (left), *star5* (middle) and *star11* (right) scenes from [54], which were also used by [7], in our experiments.

For evaluation, we use the synthetic *stair4*, *star5* and *star11* scenes from [54], which were also used by [7]. As Fig. 10 shows, each scene consists of 2D points forming four, five or eleven line segments. The points are perturbed by Gaussian noise ($\sigma = 0.0075$) and contain $50 - 60\%$ outliers.

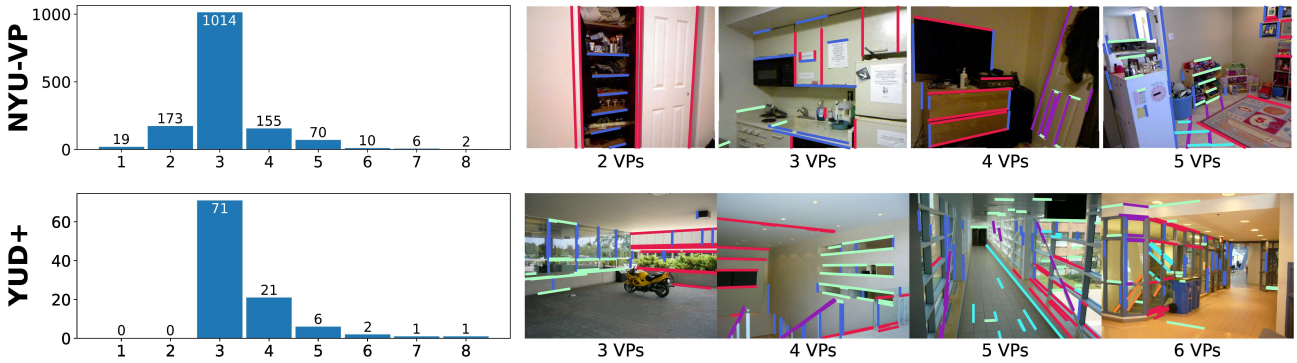

Figure 11: **Vanishing points per scene:** Histograms showing the numbers of vanishing point instances per image for our new NYU-VP dataset (top) and our YUD+ dataset extension (bottom), in addition to a few example images. We illustrate the vanishing points present in each example via colour-coded line segments.

B.2. Vanishing Point Estimation

NYU-VP In Fig. 11 (top), we show a histogram of the number of vanishing points per image in our new NYU-VP dataset. In addition, we show a few example images for different numbers of vanishing points. NYU-VP solely consists of indoor scenes.

YUD+ In Fig. 11 (bottom), we show a histogram of the number of vanishing points per image in our new YUD+ dataset extension. By comparison, the original YUD [17] contains exactly three vanishing point labels for each of the 102 scenes. YUD contains both indoor and outdoor scenes.

B.3. Homography Estimation

For self-supervised training for the task of homography estimation, we use SIFT [34] feature correspondences extracted from the structure-from-motion scenes of [24, 51, 64]. Specifically, we used the outdoor scenes *Buckingham*, *Notredame*, *Sacre Coeur*, *St. Peter's* and *Reichstag* from [24], *Fountain* and *Herzjesu* from [51], and 16 indoor scenes from SUN3D [64]. We use the SIFT correspondences computed and provided by Brachmann and Rother [12], and discard suspected gross outliers with a matching score ratio greater than 0.9. As this dataset is imbalanced in the sense that some scenes contain significantly more image pairs than others – for *St. Peter's* we have 9999 image pairs, but for *Reichstag* we only have 56 – we apply a rebalancing sampling during training: instead of sampling image pairs uniformly at random, we uniformly sample one of the scenes first, and then we sample an image pair from within this scene. This way, each scene is sampled during training at the same rate. During training, we augment the data by randomly flipping all points horizontally or vertically, and shifting and scaling them along both axes independently by up to $\pm 10\%$ of the image width or height.

C. Additional Experimental Results

C.1. Line Fitting

Sampling Efficiency In order to analyse the efficiency of the conditional sampling of CONSAC compared to a Sequential RANSAC, we computed the $F1$ score w.r.t. estimated model instances on the *stair4*, *star5* and *star11* line fitting scenes from [54] for various combinations of single-instance samples S and multi-instance samples P. As Fig. 12 shows, CONSAC achieves higher $F1$ scores with fewer hypotheses on *stair4* and *star5*. As we trained CONSAC on data containing only four line segments, while *star5* depicts five lines, this demonstrates that CONSAC is able to generalise beyond the number of model instances it has been trained for. On *star11*, which contains eleven lines, it does not perform as well, suggesting that this generalisation may not extend arbitrarily beyond numbers of instances CONSAC has been trained on. In practice, however, our real-world experiments on homography estimation and vanishing point estimation show that it is sufficient to simply train CONSAC on a reasonably large number of instances in order to achieve very good results.

