CONSAC: Robust Multi-Model Fitting by Conditional Sample Consensus
Supplementary Material

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

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. 1.

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 $\{\mathbf{y}_1, \dots, \mathbf{y}_C\}$ with $\mathbf{y} \sim p(\mathbf{y}|\mathbf{s}; \mathbf{w})$.
            $\mathbf{h} \leftarrow f_{\mathsf{S}}(\{\mathbf{y}_1, \dots, \mathbf{y}_C\})$
            $\mathcal{H} \leftarrow \mathcal{H} \cup \{\mathbf{h}\}$
        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
$\hat{\mathcal{M}} \leftarrow \arg\max_{\mathcal{M} \in \mathcal{P}} g_{\mathsf{m}}(\mathcal{M}, \mathcal{Y})$

			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	$\|\mathcal{Y}\|$	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	$\|\mathcal{Y}\|$	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 1: **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. Neural Network

We use a neural network similar to PointNet [20] and based on [4, 28] for prediction of conditional sampling weights in CONSAC. Fig. 1 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

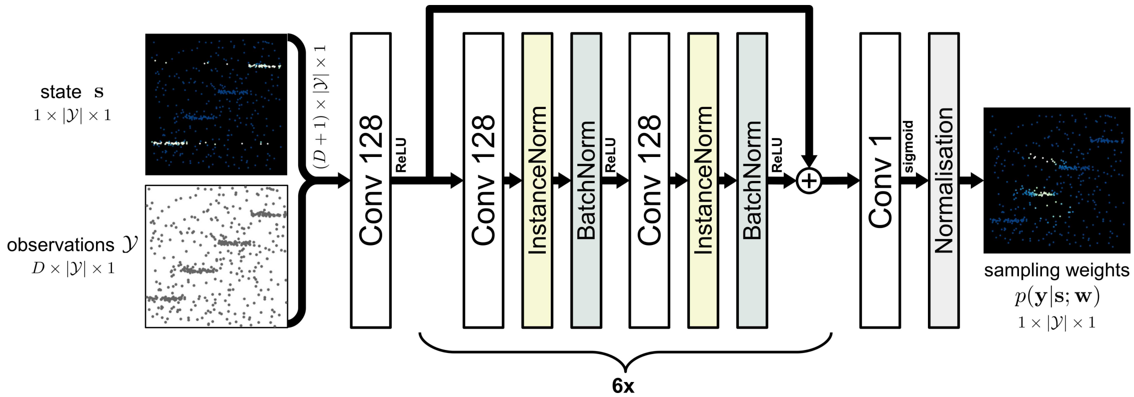

Figure 1: **CONSAC neural network architecture** used for all experiments. We stack observations $\mathcal{Y}$, e.g. line segments or point correspondences (*not* an image), and state $\mathbf{s}$ into a tensor of size $(D+1) \times |\mathcal{Y}| \times 1$, and feed it into the network. The network is composed of linear 1×1 convolutional layers interleaved with instance normalisation [24], batch normalisation [9] and ReLU [6] layers which are arranged as residual blocks [7]. Only using 1×1 convolutions, the network is order invariant w.r.t. observations $\mathcal{Y}$. The architecture is based on [4, 28].

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 [6] activation function, we apply six residual blocks [7]. Each residual block is composed of two series of convolutions (1×1, 128 channels), instance normalisation [24], batch normalisation [9] (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 [19] version 1.2.0.

A.1.1 Training Procedure

We train the neural network using the Adam [11] optimiser and utilise a cosine annealing learning rate schedule [13]. 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 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.*:

$$\ell = \ell_{\mathsf{self}} + \kappa \cdot \ell_{\mathsf{im}} \tag{1}$$

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

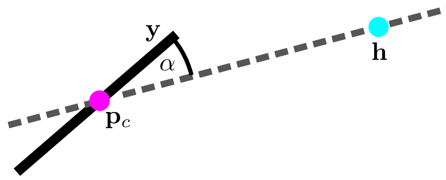

Figure 2: Visualisation of the angle α used for the vanishing point estimation residual function $r(\mathbf{y}, \mathbf{h})$.

