PARSAC: Accelerating Robust Multi-Model Fitting with Parallel Sample Consensus

Florian Kluger, Bodo Rosenhahn

Leibniz University Hannover
kluger@tnt.uni-hannover.de

Abstract

We present a real-time method for robust estimation of multiple instances of geometric models from noisy data. Geometric models such as vanishing points, planar homographies or fundamental matrices are essential for 3D scene analysis. Previous approaches discover distinct model instances in an iterative manner, thus limiting their potential for speedup via parallel computation. In contrast, our method detects all model instances independently and in parallel. A neural network segments the input data into clusters representing potential model instances by predicting multiple sets of sample and inlier weights. Using the predicted weights, we determine the model parameters for each potential instance separately in a RANSAC-like fashion. We train the neural network via task-specific loss functions, i.e. we do not require a ground-truth segmentation of the input data. As suitable training data for homography and fundamental matrix fitting is scarce, we additionally present two new synthetic datasets. We demonstrate state-of-the-art performance on these as well as multiple established datasets, with inference times as small as five milliseconds per image.

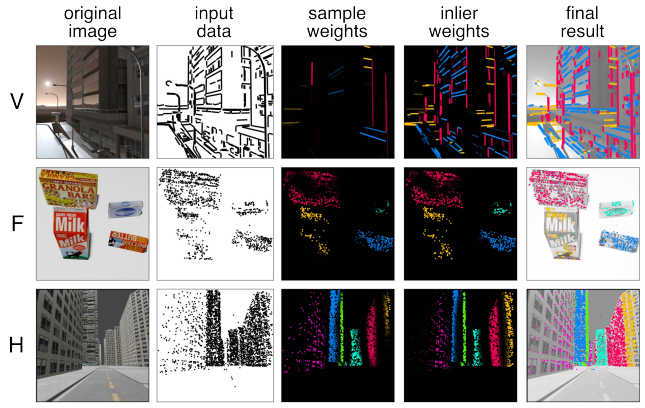

Figure 1: Applications: PARSAC estimates multiple vanishing points (V, top), fundamental matrices (F, middle) or homographies (H, bottom). We visualise distinct model instances using different colour hues. Brightness in columns three and four is proportional to the corresponding weight.

1 Introduction

In Computer Vision, we commonly aim to explain high-dimensional and noisy observations using low-dimensional geometric models. These models can give important insights into the structure of our data, and are crucial for 3D scene analysis and reconstruction. For example, by fitting vanishing points to a set of 2D line segments, we can deduce information about the 3D layout of a scene from a single view (Zhou et al. 2019b; Zou et al. 2018). Similarly, fitting homographies to point correspondences of image pairs allows us to reason about dominant 3D planes within a scene. Robust estimation of fundamental matrices is indispensable for applications such as SLAM (Mur-Artal and Tardós 2017) and two-view motion segmentation (Ozbay, Camps, and Sznaier 2022). We typically obtain the features for model fitting using low-level algorithms, such as SIFT (Lowe 2004) and LSD (Von Gioi et al. 2008). As these methods are imperfect, model fitting algorithms must be able to identify erroneous features (outliers) in order to robustly determine model parameters using valid features (inliers) only (Fischler and Bolles 1981). Given only one instance of a geometric model in our data, we thus have to segment our data into two groups: inliers and outliers. However, as the number of model instances increases, the number of data groups also increases, and inliers of one model act as pseudo-outliers for all other models. Consequently, robustly estimating the parameters of all models becomes more difficult, especially if the number of model instances is unknown beforehand.

Early approaches solve this problem by repeatedly applying a robust estimator like RANSAC (Fischler and Bolles 1981). After each step, they remove all data points associated with previously predicted models (Vincent and Laganiére 2001). More recent methods instead generate model hypotheses beforehand, then use clustering and optimisation techniques in order to assign data points to a model or the outlier class (Barath and Matas 2018; Barath, Matas, and Hajder 2016; Pham et al. 2014; Amayo et al. 2018; Isack and Boykov 2012; Toldo and Fusiello 2008; Magri and Fusiello 2014, 2015, 2016, 2019; Chin, Wang, and Suter 2009). Hybrid approaches combine the strengths of both methodologies by interleaving multiple sampling, clustering and optimisation steps (Barath and Matas 2019; Barath et al. 2023).

Being the first approach which applies deep learning to robust multi-model fitting, CONSAC (Kluger et al. 2020b) uses a neural network to guide hypothesis sampling in a sequential model fitting pipeline. It achieves high accuracy but is computationally costly, since its sequential approach results in a large number of neural network forward passes.

In this paper, we propose a method that predicts sample weights for all model instances *simultaneously*, thus overcoming the computational bottleneck of methods such as CONSAC. In addition to sample weights for hypothesis *sampling*, we also predict inlier weights for hypothesis *selection*. Using these weights, we recover model instances in a RANSAC-like fashion independently and *in parallel*. This translates into a significant speedup in practice: our method achieves inference times as small as five milliseconds per image. It is considerably faster than its competitors while still achieving state-of-the-art accuracy. In a nod to RANSAC, we call our method PARSAC: *Parallel Sample Consensus*. As shown in Fig. 1, PARSAC can be used for various applications.

Since PARSAC is a machine learning based approach, we require suitable training data. For vanishing point estimation, we have seen an emergence of new large-scale datasets, such as SU3 (Zhou et al. 2019b) and NYU-VP (Kluger et al. 2020b), in recent years. For fundamental matrix and homography estimation, however, AdelaideRMF (Wong et al. 2011) is still the only publicly available dataset. It consists of merely 38 labelled image pairs, with no separate training set. This poses the danger that researchers inadvertently overfit their algorithms on this small dataset, because there is no other data to verify their performance on. In order to alleviate this issue, we present two new synthetic datasets: *HOPE-F* for fundamental matrix fitting and *Synthetic Metropolis Homographies (SMH)* for homography fitting.