Sampling Weights Throughout Training We looked at the development of sampling weights as neural network training progresses, using *star5* as an example. As Fig. 13 shows, sampling weights are randomly – but not uniformly – distributed throughout all instance sampling steps before training has begun. At 1000 iterations, we observe that the neural network starts to focus on different regions of the data throughout the instance sampling steps. From thereon, this focus gets smaller and more accurate as training progresses. After 100000 iterations, the network has learned to focus on points mostly belonging to just one or two true line segments.

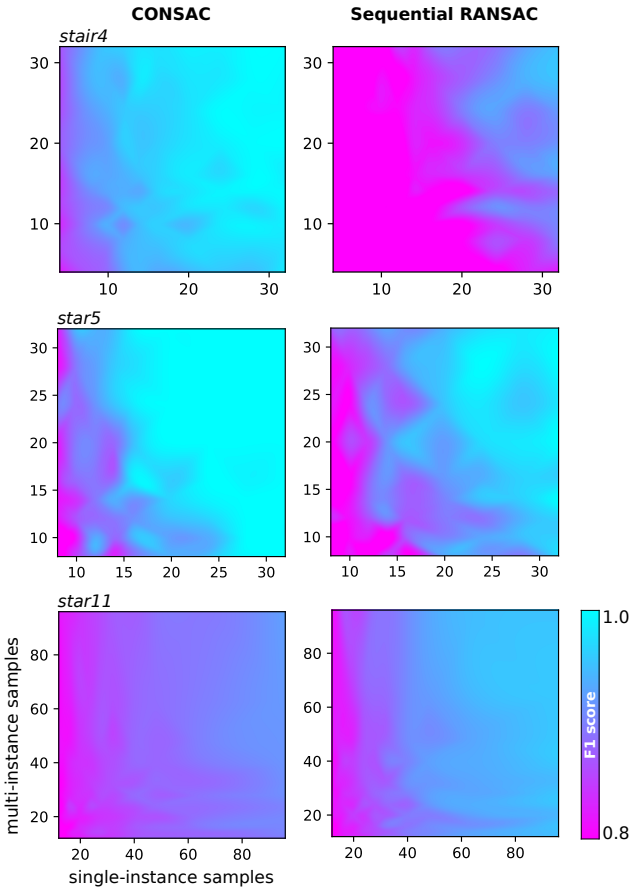

Figure 12: **Line fitting:** Using the *stair4* (top), *star5* (middle) and *star11* (bottom) line fitting scenes from [54], we compute the $F1$ scores for various combinations of single-instance samples S (abscissa) and multi-instance samples P (ordinate) and plot them as a heat map. We compare CONSAC (left) with Sequential RANSAC (right). Magenta = low, cyan = high $F1$ score.

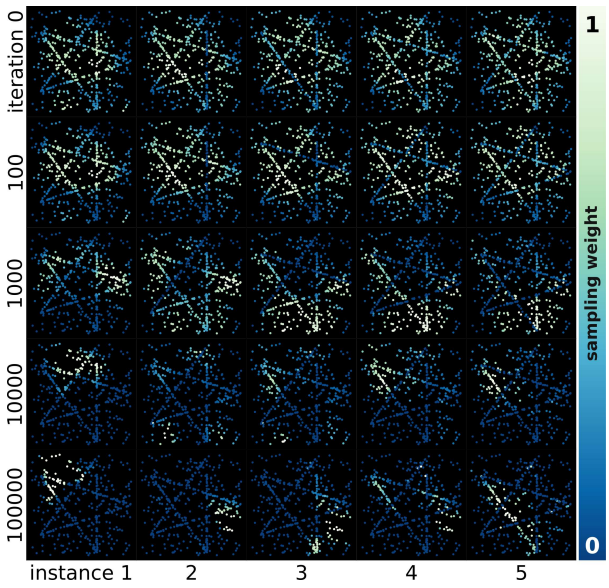

Figure 13: **Line fitting:** We show how the sampling weights at each instance sampling step develop as neural network training progresses, using the *star5* line fitting scene from [54] as an example. Each row depicts the sampling weights used to sample the eventually selected best multi-hypothesis $\hat{\mathcal{M}}$. **Top to bottom:** training iterations $0 - 100000$. **Left to right:** model instance sampling steps $1 - 5$. Sampling weights: Blue = low, white = high.