from [3]:

$$g_\mathsf{i}(\mathbf{y}, \mathbf{h}) = 1 - \sigma(\beta r(\mathbf{y}, \mathbf{h}) - \beta\tau)\,, \tag{2}$$

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:

$$\sigma(x) = \frac{1}{1 + e^{-x}}\,. \tag{3}$$

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

$$g_\mathsf{m}(\mathcal{M}, \mathcal{Y}) = \sum_{\mathbf{y} \in \mathcal{Y}} \max_{\mathbf{h} \in \mathcal{M}} g_\mathsf{i}(\mathbf{y}, \mathbf{h})\,. \tag{4}$$

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:

$$g_\mathsf{s}(\mathbf{h}, \mathcal{Y}, \mathcal{M}) = g_\mathsf{m}(\mathcal{M} \cup \{\mathbf{h}\}, \mathcal{Y})\,. \tag{5}$$

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:

$$r(\mathbf{y}, \mathbf{h}) = |\mathbf{y}^\mathsf{T}\mathbf{h}|\,. \tag{6}$$

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. 2, 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:

$$r(\mathbf{y}, \mathbf{h}) = 1 - \cos\alpha = 1 - \frac{|\mathbf{l}_{\mathbf{y},1:2}^\mathsf{T}\mathbf{l}_{c,1:2}|}{\|\mathbf{l}_{\mathbf{y},1:2}\|\|\mathbf{l}_{c,1:2}\|}\,. \tag{7}$$

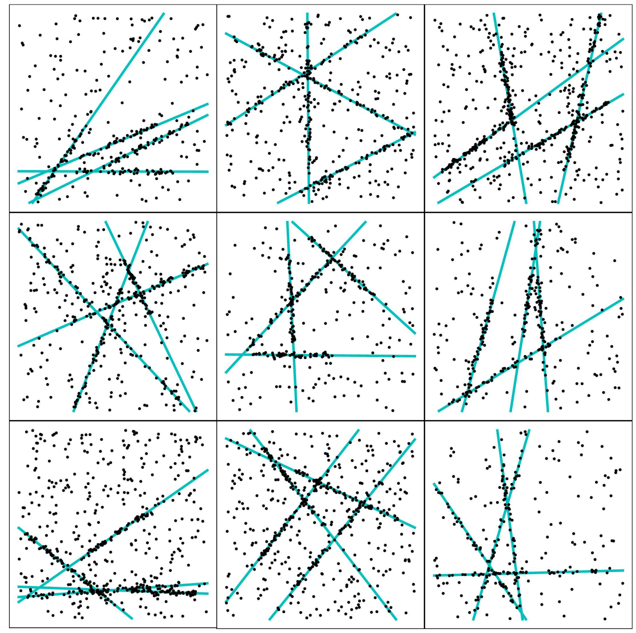

Figure 3: **Line fitting:** we show examples from the synthetic dataset we used to train CONSAC on the line fitting problem. Each scene consists of four lines placed at random, with points sampled along them, perturbed by Gaussian noise and outliers. Cyan = ground truth lines.

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:

$$r(\mathbf{y}, \mathbf{h}) = \|\mathbf{p}_1 - \mathbf{p}_1'\|^2 + \|\mathbf{p}_2 - \mathbf{p}_2'\|^2\,, \tag{8}$$

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. 3 shows a few examples from this dataset.

For evaluation, we use the synthetic *stair4*, *star5* and *star11* scenes from [23], which were also used by [2]. As Fig. 4 shows, each scene consists of 2D points forming four,

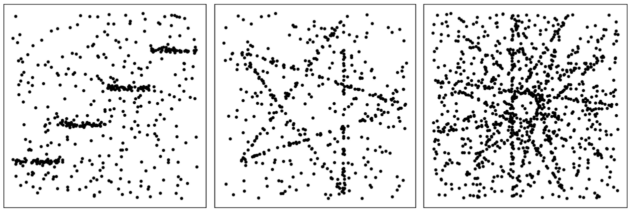

Figure 4: **Line fitting:** we use the synthetic *stair4* (left), *star5* (middle) and *star11* (right) scenes from [23], which were also used by [2], in our experiments.

five or eleven line segments. The points are perturbed by Gaussian noise ($\sigma = 0.0075$) and contain $50 - 60\%$ outliers.