In summary, our **main contributions**[1] are:

- PARSAC, the first learning-based *real-time* method for robust multi-model fitting. It uses a neural network to segment the data into clusters based on model instance affinity in a single forward pass. This segmentation allows for a parallelised discovery of geometric models.
- Two new large-scale synthetic datasets – *HOPE-F* and *SMH* – for fundamental matrix and homography fitting. They are the first datasets to provide sufficient training data for supervised learning of multiple fundamental matrix or planar homography fitting.
- We achieve state-of-the-art results for vanishing point estimation, fundamental matrix estimation and homography estimation on multiple datasets.
- PARSAC is significantly faster than its competitors. Requiring just five milliseconds per image for vanishing point estimation, it is five times faster than the second fastest method, and 25 times faster than the fastest competitor with comparable accuracy.

[1]Supplementary material, code and datasets are available at: `https://github.com/fkluger/parsac`

2 Related Work

2.1 Multi-Model Fitting

Robust model fitting is a technique widely used for estimating geometric models from data contaminated with outliers. The most commonly used approach, RANSAC (Fischler and Bolles 1981), randomly samples minimal sets of observations to generate model hypotheses, and selects the hypothesis with the largest consensus set, i.e. observations which are inliers. While effective in the single-instance case, RANSAC fails if multiple model instances are apparent in the data. In this case, we need to apply robust *multi-model* fitting techniques. Sequential RANSAC (Vincent and Laganiére 2001) fits multiple models sequentially by applying RANSAC repeatedly, removing inliers from the set of all observations after each iteration. More recent methods, such as PEARL (Isack and Boykov 2012), instead optimise a global mixed-integer cost function (Rosenhahn 2023) initialised via a stochastic sampling, in order to fit multiple models simultaneously (Pham et al. 2014; Amayo et al. 2018). Multi-X (Barath and Matas 2018) generalises this methodology to *multi-class* problems, i.e. cases where multiple models of possibly different types may fit the data. As a hybrid approach, Progressive-X (Barath and Matas 2019) guides hypothesis generation via intermediate estimates by interleaving sampling and optimisation steps. J- and T-Linkage (Toldo and Fusiello 2008; Magri and Fusiello 2014) both use preference analysis (Chin, Wang, and Suter 2009) to cluster observations agglomeratively, but use different methods to define the preference sets. MCT (Magri and Fusiello 2019) is a multi-class generalisation of T-Linkage, while RPA (Magri and Fusiello 2015) uses spectral instead of agglomerative clustering. Progressive-X+ (Barath et al. 2023) combines Progressive-X with preference analysis and allows observations to be assigned to multiple models. It is faster and more accurate than its predecessor. Fast-CP (Ozbay, Camps, and Sznaier 2022) – specifically tailored for fundamental matrix fitting – utilises Christoffel polynomials in order to segment observations into clusters belonging to the same model. Most recently, the authors of (Farina et al. 2023) utilise quantum computing for multi-model fitting. While seminal in this regard, they make strong assumptions such as the absence of true outliers. CONSAC (Kluger et al. 2020b) is the first approach to utilise deep learning for multi-model fitting. Using a neural network to guide the hypothesis sampling, it discovers model instances sequentially and predicts new sampling weights after each step. This technique provides high accuracy, but is computationally expensive due to its sequential operation and multi-hypothesis sampling. While PARSAC also leverages deep learning, we are able to decouple the discovery of individual model instances, reducing run-time by two orders of magnitude.

2.2 Vanishing Point Estimation

Although multiple vanishing point (VP) estimation can be described as a multi-model fitting problem, algorithms outside of this genre have also been designed to tackle this task specifically (Antunes and Barreto 2013; Barinova et al. 2010; Kluger et al. 2017; Lezama et al. 2014; Simon, Fond,

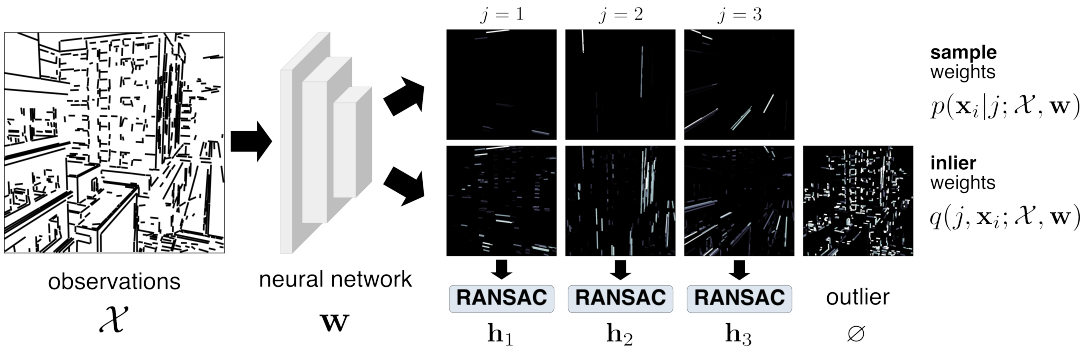

Figure 2: PARSAC Overview: Given observations $\mathcal{X}$, e.g. line segments or point correspondences, we predict sample weights p and inlier weights q for each observation and putative geometric model using a neural network. For each putative geometric model j, we independently sample model hypotheses in a RANSAC-like fashion, using the predicted sample weights. We then select the best models which have the largest weighted inlier counts, using the predicted inlier weights. An additional set of weights captures potential outliers.