C.2. Vanishing Point Estimation

Evaluation Metric We denote ground truth VPs of an image by $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_M\}$ and estimates by $\hat{\mathcal{V}} = \{\hat{\mathbf{v}}_1, \ldots, \hat{\mathbf{v}}_N\}$. We compute the error between two particular VP instances via the angle $e(\mathbf{v}, \hat{\mathbf{v}})$ between their corresponding directions in 3D using camera intrinsics $\mathbf{K}$:

$$e(\mathbf{v}, \hat{\mathbf{v}}) = \arccos \frac{\left| \left(\mathbf{K}^{-1}\mathbf{v}\right)^{\mathsf{T}} \mathbf{K}^{-1}\hat{\mathbf{v}} \right|}{||\mathbf{K}^{-1}\mathbf{v}|| \cdot ||\mathbf{K}^{-1}\hat{\mathbf{v}}||} . \tag{23}$$

We use this error to define the cost matrix $\mathbf{C}$: $C_{ij} = e(\mathbf{v}_i, \hat{\mathbf{v}}_j)$ in Sec. 5.2.1 of the main paper.

Results For vanishing point estimation, we provide recall curves for errors up to $10°$ in Fig. 14 for our new NYU-VP dataset, for our YUD+ dataset extension, as well as the original YUD [17]. We compare CONSAC with the robust multi-model fitting approaches T-Linkage [35], Sequential RANSAC [59], Multi-X [6], RPA [36] and RansaCov [37], as well as the task-specific vanishing point estimators of Zhai et al. [67], Simon et al. [50] and Kluger et al. [29]. We selected the result with the median area under the curve (AUC) of five runs for each method. CONSAC does not find more vanishing points within the $10°$ range than state-of-the-art vanishing point estimators, indicated by similar recall values at $10°$. However, it does estimate vanishing points more accurately on NYU-VP and YUD+, as the high recall values for low errors ($< 4°$) show. On YUD [17], CONSAC achieves similar or slightly worse recall. Compared to other robust estimators, however, CONSAC performs better than all methods on all datasets across the whole error range. In Fig. 16, we show additional qualitative results from the NYU-VP dataset, and in Fig. 17, we show additional qualitative results from the YUD+ dataset.

C.3. Homography Estimation

We provide results computed on AdelaideRMF [63] for all scenes seperately. In Fig. 15, we compare CONSAC-S – *i.e.* CONSAC trained in a self-supervised manner – to Progressive-X [7], Multi-X [6], PEARL [26], RPA [36], RansaCov [37] and T-Linkage [35]. We adapted the graph