B.2. Vanishing Point Estimation

NYU-VP In Fig. 5 (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. 5 (bottom), we show a histogram of the number of vanishing points per image in our new YUD+ dataset extension. By comparison, the original YUD [5] 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 [14] feature correspondences extracted from the structure-from-motion scenes of [8, 22, 27]. Specifically, we used the outdoor scenes *Buckingham*, *Notredame*, *Sacre Coeur*, *St. Peter's* and *Reichstag* from [8], *Fountain* and *Herzjesu* from [22], and 16 indoor scenes from SUN3D [27]. We use the SIFT correspondences computed and provided by Brachmann and Rother [4], 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 [23] for various combinations of single-instance samples S and multi-instance samples P. As Fig. 6 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. 7 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.

C.2. Vanishing Point Estimation

Evaluation Metric We denote ground truth VPs of an image by $\mathcal{V} = \{\mathbf{v}_1, \dots, \mathbf{v}_M\}$ and estimates by $\hat{\mathcal{V}} = \{\hat{\mathbf{v}}_1, \dots, \hat{\mathbf{v}}_N\}$. 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 $\mathbf{K}$:

$$e(\mathbf{v}, \hat{\mathbf{v}}) = \arccos \frac{\left| \left(\mathbf{K}^{-1}\mathbf{v}\right)^{\mathsf{T}} \mathbf{K}^{-1}\hat{\mathbf{v}} \right|}{||\mathbf{K}^{-1}\mathbf{v}|| \cdot ||\mathbf{K}^{-1}\hat{\mathbf{v}}||} . \qquad (9)$$

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. 8 for our new NYU-VP dataset, for our YUD+ dataset extension, as well as the original YUD [5]. We compare CONSAC with the robust

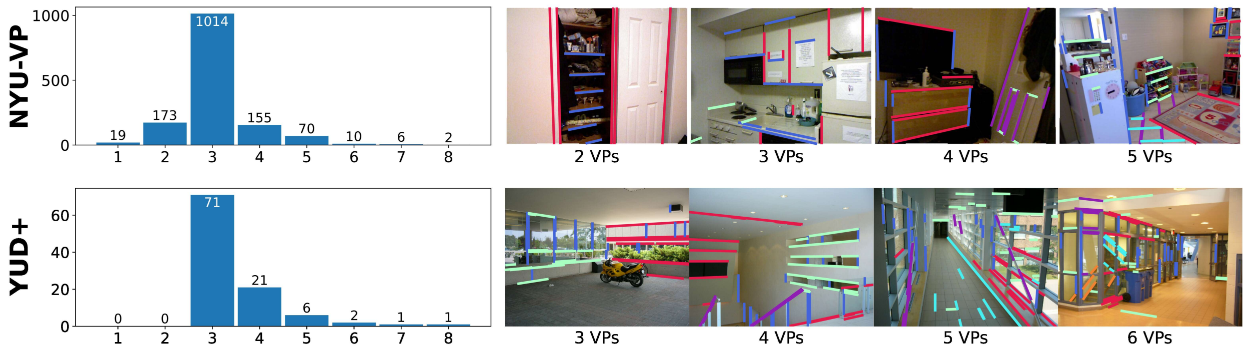

Figure 5: **Vanishing points per scene:** Histograms showing the numbers of vanishing point instances per image for our new NYU-VP dataset (top) and our YUD+ dataset extension (bottom), in addition to a few example images. We illustrate the vanishing points present in each example via colour-coded line segments.