and Berger 2018; Tardif 2009; Vedaldi and Zisserman 2012; Wildenauer and Hanbury 2012; Xu, Oh, and Hoogs 2013; Zhai, Workman, and Jacobs 2016; Zhou et al. 2019a; Liu, Zhou, and Zhao 2021; Wu et al. 2021; Lin et al. 2022). Unlike more general multi-model fitting methods, they usually exploit additional, domain-specific knowledge. Zhai et al. first predict a horizon line (Kluger et al. 2020a) from the RGB image via a convolutional neural network (CNN), on which they then condition their VP estimates. Simon et al. also condition some VPs on the horizon line, as well as on the zenith VP. Kluger et al. predict initial VP estimates via a CNN, and refine them using an expectation maximisation (Dempster, Laird, and Rubin 1977) algorithm. Zhou et al. propose a CNN with a conic convolution operator in order to find VPs for Manhattan-world scenes. Lin et al. incorporate a Hough transform and a Gaussian sphere mapping into their CNN in order to exploit geometric priors induced by the VP fitting problem. General purpose robust multi-model fitting algorithms, such as our proposed PARSAC, do not rely on such domain-specific priors.

3 Method

From a set of N observations $\mathbf{x}_i \in \mathcal{X}$, contaminated with noise and outliers, we seek to detect M instances of a model $\mathbf{h}$ that describe our data. As visualised in Fig. 2, PARSAC discovers the model instances $\mathcal{M} = (\mathbf{h}_1, \dots, \mathbf{h}_M)$ in parallel, guided by a neural network with parameters $\mathbf{w}$:

1. For each observation $\mathbf{x}_i$ and putative model instance $\mathbf{h}_j$, the network predicts $M^* \geq M$ sample weights $p(\mathbf{x}_i|j;\mathcal{X},\mathbf{w})$ and inlier weights $q(j,\mathbf{x}_i;\mathcal{X},\mathbf{w})$.
2. We generate a set of putative model instances $\mathcal{M}^* = \{\mathbf{h}_1, \dots, \mathbf{h}_{M^*}\}$ in parallel in a RANSAC-like fashion. The sample weights guide the sampling of hypotheses, and we perform hypothesis selection via *weighted* inlier counting using the inlier weights.
3. We greedily rank the putative instances based on unique and overlapping inlier counts and discard models with an insufficient amount of inliers, giving us the final sequence of model instances $\mathcal{M} \subseteq \mathcal{M}^*$.

Sample and Inlier Weight Prediction For each observation $\mathbf{x}_i \in \mathcal{X}$ and putative model $\mathbf{h}_j \in \mathcal{M}^*$, we predict sample weights $p(\mathbf{x}_i|j;\mathcal{X},\mathbf{w})$ and inlier weights $q(j,\mathbf{x}_i;\mathcal{X},\mathbf{w})$. In addition, we predict outlier weights $q(\varnothing,\mathbf{x}_i;\mathcal{X},\mathbf{w})$. A neural network with parameters $\mathbf{w}$ computes two weight matrices $\mathbf{P}, \mathbf{Q}$ with size $N \times M^* + 1$. By normalising over the first or second dimension, respectively, we obtain the sample and inlier weights:

$$p(\mathbf{x}_i|j;\mathcal{X},\mathbf{w}) = \frac{\mathbf{P}_{i,j}}{\sum_{k=1}^{N} \mathbf{P}_{k,j}}\,, \tag{1}$$

$$q(j,\mathbf{x}_i;\mathcal{X},\mathbf{w}) = \frac{\mathbf{Q}_{i,j}}{\sum_{k=1}^{M^*+1} \mathbf{Q}_{i,k}}\,. \tag{2}$$

Sample weights $p(\mathbf{x}_i|j;\mathcal{X},\mathbf{w})$ determine how likely an observation $\mathbf{x}_i$ is to be sampled for generating a model hypothesis for the putative instance $\mathbf{h}_j$, while inlier weights $q(j,\mathbf{x}_i;\mathcal{X},\mathbf{w})$ predict whether an observation $\mathbf{x}_i$ is an inlier of putative instance $\mathbf{h}_j$. The additional outlier weights:

$$q(\varnothing,\mathbf{x}_i;\mathcal{X},\mathbf{w}) = q(M^*+1,\mathbf{x}_i;\mathcal{X},\mathbf{w})\,, \tag{3}$$

are not used for hypothesis sampling or selection, but allow the neural network to remove outliers from the putative models so that they do not interfere with the following steps.

Parallel Hypothesis Sampling and Selection For each putative model $\mathbf{h}_j$, we sample S minimal sets of observations $\mathcal{C} = \{\mathbf{x}_1, \dots, \mathbf{x}_C\} \subset \mathcal{X}$ independently and in parallel, using the predicted sample weights $p(\mathbf{x}_i|j;\mathcal{X},\mathbf{w})$. A minimal solver f_S then computes parameters of a model hypothesis $\mathbf{h}' = f_\mathsf{S}(\mathcal{C})$ for each minimal set. We thus get a set of model hypotheses $\mathcal{S}_j = \{\mathbf{h}'_{j,1}, \dots, \mathbf{h}'_{j,S}\}$ for each putative model. Given a distance metric $d(\mathbf{x},\mathbf{h})$ measuring the error between an observation $\mathbf{x}$ and a geometric model $\mathbf{h}$, we then compute the *weighted* inlier counts of all hypotheses:

$$I_\mathrm{w}(\mathbf{h}'_{j,k};\mathcal{X},\mathbf{w}) = \sum_{\mathbf{x}_i \in \mathcal{X}} s(d(\mathbf{x}_i,\mathbf{h}'_{j,k})) \cdot q(j,\mathbf{x}_i;\mathcal{X},\mathbf{w})\,, \tag{4}$$

using the predicted inlier weights and an inlier scoring function $s(\cdot) \in [0,1]$. We implement $s(\cdot)$ as a soft inlier measure,

with inlier threshold τ and softness parameter β, as proposed by (Brachmann and Rother 2018). During inference, we then select the hypothesis with the largest weighted inlier count:

$$\mathbf{h}_j = \underset{\mathbf{h}'_{j,k} \in \mathcal{S}_j}{\arg\max} I_{\mathrm{w}}(\mathbf{h}'_{j,k}; \mathcal{X}, \mathbf{w}) \,. \tag{5}$$