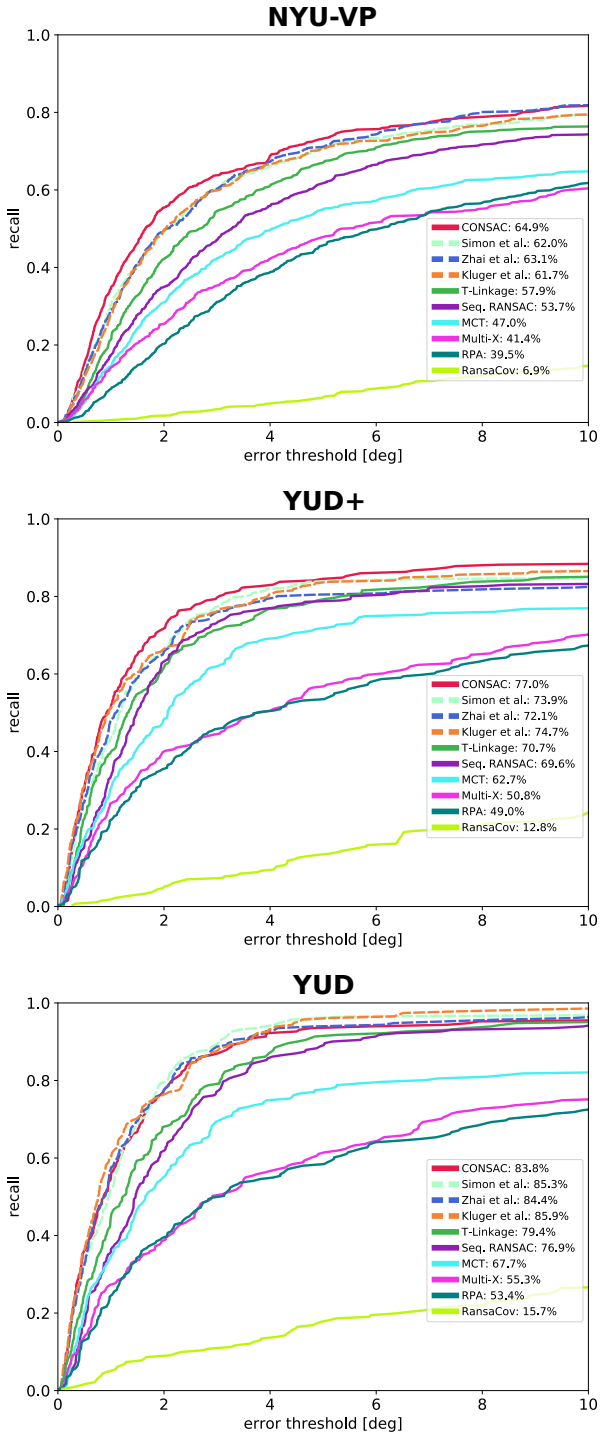

Figure 14: **Vanishing point estimation:** Recall curves for errors up to 10° for all methods which we considered in our experiments. We selected the result with the *median* AUC out of five runs for each method. Robust estimators are represented with solid lines, task-specific VP estimators with dashed lines. **Top:** Results on our new NYU-VP dataset. **Middle:** Results on our new YUD+ dataset extension. **Bottom:** Results on the original YUD [17].

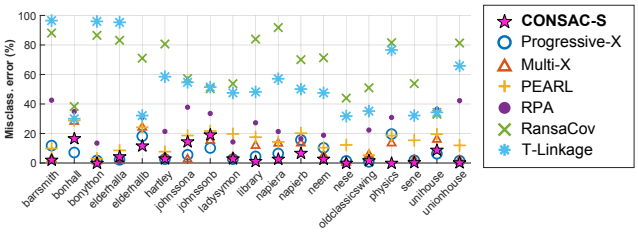

Figure 15: **Homography estimation:** Misclassification errors (in %, average over five runs) for all homography estimation scenes of AdelaideRMF [63]. Graph adapted from [7].

	no. of planes	CONSAC-S	MCT [38]	Sequential RANSAC
barrsmith	2	**2.07**	11.29	12.95
bonhall	6	**16.63**	29.29	20.43
bonython	1	**0.00**	2.42	**0.00**
elderhalla	2	**4.39**	21.41	16.36
elderhallb	3	**11.69**	20.31	18.67
hartley	2	**2.94**	15.19	9.38
johnsona	4	**14.48**	18.77	28.04
johnsonb	6	**19.17**	33.87	27.46
ladysymon	2	**2.95**	16.46	3.80
library	2	**1.21**	14.79	11.35
napiera	2	**2.72**	21.32	11.66
napierb	3	**6.72**	16.83	21.24
neem	3	**2.74**	14.36	14.44
nese	2	**0.00**	12.83	0.47
oldclass.	2	1.69	15.20	**1.32**
physics	1	**0.00**	3.21	**0.00**
sene	2	**0.40**	4.80	2.00
unihouse	5	**8.84**	34.10	10.69
unionhouse	1	**0.30**	1.51	1.51
average		**5.21**	16.21	11.14

Table 6: **Homography estimation:** Misclassification errors (in %, average over five runs) for all homography estimation scenes of AdelaideRMF [63].

directly from [7]. CONSAC-S achieves state-of-the-art performance on 13 of 19 scenes. Tab. 6 compares CONSAC-S with MCT [38] and Sequential RANSAC. We computed results for MCT using code provided by the authors, and used our own implementation for Sequential RANSAC, since no results obtained using the same evaluation protocol (average over five runs) were available in previous works. In Fig. 18, we show additional qualitative results from the AdelaideRMF [63] dataset.

Figure 16: Three qualitative examples for VP estimation with CONSAC on our NYU-VP dataset. For each example we show the original image, extracted line segments, line assignments to ground truth VPs, and to final estimates in the first row. In the second and third row, we visualise the generation of the multi-hypothesis $\hat{\mathcal{M}}$ eventually selected by CONSAC. The second row shows the sampling weights per line segment which were used to generate each hypothesis $\hat{\mathbf{h}} \in \hat{\mathcal{M}}$. The third row shows the resulting state $\mathbf{s}$. (Blue = low, white = high.) Between rows two and three, we indicate the individual VP errors. The checkerboard pattern and "—" entries indicate instances for which no ground truth is available. The last example is a failure case, where only two out of four VPs were correctly estimated.