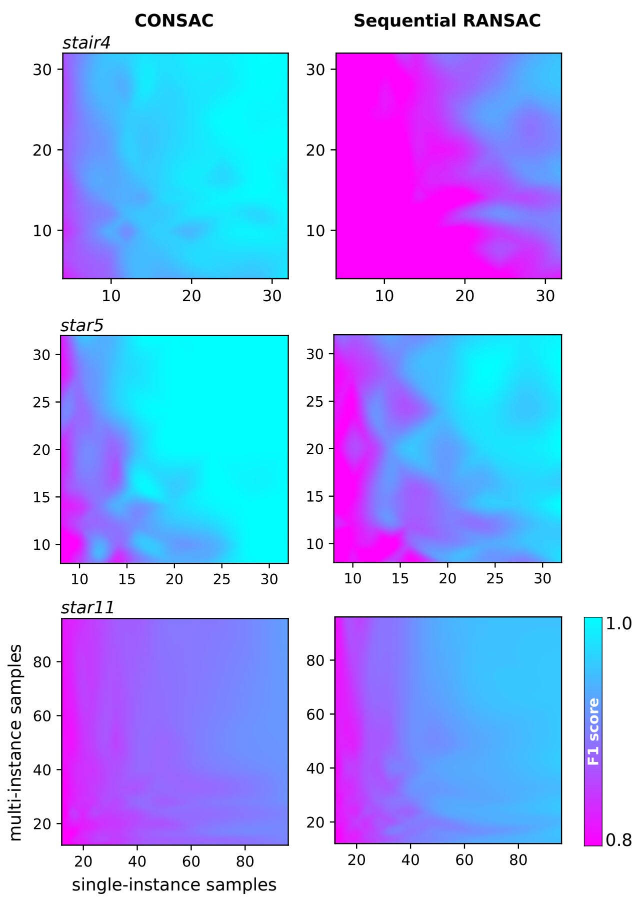

Figure 6: **Line fitting:** Using the *stair4* (top), *star5* (middle) and *star11* (bottom) line fitting scenes from [23], we compute the $F1$ scores for various combinations of single-instance samples S (abscissa) and multi-instance samples P (ordinate) and plot them as a heat map. We compare CONSAC (left) with Sequential RANSAC (right). Magenta = low, cyan = high $F1$ score.

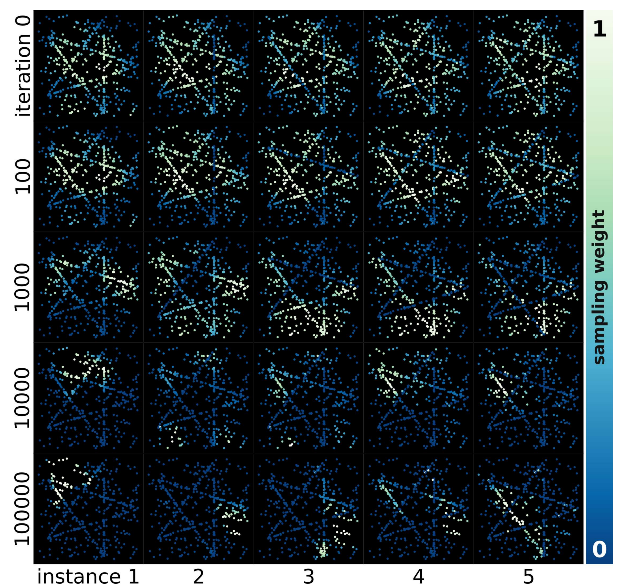

Figure 7: **Line fitting:** We show how the sampling weights at each instance sampling step develop as neural network training progresses, using the *star5* line fitting scene from [23] as an example. Each row depicts the sampling weights used to sample the eventually selected best multi-hypothesis $\hat{\mathcal{M}}$. **Top to bottom:** training iterations $0 - 100000$. **Left to right:** model instance sampling steps $1 - 5$. Sampling weights: Blue = low, white = high.

multi-model fitting approaches T-Linkage [15], Sequential RANSAC [25], Multi-X [1], RPA [16] and RansaCov [17], as well as the task-specific vanishing point estimators of Zhai et al. [29], Simon et al. [21] and Kluger et al. [12]. 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