Combining the results of all putative models, we obtain the set of putative model instances:

$$\mathcal{M}^* = \{\mathbf{h}_1, \ldots, \mathbf{h}_{M^*}\} \,. \tag{6}$$

Using the weighted inlier counts to select the best hypothesis for each putative model is crucial. If we were to use the unweighted inlier count instead, as in (Kluger et al. 2020b) or vanilla RANSAC, we would more likely select very similar hypotheses for every putative model, i.e. the hypotheses with the largest consensus sets within all observations $\mathcal{X}$. We would thus ignore other less prominent but valid models that describe our data. Instead, our weighted inlier count ensures that we get a more diverse set of model instances if our neural network is able to segment the observations in a sufficiently meaningful way.

Instance Ranking We extract the final sequence of model instances $\mathcal{M}$ from the putative models $\mathcal{M}^*$ by greedily ranking the models in $\mathcal{M}^*$. Initially, the model sequence $\mathcal{M} = (\,)$ and the set of current inliers $\mathcal{I} = \varnothing$ are empty. Iteratively, we determine the number of unique inliers $I^{\mathrm{u}}_{\mathbf{h}}$ and overlapping inliers $I^{\mathrm{o}}_{\mathbf{h}}$ of each model $\mathbf{h} \in \mathcal{M}^*$ w.r.t. $\mathcal{I}$:

$$I^{\mathrm{u}}_{\mathbf{h}} = |\mathcal{I}_{\mathbf{h}} \setminus \mathcal{I}| \,, \quad I^{\mathrm{o}}_{\mathbf{h}} = |\mathcal{I}_{\mathbf{h}} \cap \mathcal{I}| \,, \tag{7}$$

with $\mathcal{I}_{\mathbf{h}} = \{\mathbf{x} \,|\, d(\mathbf{x}, \mathbf{h}) < \tau, \, \mathbf{x} \in \mathcal{X}\}$ being the set of inliers for $\mathbf{h}$ using inlier threshold τ. We then select the model which maximises unique and minimises overlapping inliers: $\mathbf{h} = \arg\max_{\mathbf{h}^* \in (\mathcal{M}^* \setminus \mathcal{M})} I^{\mathrm{u}}_{\mathbf{h}^*} - I^{\mathrm{o}}_{\mathbf{h}^*}$.

If the number of unique inliers is larger than the number of overlapping inliers by at least the minimal set size, i.e. $I^{\mathrm{u}}_{\mathbf{h}} - I^{\mathrm{o}}_{\mathbf{h}} \geq C$, we append $\mathbf{h}$ to $\mathcal{M}$ and update $\mathcal{I}$, and repeat the process: $\mathcal{M} \leftarrow \mathcal{M} \frown (\mathbf{h}) \,, \quad \mathcal{I} \leftarrow \mathcal{I} \cup \mathcal{I}_{\mathbf{h}} \,.$

Cluster Assignment After ranking model instances, we cluster observations $\mathbf{x} \in \mathcal{X}$ by model affinity. Starting with the first model $\mathbf{h}_1 \in \mathcal{M}$, we sequentially assign observation $\mathbf{x}$ to a model $\mathbf{h}_j$ if either $d(\mathbf{x}, \mathbf{h}_j) < \tau$ and $\mathbf{h}_j \in \arg\min_{\mathbf{h} \in \mathcal{M}} d(\mathbf{x}, \mathbf{h})$, or $d(\mathbf{x}, \mathbf{h}_j) < \tau_a$ and $\mathbf{x}$ is not yet assigned to a higher ranked model $\mathbf{h}_k$ with $k < j$. Here, τ denotes the inlier threshold and τ_a is the assignment threshold $\tau_a > \tau$. Observations which are not assigned to any model are labelled as outliers. The result is a set of cluster labels $\mathcal{Y}$.

3.1 Neural Network Training

We seek to optimise neural network parameters $\mathbf{w}$ such that we obtain geometric model instances $\mathcal{M}$ which accurately describe our data $\mathcal{X}$. Similarly to (Brachmann et al. 2016; Brachmann and Rother 2019; Kluger et al. 2020b, 2021), we minimise the expected value of a task loss $\ell(\mathcal{M})$ which measures the quality of the estimated model instances:

$$\mathcal{L}(\mathbf{w}) = \mathbb{E}_{\mathcal{M} \sim p(\mathcal{M}|\mathcal{X};\mathbf{w})} \left[\ell(\mathcal{M})\right] \,. \tag{8}$$