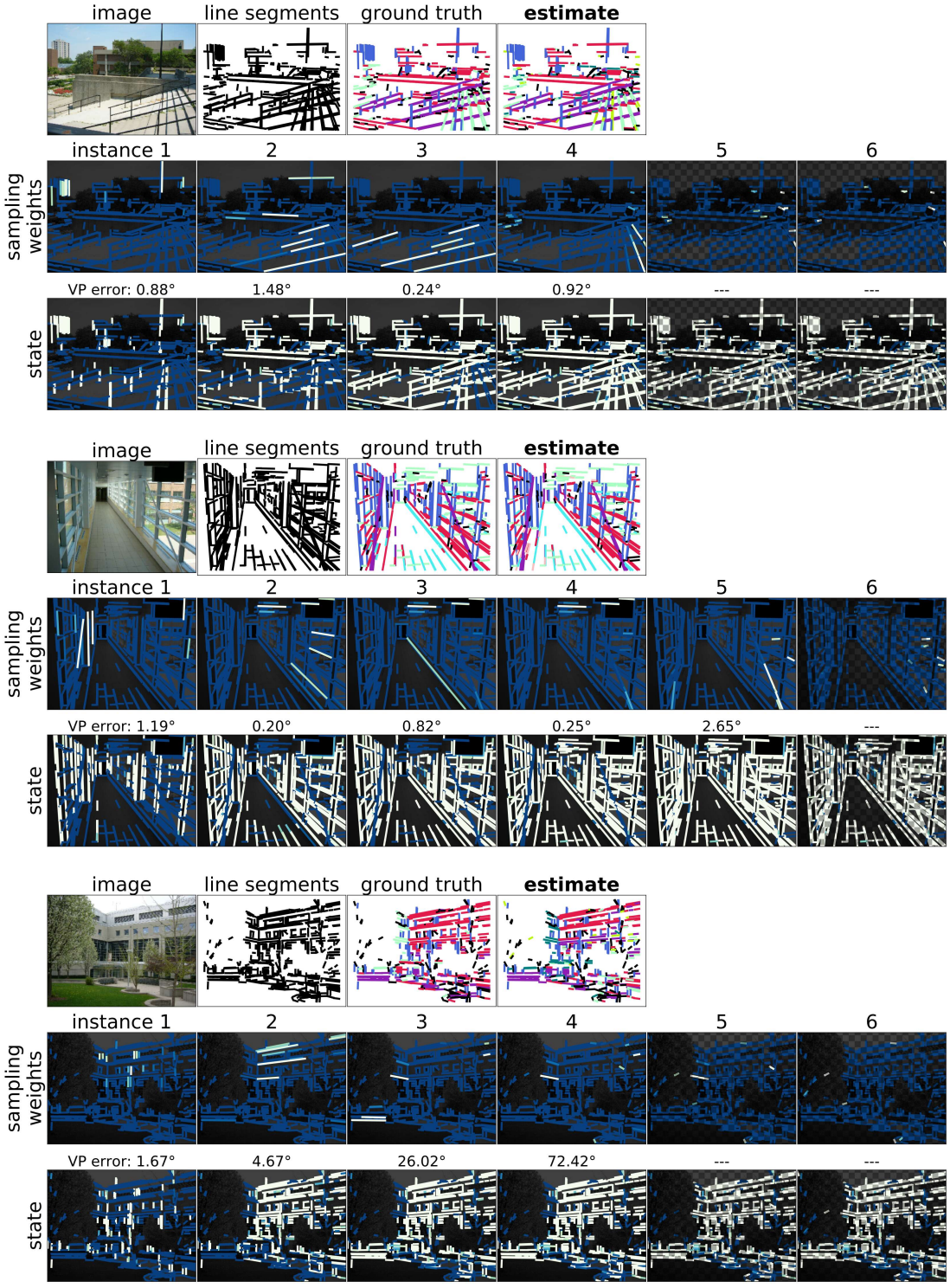

Figure 17: Three qualitative examples for VP estimation with CONSAC on the YUD+ dataset. For each example we show the original image, extracted line segments, line assignments to ground truth VPs, and to final estimates in the first row. In the second and third row, we visualise the generation of the multi-hypothesis $\hat{\mathcal{M}}$ eventually selected by CONSAC. The second row shows the sampling weights per line segment which were used to generate each hypothesis $\hat{\mathbf{h}} \in \hat{\mathcal{M}}$. The third row shows the resulting state $\mathbf{s}$. (Blue = low, white = high.) Between rows two and three, we indicate the individual VP errors. The checkerboard pattern and "—" entries indicate instances for which no ground truth is available. The last example is a failure case, where only two out of four VPs were correctly estimated.