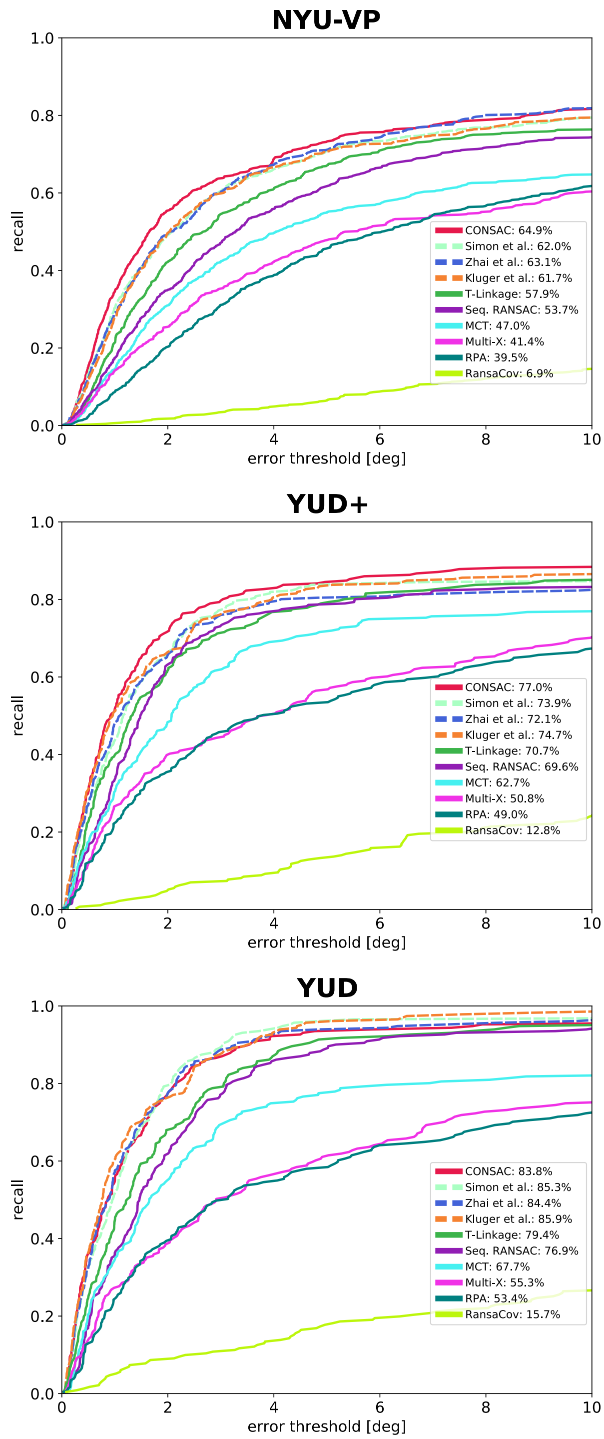

Figure 8: **Vanishing point estimation:** Recall curves for errors up to 10° for all methods which we considered in our experiments. We selected the result with the *median* AUC out of five runs for each method. Robust estimators are represented with solid lines, task-specific VP estimators with dashed lines. **Top:** Results on our new NYU-VP dataset. **Middle:** Results on our new YUD+ dataset extension. **Bottom:** Results on the original YUD [5].

	no. of planes	CONSAC-S	MCT [18]	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 2: **Homography estimation:** Misclassification errors (in %, average over five runs) for all homography estimation scenes of AdelaideRMF [26].

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 [5], 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. 10, we show additional qualitative results from the NYU-VP dataset, and in Fig. 11, we show additional qualitative results from the YUD+ dataset.

C.3. Homography Estimation

We provide results computed on AdelaideRMF [26] for all scenes seperately. In Fig. 9, we compare CONSAC-S – *i.e.* CONSAC trained in a self-supervised manner – to Progressive-X [2], Multi-X [1], PEARL [10], RPA [16], RansaCov [17] and T-Linkage [15]. We adapted the graph directly from [2]. CONSAC-S achieves state-of-the-art performance on 13 of 19 scenes. Tab. 2 compares CONSAC-S with MCT [18] 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. 12, we show additional qualitative results from the AdelaideRMF [26] dataset.