In order to facilitate this, we must turn the deterministic hypothesis selection of Eq. 5 into a probabilistic action $\mathbf{h}_j \sim p(\mathbf{h}_j|\mathcal{S}_j)$. Otherwise, we would not be able to compute gradients of $\mathcal{L}$ w.r.t. inlier weights $q(j, \mathbf{x}_i; \mathcal{X}, \mathbf{w})$. Based on (Brachmann and Rother 2019), we model the likelihood of selecting a hypothesis $\mathbf{h}'$ from a set of hypotheses $\mathcal{S}$ as the softmax over weighted inlier counts I_{w}:

$$p(\mathbf{h}_j|\mathcal{S}_j; \mathcal{X}, \mathbf{w}) = \frac{\exp\left(\alpha_s \cdot I_{\mathrm{w}}(\mathbf{h}_j; \mathcal{X}, \mathbf{w})\right)}{\sum_{\mathbf{h}'_{j,k} \in \mathcal{S}_j} \exp\left(\alpha_s \cdot I_{\mathrm{w}}(\mathbf{h}'_{j,k}; \mathcal{X}, \mathbf{w})\right)} \,,$$

with α_s being a scaling factor which controls how discriminating the resulting categorical distribution becomes. We thus consider the joint probability of first sampling hypotheses sets $\mathcal{H} = \{\mathcal{S}_1, \ldots, \mathcal{S}_{M^*}\}$ and then model instances $\mathcal{M}$:

$$\begin{aligned}
&p(\mathcal{M}, \mathcal{H}|\mathcal{X}, \mathbf{w}) = p(\mathcal{M}|\mathcal{H}; \mathcal{X}, \mathbf{w}) \cdot p(\mathcal{H}|\mathcal{X}, \mathbf{w}) \,, \\
&\text{with} \quad p(\mathcal{M}|\mathcal{H}; \mathcal{X}, \mathbf{w}) = \prod_{j=1}^{M^*} p(\mathbf{h}_j|\mathcal{S}_j; \mathcal{X}, \mathbf{w}) \,, \\
&\text{with} \quad p(\mathcal{H}|\mathcal{X}, \mathbf{w}) = \prod_{j=1}^{M^*} p(\mathcal{S}_j|\mathcal{X}, \mathbf{w}) \,, \\
&\text{with} \quad p(\mathcal{S}_j|\mathcal{X}, \mathbf{w}) = \prod_{\mathbf{h}'_{j,k} \in \mathcal{S}_j} p(\mathbf{h}'_{j,k}|\mathcal{X}, \mathbf{w}) \,, \\
&\text{with} \quad p(\mathbf{h}'_{j,k}|\mathcal{X}, \mathbf{w}) = \prod_{\mathbf{x}_i \in \mathcal{C}_{j,k}} p(\mathbf{x}_i|j; \mathcal{X}, \mathbf{w}) \,.
\end{aligned}$$

Using (Schulman et al. 2015), we hence define the gradients of the expected task loss w.r.t. network parameters $\mathbf{w}$ as:

$$\frac{\partial}{\partial \mathbf{w}} \mathcal{L}(\mathbf{w}) = \mathbb{E}_{\mathcal{M},\mathcal{H}} \left[\ell(\mathcal{M}) \frac{\partial}{\partial \mathbf{w}} \log p(\mathcal{M}, \mathcal{H}|\mathcal{X}, \mathbf{w})\right] \,, \tag{9}$$

which we approximate by first drawing K samples of hypotheses sets $\mathcal{S}_j \sim p(\mathcal{S}_j|\mathcal{X}, \mathbf{w})$ for each putative model j, and then sampling $\tilde{K}$ samples of $\mathcal{M} \sim p(\mathcal{M}|\mathcal{H}; \mathcal{X}, \mathbf{w})$:

$$\frac{\partial}{\partial \mathbf{w}} \mathcal{L}(\mathbf{w}) \approx \frac{1}{K\tilde{K}} \sum_{i=1}^{K} \sum_{j=1}^{\tilde{K}} \ell(\mathcal{M}_{ij}) \frac{\partial}{\partial \mathbf{w}} \log p(\cdot) \,,$$

with $p(\cdot) = p(\mathcal{M}_{ij}, \mathcal{H}_i|\mathcal{X}, \mathbf{w})$. As in (Brachmann and Rother 2019), we subtract the mean of $\ell(\cdot)$ as a baseline to reduce the variance of the gradient.

Task Loss The type of loss we use depends on the task we seek to solve, as well as the type of ground truth data available. If we want to optimise the parameters of our models $\mathcal{M}$ w.r.t. a set of ground truth models $\hat{\mathcal{M}}$, we utilise a task specific loss $\ell_{\mathrm{s}}(\mathbf{h}, \hat{\mathbf{h}})$ which measures the error between each ground truth and estimated model. Using this loss, we determine a cost matrix $\mathbf{C}$, with $C_{ij} = \ell_{\mathrm{s}}(\mathbf{h}_i, \hat{\mathbf{h}}_j)$. If $|\mathcal{M}| > |\hat{\mathcal{M}}|$, we only consider the first $|\hat{\mathcal{M}}|$ models in $\mathcal{M}$. We then define the final task loss as the minimal assignment cost via the Hungarian method (Kuhn 1955) $h(\cdot)$:

$$\ell(\mathcal{M}) = h(\mathbf{C}). \tag{10}$$

Alternatively, if our ground truth consists of cluster labels $\hat{\mathcal{Y}}$, we compute the misclassification error (ME) w.r.t. our predicted cluster assignments $\mathcal{Y}$ which arise from $\mathcal{M}$:

$$\ell(\mathcal{M}) = \mathrm{ME}(\mathcal{Y}, \hat{\mathcal{Y}}) \,. \tag{11}$$

Eq. 9 does not require the task loss $\ell(\cdot)$ to be differentiable.