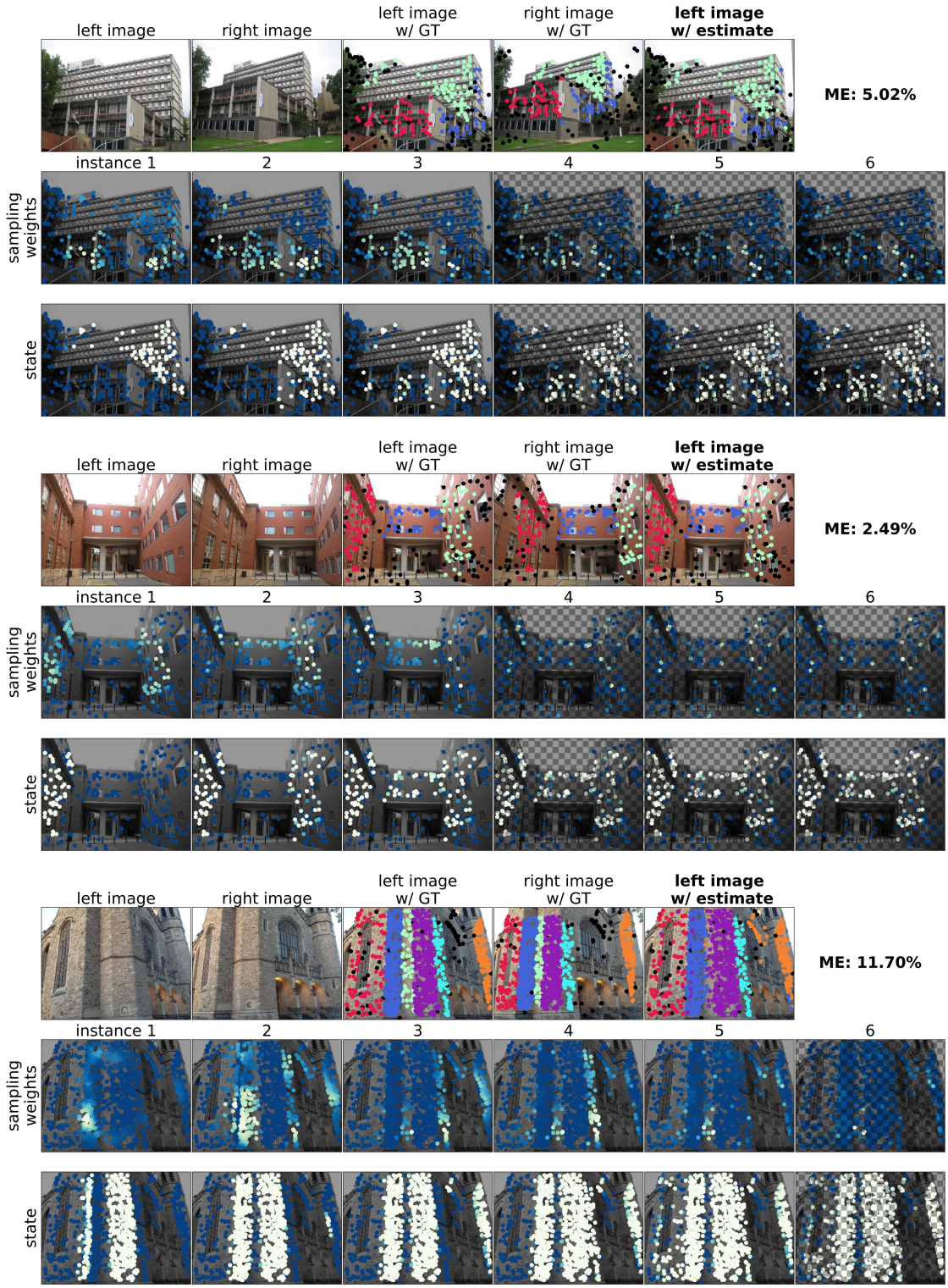

Figure 18: Three qualitative examples for homography estimation with CONSAC-S on the AdelaideRMF [63] dataset. For each example we show the original images, points with ground truth labels, final estimates, and the misclassification error (ME) in the first row. In the second and third row, we visualise the generation of the multi-hypothesis $\hat{\mathcal{M}}$ eventually selected by CONSAC. The second row shows the sampling weights per point correspondence which were used to generate each hypothesis $\hat{\mathbf{h}} \in \hat{\mathcal{M}}$. The third row shows the resulting state $\mathbf{s}$. (Blue = low, white = high.) The checkerboard pattern indicates instances which were discarded by CONSAC in the final instance selection step.