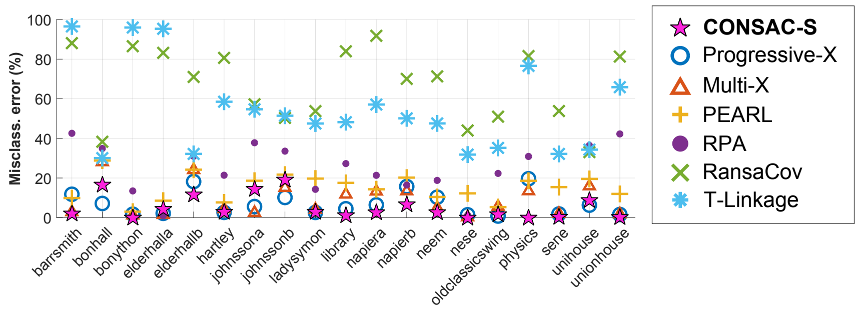

Figure 9: **Homography estimation:** Misclassification errors (in %, average over five runs) for all homography estimation scenes of AdelaideRMF [26]. Graph adapted from [2].

References

[1] Daniel Barath and Jiri Matas. Multi-class model fitting by energy minimization and mode-seeking. In *ECCV*, 2018. 5, 6

[2] Daniel Barath and Jiri Matas. Progressive-X: Efficient, anytime, multi-model fitting algorithm. *ICCV*, 2019. 3, 4, 6, 7

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

[4] Eric Brachmann and Carsten Rother. Neural-guided RANSAC: Learning where to sample model hypotheses. In *ICCV*, 2019. 1, 2, 4

[5] Patrick Denis, James H Elder, and Francisco J Estrada. Efficient edge-based methods for estimating manhattan frames in urban imagery. In *ECCV*, 2008. 4, 6

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *ICCV*, 2015. 2

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

[8] 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. 4

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 2

[10] Hossam Isack and Yuri Boykov. Energy-based geometric multi-model fitting. *IJCV*, 2012. 6

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014. 2

[12] Florian Kluger, Hanno Ackermann, Michael Ying Yang, and Bodo Rosenhahn. Deep learning for vanishing point detection using an inverse gnomonic projection. In *GCPR*, 2017. 5

[13] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 2

[14] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 4

[15] Luca Magri and Andrea Fusiello. T-linkage: A continuous relaxation of j-linkage for multi-model fitting. In *CVPR*, 2014. 5, 6

[16] Luca Magri and Andrea Fusiello. Robust multiple model fitting with preference analysis and low-rank approximation. In *BMVC*, 2015. 5, 6

[17] Luca Magri and Andrea Fusiello. Multiple model fitting as a set coverage problem. In *CVPR*, 2016. 5, 6

[18] Luca Magri and Andrea Fusiello. Fitting multiple heterogeneous models by multi-class cascaded t-linkage. In *CVPR*, 2019. 6

[19] 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. 2

[20] 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. 1

[21] Gilles Simon, Antoine Fond, and Marie-Odile Berger. A-contrario horizon-first vanishing point detection using second-order grouping laws. In *ECCV*, 2018. 5

[22] 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. 4

[23] Roberto Toldo and Andrea Fusiello. Robust multiple structures estimation with j-linkage. In *ECCV*, 2008. 3, 4, 5

[24] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. In *CoRR*, 2016. 2

[25] Etienne Vincent and Robert Laganiére. Detecting planar homographies in an image pair. In *ISPA*, 2001. 5

[26] Hoi Sim Wong, Tat-Jun Chin, Jin Yu, and David Suter. Dynamic and hierarchical multi-structure geometric model fitting. In *ICCV*, 2011. 6, 7, 10

[27] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3D: A database of big spaces reconstructed using SfM and object labels. In *ICCV*, 2013. 4

[28] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *CVPR*, 2018. 1, 2

[29] Menghua Zhai, Scott Workman, and Nathan Jacobs. Detecting vanishing points using global image context in a non-manhattan world. In *CVPR*, 2016. 5

Figure 10: 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.