(a) HOPE-F Dataset (b) Synthetic Metropolis Homographies (SMH) Dataset

Figure 3: We propose two new datasets: HOPE-F for fundamental matrix fitting and SMH for homography fitting. For each dataset we show one example image pair: the left pair of each example shows the RGB images, the right pair visualises the pre-computed SIFT key-points, colour coded by ground truth label. Please refer to the supplementary for additional examples.

	scenes (train)	scenes (test)	instances	points	outliers
HOPE-F	3600	400	1–4	10–2k	0–90%
Adelaide-F	0	19	1–4	165–360	27–73%
SMH	44112	3890	1–32	12–16k	1–96%
Adelaide-H	0	19	1–7	106–2k	6–76%

Table 1: We compare our new HOPE-F and SMH datasets with AdelaideRMF (Wong et al. 2011). Our new datasets contain significantly more scenes, with higher numbers of annotated model instances, as well as larger ranges of key-point correspondences and outlier ratios per scene.

4 Multi-Model Fitting Datasets

A variety of tasks can be solved with robust multi-model fitting algorithms, and each task requires data with different modalities for training, parameter tuning and evaluation. Early benchmark datasets for vanishing point estimation, such as the York Urban Dataset (YUD) (Denis, Elder, and Estrada 2008) with 102 images, are relatively small. This makes them unsuitable for contemporary deep learning methods, which require large amounts of training data. Moreover, evaluation on such small datasets does not allow us to draw well-founded conclusions w.r.t. the performance of a method under a larger variety of conditions. The SceneCity Urban 3D (SU3) dataset (Zhou et al. 2019b), on the other hand, contains 23000 synthetic outdoor images showing a 3D model of a city rendered from various perspectives. They adhere to the Manhattan world assumption and are annotated with three vanishing points each. Complementary to this, the NYU-VP dataset (Kluger et al. 2020b) augments 1449 real-world non-Manhattan indoor images of the NYU Depth v2 (Silberman et al. 2012) dataset with vanishing point annotations. Its authors also presented YUD+, which adds additional non-Manhattan vanishing point annotations to YUD. For fundamental matrix and homography estimation, however, the amount of data available is still very limited. The AdelaideRMF (Wong et al. 2011) dataset contains just 19 scenes for fundamental matrices and 19 for homographies. Even though it is very small and was published over a decade ago, it is still the default benchmark for these two tasks. We therefore present two new synthetic datasets: *HOPE-F* for fundamental matrix fitting, and *Synthetic Metropolis Homographies* for homography fitting.

4.1 HOPE-F

For fundamental matrix fitting, we construct a new dataset inspired by the scenes found in Adelaide-F. We use a subset of 22 textured 3D meshes from the HOPE dataset (Tyree et al. 2022) which depict household objects. For each scene, we select between one and four of these meshes and place them on a 3D plane at randomised positions. We then render two images of the scene using one virtual camera with randomised pose and focal length. Before rendering the second image, we randomise the object positions again. With known intrinsic camera parameters $\mathbf{K}$ and relative object poses $\mathbf{R}_i, \mathbf{t}_i$, $i \in \{1, \dots, 4\}$, we compute the ground truth fundamental matrices $\mathbf{F}_i$ for each image pair. We then compute SIFT (Lowe 2004) key-point feature correspondences for each image pair. We assign each correspondence either to one of the ground truth fundamental matrices, or to the outlier class. These assignments represent the ground truth cluster labels. Via this procedure, we generate a total of 4000 image pairs with key-point features, of which we reserve 400 as the test set. Fig. 3a shows an example image pair from this dataset, which we call *HOPE-F*. As Tab. 1 shows, it is significantly larger than the Adelaide-F dataset. Please refer to the supplementary for additional details.

4.2 Synthetic Metropolis Homographies (SMH)

For homography fitting, we construct a new dataset using a synthetic 3D model of a city. Using the cars present in the 3D model as anchor points, we define seven camera trajectories, one of which is reserved for the test set. Along each trajectory, we render sequences of images with varying step sizes and focal lengths. To determine the ground truth homographies for each image pair, we first compute the normal forms $(\mathbf{n}_i, d_i)$ of the planes defined by each mesh polygon visible in the images. Using known camera intrinsics $\mathbf{K}$ and relative camera pose $\mathbf{R}, \mathbf{t}$, we compute ground truth homographies $\mathbf{H}_i$ for every plane in the scene for each image pair. We then compute SIFT correspondences for each image pair and assign them either to one of the homographies, or to the outlier class. Via this procedure, we generate a total of 48002 image pairs with key-point feature correspondences, ground truth cluster labels and ground truth homographies. Fig. 3b shows an example image pair from this dataset, which we call *Synthetic Metropolis Homographies (SMH)*. As Tab. 1 shows, it is significantly larger than the Adelaide-H dataset with a much wider range of ground truth homographies. Please refer to the supplementary for additional details.

Datasets			SU3 (Zhou et al. 2019b)						YUD (Denis, Elder, and Estrada 2008)					
Metrics	time GPU	time CPU	AUC @ 1°		AUC @ 3°		AUC @ 5°		AUC @ 3°		AUC @ 5°		AUC @ 10°	
robust estimators (on pre-extracted line segments)														
PARSAC	**4.91**	**11.59**	67.44	**±0.14**	85.70	±0.13	90.17	±0.11	**63.92**	**±0.25**	**75.90**	**±0.15**	**86.37**	**±0.08**
CONSAC	3941	2758	51.58	±0.18	78.52	**±0.10**	85.69	**±0.06**	59.06	±0.48	71.33	±0.44	82.79	±0.29
J-Linkage	—	1139	47.66	±0.54	74.96	±0.49	83.01	±0.35	55.75	±2.53	68.69	±2.22	81.06	±1.52
T-Linkage	—	271.4	40.57	±0.44	71.55	±0.26	80.90	±0.21	52.19	±1.91	66.10	±1.41	79.49	±0.87
Progressive-X	—	28.92	63.14	±0.33	80.49	±0.40	84.82	±0.40	50.41	±0.84	60.10	±0.76	68.47	±0.72
task-specific methods (full information)														
NeurVPS	766.0	—	**79.47**		**92.57**		**95.44**		50.63		63.12		77.58	
DeepVP	130.3	—	56.63		84.05		90.24		58.19		72.25		84.98	
Contrario	—	767.2	32.56	±0.19	67.85	±0.17	77.72	±0.11	59.86	±0.59	72.61	±0.47	83.14	±0.24

Table 2: Manhattan-world VP estimation: Average AUC values (in %, higher is better) and their standard deviations over five runs for vanishing point estimation on the SU3 and YUD datasets. We omit standard deviations for deterministic methods. Average run times are given in milliseconds. For robust estimators, we do not include time required for line segment detection: LSD (Von Gioi et al. 2008) requires 22.59 ms on average.