References

[1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M. Seitz, and Richard Szeliski. Building rome in a day. *Commun. ACM*, 2011. 1

[2] Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *IJCV*, 2018. 1

[3] Paul Amayo, Pedro Piniés, Lina M Paz, and Paul Newman. Geometric multi-model fitting with a convex relaxation algorithm. In *CVPR*, 2018. 2

[4] Michel Antunes and Joao P Barreto. A global approach for the detection of vanishing points and mutually orthogonal vanishing directions. In *CVPR*, 2013. 3

[5] Daniel Barath and Jiří Matas. Graph-cut RANSAC. In *CVPR*, 2018. 3

[6] Daniel Barath and Jiri Matas. Multi-class model fitting by energy minimization and mode-seeking. In *ECCV*, 2018. 2, 7, 8, 13

[7] Daniel Barath and Jiri Matas. Progressive-X: Efficient, anytime, multi-model fitting algorithm. *ICCV*, 2019. 2, 3, 7, 8, 11, 13, 14

[8] Daniel Barath, Jiri Matas, and Levente Hajder. Multi-H: Efficient recovery of tangent planes in stereo images. In *BMVC*, 2016. 2

[9] Olga Barinova, Victor Lempitsky, Elena Tretiak, and Pushmeet Kohli. Geometric image parsing in man-made environments. In *ECCV*, 2010. 3

[10] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Uncertainty-driven 6D pose estimation of objects and scenes from a single RGB image. In *CVPR*, 2016. 1

[11] Eric Brachmann and Carsten Rother. Learning less is more-6D camera localization via 3D surface regression. In *CVPR*, 2018. 1, 10

[12] Eric Brachmann and Carsten Rother. Neural-guided RANSAC: Learning where to sample model hypotheses. In *ICCV*, 2019. 2, 3, 4, 6, 7, 8, 9, 10, 12

[13] Tat-Jun Chin, Hanzi Wang, and David Suter. Robust fitting of multiple structures: The statistical learning approach. In *ICCV*, 2009. 2, 3

[14] Tat-Jun Chin, Jin Yu, and David Suter. Accelerated hypothesis generation for multistructure data via preference analysis. *TPAMI*, 2011. 3

[15] Ondrej Chum and Jiri Matas. Matching with PROSAC-progressive sample consensus. In *CVPR*, 2005. 3

[16] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 1977. 3, 5

[17] Patrick Denis, James H Elder, and Francisco J Estrada. Efficient edge-based methods for estimating manhattan frames in urban imagery. In *ECCV*, 2008. 2, 5, 6, 7, 12, 13, 14

[18] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Deep image homography estimation. In *RSS Workshops*, 2016. 2

[19] 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, 2, 3

[20] Adriano Garcia, Edward Mattison, and Kanad Ghose. High-speed vision-based autonomous indoor navigation of a quadcopter. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 338–347, 2015. 1

[21] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. 1

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *ICCV*, 2015. 9, 10

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 9, 10

[24] Jared Heinly, Johannes Lutz Schönberger, Enrique Dunn, and Jan-Michael Frahm. Reconstructing the World* in Six Days *(As Captured by the Yahoo 100 Million Image Dataset). In *CVPR*, 2015. 1, 7, 12

[25] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 9, 10

[26] Hossam Isack and Yuri Boykov. Energy-based geometric multi-model fitting. *IJCV*, 2012. 2, 8, 13

[27] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A convolutional network for real-time 6-DoF camera relocalization. In *ICCV*, 2015. 2

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014. 9

[29] Florian Kluger, Hanno Ackermann, Michael Ying Yang, and Bodo Rosenhahn. Deep learning for vanishing point detection using an inverse gnomonic projection. In *GCPR*, 2017. 3, 7, 13

[30] Florian Kluger, Hanno Ackermann, Michael Ying Yang, and Bodo Rosenhahn. Temporally consistent horizon lines. In *ICRA*, 2020. 1

[31] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 1955. 4

[32] José Lezama, Rafael Grompone von Gioi, Gregory Randall, and Jean-Michel Morel. Finding vanishing points via point alignments in image primal and dual domains. In *CVPR*, 2014. 3

[33] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 9

[34] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 1, 5, 12

[35] Luca Magri and Andrea Fusiello. T-linkage: A continuous relaxation of j-linkage for multi-model fitting. In *CVPR*, 2014. 2, 3, 7, 8, 13