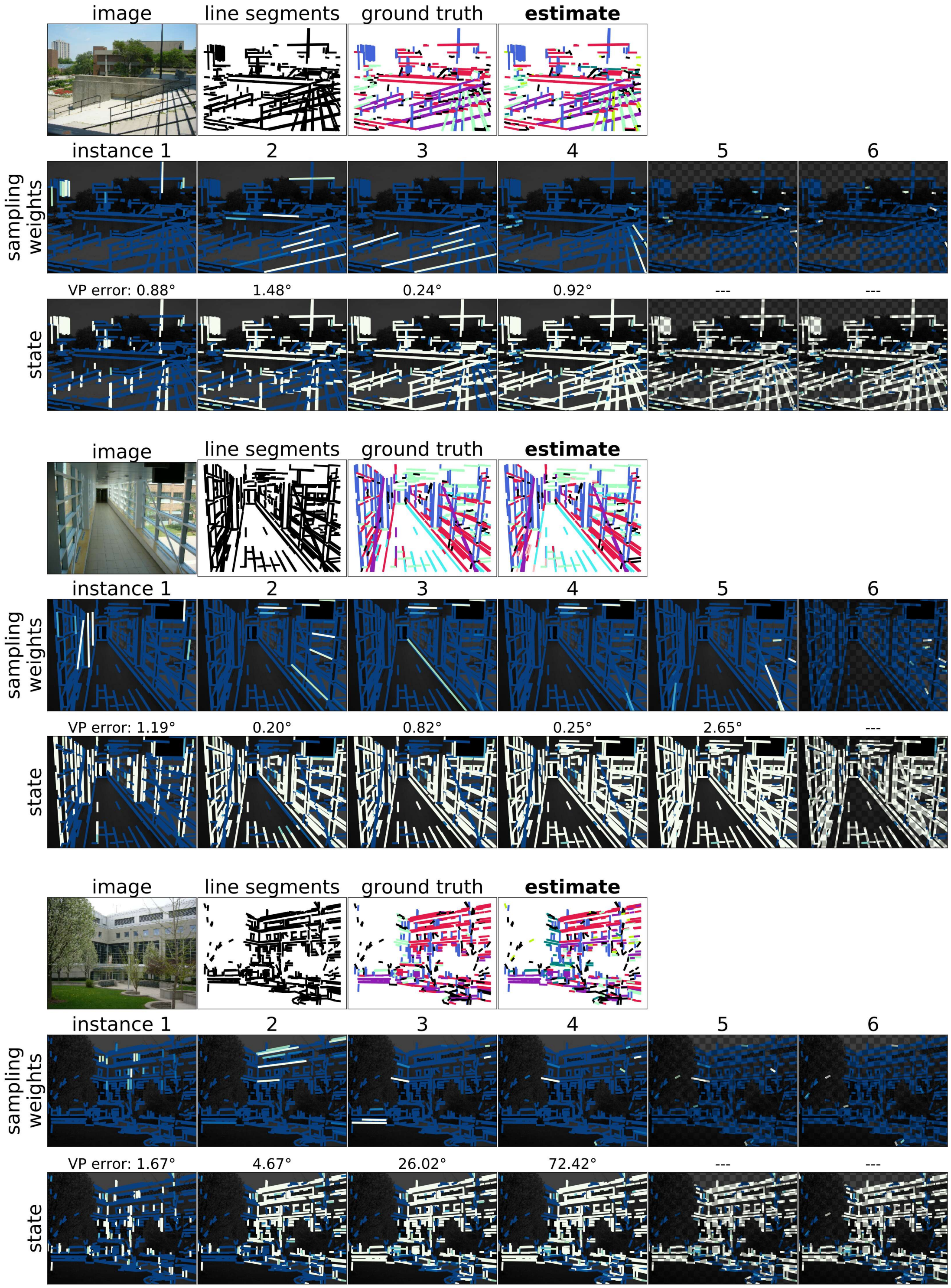

Figure 11: Three qualitative examples for VP estimation with CONSAC on the YUD+ dataset. For each example we show the original image, extracted line segments, line assignments to ground truth VPs, and to final estimates in the first row. In the second and third row, we visualise the generation of the 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.