Datasets			NYU-VP (Kluger et al. 2020b)						YUD+ (Kluger et al. 2020b)					
Metrics	time GPU	time CPU	AUC @ 3°		AUC @ 5°		AUC @ 10°		AUC @ 3°		AUC @ 5°		AUC @ 10°	
robust estimators (on pre-extracted line segments)														
PARSAC	**5.16**	**9.38**	39.93	**±0.16**	51.65	**±0.10**	64.58	**±0.13**	**54.98**	±0.63	**65.48**	**±0.37**	**74.74**	±0.41
CONSAC	3843	2841	38.84	±0.34	50.58	±0.37	64.25	±0.38	52.68	**±0.45**	63.73	±0.65	74.44	±0.76
J-Linkage	—	1571	30.61	±0.81	42.26	±0.80	56.58	±0.64	48.62	±1.29	60.40	±1.10	72.36	±0.82
T-Linkage	—	278.7	30.93	±0.64	42.95	±0.68	57.75	±0.68	46.91	±0.66	59.45	±0.82	71.74	±0.54
Progressive-X	—	26.15	38.73	±0.17	49.25	±0.17	60.71	±0.19	50.13	±0.78	60.01	±0.69	68.53	±0.68
task-specific methods (full information)														
DeepVP	176.7	—	**43.54**		**55.87**		**69.53**		48.06		59.57		71.34	
Contrario	—	833.2	35.91	±0.40	47.61	±0.42	61.66	±0.38	51.50	±0.52	62.81	±0.50	72.55	**±0.38**

Table 3: Non-Manhattan VP estimation: Average AUC values (in %, higher is better) and their standard deviations over five runs for vanishing point estimation on the NYU-VP and YUD+ datasets. We omit standard deviations for deterministic methods. Average run times are given in milliseconds. For robust estimators, we do not include time required for line segment detection: LSD (Von Gioi et al. 2008) requires 22.59 ms on average.

Fundamental Matrices	time GPU	time CPU	HOPE-F ME		HOPE-F SE		Adelaide-F (Wong et al. 2011) ME		Adelaide-F (Wong et al. 2011) SE	
PARSAC	**12.25**	**19.81**	**14.97**	**±8.51**	**3.07**	**±3.39**	9.83	**±4.17**	2.80	±2.37
Fast-CP	–	33.65	24.59	±13.3	5.56	±6.67	**4.92**	±5.23	1.43	±1.31
Progressive-X+	–	67.38	43.23	±15.7	9.72	±15.8	6.57	±6.64	**0.84**	**±0.88**
Progressive-X	–	1043	22.78	±15.6	29.9	±105	12.85	±14.1	2.19	±4.51
Homographies	time GPU	time CPU	SMH ME		SMH TE		Adelaide-H (Wong et al. 2011) ME		Adelaide-H (Wong et al. 2011) TE	
PARSAC	**64.00**	1758	**20.50**	±15.5	**1.81**	**±20.8**	8.63	±8.01	5.34	±7.36
CONSAC	4314	58913	33.45	±18.5	3.34	±25.7	**5.66**	**±7.05**	3.44	±7.44
Progressive-X+	–	**501.4**	52.69	±18.3	7.19	±39.3	7.38	±7.64	3.74	±5.13
Progressive-X	–	674.2	20.60	**±15.3**	25.5	±113	8.05	±10.0	**3.14**	**±3.81**

Table 4: Fundamental Matrix and Homography Estimation: Average misclassification errors (ME, in %, lower is better), Sampson errors (SE, in pixel, lower is better), transfer errors (TE, in pixel, lower is better), and their standard deviations over five runs for fundamental matrix fitting on our new HOPE-F dataset and the Adelaide dataset, and for homography fitting on our new SMH dataset and the Adelaide dataset. Average run time per scene is given in milliseconds. We do not include time required for feature extraction: SIFT (Lowe 2004) requires 106.3 ms on average.

5 Experiments

For sampling and inlier weight prediction, we implement a neural network based on (Kluger et al. 2020b). Please refer to the supplementary for implementation details, a description of all evaluation metrics, and additional experimental results and discussions. We compute the results for all competitors using code provided by the respective authors, unless stated otherwise. For non-deterministic approaches, we report mean and standard deviation over five runs for all metrics. In addition, we measure average computation times with and without GPU acceleration, if applicable. We mark best results in **bold** and second best results with underline.