[36] Luca Magri and Andrea Fusiello. Robust multiple model fitting with preference analysis and low-rank approximation. In *BMVC*, 2015. 2, 3, 7, 8, 13

[37] Luca Magri and Andrea Fusiello. Multiple model fitting as a set coverage problem. In *CVPR*, 2016. 2, 3, 7, 8, 13

[38] Luca Magri and Andrea Fusiello. Fitting multiple heterogeneous models by multi-class cascaded t-linkage. In *CVPR*, 2019. 2, 3, 7, 8, 14

[39] Raul Mur-Artal and Juan D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras. *T-RO*, 2017. 1

[40] D Nasuto and JM Bishop R Craddock. NAPSAC: High noise, high dimensional robust estimation-it's in the bag. In *BMVC*, 2002. 3

[41] 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. 9

[42] Trung Thanh Pham, Tat-Jun Chin, Konrad Schindler, and David Suter. Interacting geometric priors for robust multimodel fitting. *Transactions on Image Processing*, 2014. 2

[43] Pulak Purkait, Tat-Jun Chin, Hanno Ackermann, and David Suter. Clustering with hypergraphs: The case for large hyperedges. In *ECCV*, 2014. 3

[44] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 9

[45] François Rameau, Hyowon Ha, Kyungdon Joo, Jinsoo Choi, Kibaek Park, and In So Kweon. A real-time augmented reality system to see-through cars. *TVCG*, 2016. 1

[46] René Ranftl and Vladlen Koltun. Deep fundamental matrix estimation. In *ECCV*, 2018. 2

[47] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *TPAMI*, 2016. 1

[48] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *CVPR*, 2016. 1

[49] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *ECCV*, 2012. 5

[50] Gilles Simon, Antoine Fond, and Marie-Odile Berger. A-contrario horizon-first vanishing point detection using second-order grouping laws. In *ECCV*, 2018. 3, 7, 13

[51] Christoph Strecha, Wolfgang Von Hansen, Luc Van Gool, Pascal Fua, and Ulrich Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *CVPR*, 2008. 7, 12

[52] Weiwei Sun, Wei Jiang, Eduard Trulls, Andrea Tagliasacchi, and Kwang Moo Yi. Attentive context normalization for robust permutation-equivariant learning. In *CVPR*, 2020. 2

[53] Jean-Philippe Tardif. Non-iterative approach for fast and accurate vanishing point detection. In *ICCV*, 2009. 3

[54] Roberto Toldo and Andrea Fusiello. Robust multiple structures estimation with j-linkage. In *ECCV*, 2008. 2, 3, 5, 6, 11, 12, 13

[55] Philip HS Torr and Andrew Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 2000. 3

[56] Roberto Tron and René Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. In *CVPR*, 2007. 2, 5, 6

[57] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. In *CoRR*, 2016. 9, 10

[58] Andrea Vedaldi and Andrew Zisserman. Self-similar sketch. In *ECCV*, 2012. 3

[59] Etienne Vincent and Robert Laganiére. Detecting planar homographies in an image pair. In *ISPA*, 2001. 2, 4, 7, 8, 13

[60] Rafael Grompone Von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. LSD: A fast line segment detector with a false detection control. *TPAMI*, 2008. 5

[61] Bastian Wandt and Bodo Rosenhahn. Repnet: Weakly supervised training of an adversarial reprojection network for 3d human pose estimation. In *CVPR*, 2019. 1

[62] Horst Wildenauer and Allan Hanbury. Robust camera self-calibration from monocular images of Manhattan worlds. In *CVPR*, 2012. 3

[63] Hoi Sim Wong, Tat-Jun Chin, Jin Yu, and David Suter. Dynamic and hierarchical multi-structure geometric model fitting. In *ICCV*, 2011. 2, 5, 6, 7, 8, 13, 14, 17

[64] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3D: A database of big spaces reconstructed using SfM and object labels. In *ICCV*, 2013. 7, 12

[65] Yiliang Xu, Sangmin Oh, and Anthony Hoogs. A minimum error vanishing point detection approach for uncalibrated monocular images of man-made environments. In *CVPR*, 2013. 3

[66] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *CVPR*, 2018. 2, 6, 9, 10

[67] Menghua Zhai, Scott Workman, and Nathan Jacobs. Detecting vanishing points using global image context in a non-manhattan world. In *CVPR*, 2016. 3, 7, 13

[68] Wei Zhang and Jana Kŏsecká. Nonparametric estimation of multiple structures with outliers. In *Dynamical Vision*. 2007. 3