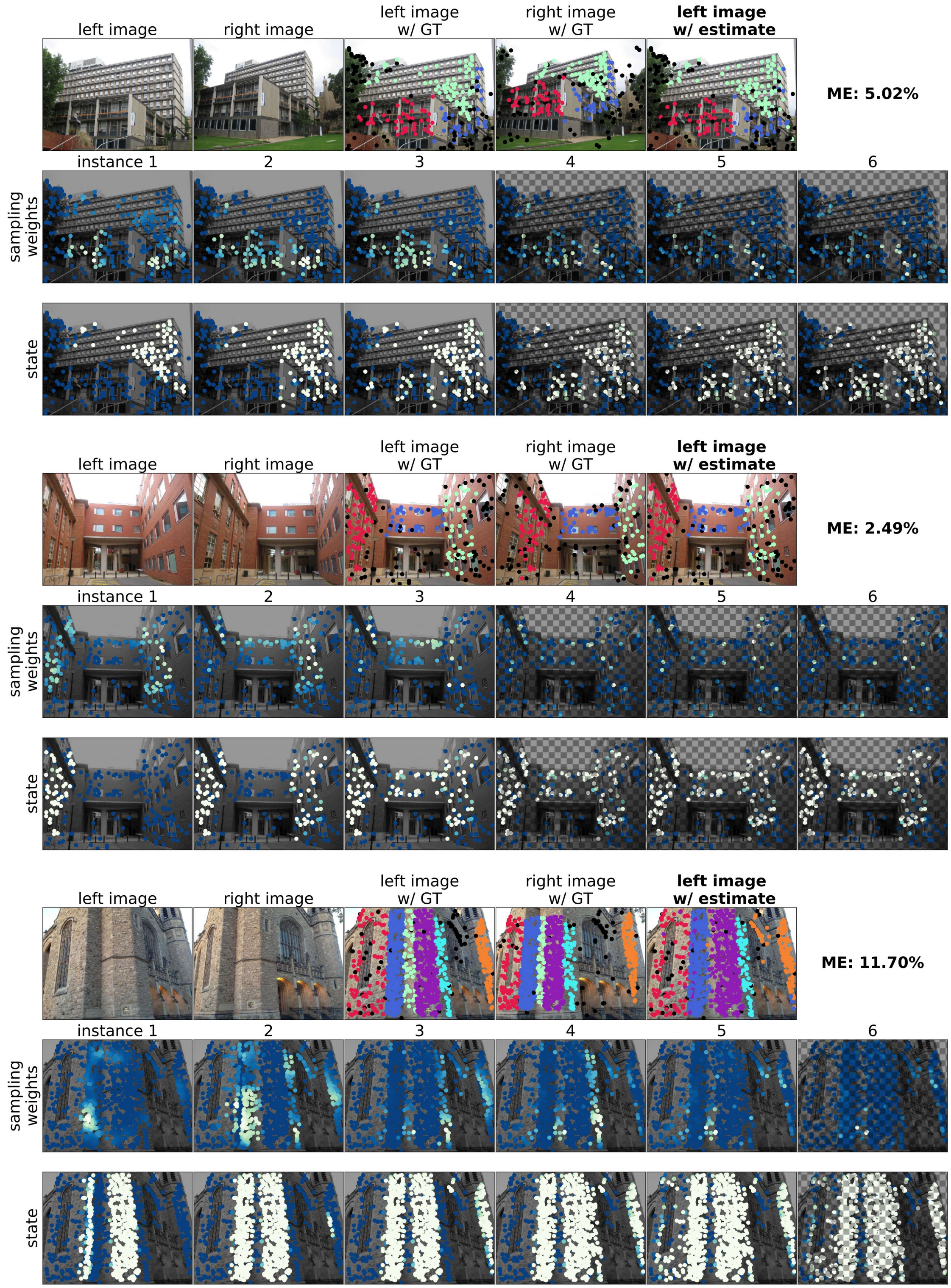

Figure 12: Three qualitative examples for homography estimation with CONSAC-S on the AdelaideRMF [26] dataset. For each example we show the original images, points with ground truth labels, final estimates, and the misclassification error (ME) in the first row. In the second and third row, we visualise 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.