5.1 Vanishing Point Estimation

For vanishing point estimation, we use the cross product as our minimal solver with subsequent inlier weighted SVD refinement. We evaluate on four datasets: SU3, YUD, YUD+ and NYU-VP. The first two represent Manhattan-world scenarios, while the latter two contain non-Manhattan scenes. We compare against the robust multi-model fitting methods CONSAC (Kluger et al. 2020b), J- and T-Linkage (Toldo and Fusiello 2008; Magri and Fusiello 2014), Progressive-X (Barath and Matas 2019), and the task-specific VP estimators NeurVPS (Zhou et al. 2019a), DeepVP (Lin et al. 2022) and Contrario-VP (Simon, Fond, and Berger 2018). For J-/T-Linkage, we use the code provided by (Kluger et al. 2020b). We adopt the evaluation metric proposed by (Kluger et al. 2020b), i.e. we measure the angle between predicted and ground truth vanishing directions, and calculate the area under the recall curve (AUC) for multiple upper bounds.

Manhattan World For the Manhattan-world scenario, we train PARSAC on SU3 and evaluate on both SU3 and YUD. As Tab. 2 shows, PARSAC outperforms all methods but NeurVPS (Zhou et al. 2019a) and DeepVP (Lin et al. 2022) by large margins on SU3. Compared to DeepVP, our method is on par w.r.t. AUC @ 5° but more than ten percentage points better w.r.t. AUC @ 1°, i.e. we are able to estimate the vanishing points more accurately. NeurVPS performs substantially better than our method on SU3, but ranks second-to-last on YUD. This indicates that NeurVPS – also trained on SU3 – is not able to generalise well beyond its training domain. PARSAC, on the other hand, clearly outperforms all competitors on YUD. In addition, it is more than five times faster than the fastest competitor – Progressive-X (Barath and Matas 2019) – which achieves lower AUC values on both datasets. Our method is also more than 26 times faster than DeepVP, which is the second fastest competitor.

Non-Manhattan World For the non-Manhattan scenario, we train our method on NYU-VP for evaluation on the same, and train it on SU3 for evaluation on YUD+. As Tab. 2 shows, DeepVP outperforms PARSAC on NYU-VP, whereas PARSAC outperforms DeepVP on YUD+. While it is not obvious which method is more accurate overall, PARSAC has a clear advantage w.r.t. computation time, being 34 times faster. The only other competitive method, CONSAC, is almost three orders of magnitude slower. We did not evaluate NeurVPS, as it is designed for the Manhattan-world case only.

5.2 Fundamental Matrix Estimation

For fundamental matrix fitting, we use the seven point algorithm (Hartley and Zisserman 2003) as our minimal solver without subsequent refinement. We compare PARSAC with Fast-CP (Ozbay, Camps, and Sznaier 2022), Progressive-X+ (Barath et al. 2023) and Progressive-X (Barath and Matas 2019). CONSAC (Kluger et al. 2020b) has no implementation for F-matrix fitting available. We evaluate on our new HOPE-F dataset as well as on the Adelaide (Wong et al. 2011) dataset. In Tab. 4, we report misclassification errors (ME) and Sampson errors (SE). PARSAC outperforms all competitors on the HOPE-F dataset by large margins, achieving an average ME which is 9.6 percentage points lower than the best competitor (Fast-CP), with an SE which is 44% lower. It is more than twice as fast as Fast-CP and more than three times faster than Progressive-X+. On Adelaide, our method performs worse than Fast-CP and Progressive-X+, albeit with a smaller ME margin of 4.9 percentage points but a comparatively large SE.

5.3 Homography Estimation

For homography fitting, we use the four point DLT (Hartley and Zisserman 2003) as our minimal solver without subsequent refinement. We compare PARSAC with Progressive-X+ (Barath et al. 2023), Progressive-X (Barath and Matas 2019) and CONSAC (Kluger et al. 2020b). Fast-CP (Ozbay, Camps, and Sznaier 2022) has no implementation for homography fitting available. We evaluate on our new SMH dataset as well as on the Adelaide (Wong et al. 2011) dataset. In Tab. 4, we report misclassification errors (ME) and transfer errors (TE). PARSAC achieves an ME which is roughly on par with Progressive-X, although the latter has a slight edge on Adelaide. CONSAC and Progressive-X+ achieve lower MEs on Adelaide, but perform significantly worse on SMH. PARSAC has the lowest TE on SMH by a large margin, but falls behind its competitors on Adelaide. These baselines, however, are roughly between 10 and 100 times slower than PARSAC.

Limitations PARSAC requires a GPU to utilise its full potential. The number of putative model instances must be set before training, and instances beyond this cannot be found.

6 Conclusion

We proposed a new robust multi-model fitting algorithm which decouples the estimation of individual instances of a geometric model. PARSAC uses a neural network to predict multiple sets of sample and inlier weights in a single forward pass. This enables us to process all model instances in parallel and in real-time, yielding a significant speed-up compared to previous approaches. PARSAC can be used for a variety of computer vision problems, such as finding vanishing points, fundamental matrices and homographies, and achieves results superior to or competitive with state-of-the-art on multiple datasets. In addition, we contribute two new datasets for fundamental matrix and homography fitting, which will benefit the development of multi-model estimators in the future.

Acknowledgements

This work was supported by the Federal Ministry of Education and Research (BMBF), Germany under the AI Service Centre KISSKI (grant no. 01IS22093C), the Centre for Digital Innovations Lower Saxony (ZDIN) and the Deutsche Forschungsgemeinschaft (DFG) under Germany's Excellence Strategy within the Cluster of Excellence PhoenixD (EXC 2122).

References

Amayo, P.; Piniés, P.; Paz, L. M.; and Newman, P. 2018. Geometric multi-model fitting with a convex relaxation algorithm. In *CVPR*.

Antunes, M.; and Barreto, J. P. 2013. A global approach for the detection of vanishing points and mutually orthogonal vanishing directions. In *CVPR*.

Barath, D.; and Matas, J. 2018. Multi-class model fitting by energy minimization and mode-seeking. In *ECCV*.

Barath, D.; and Matas, J. 2019. Progressive-X: Efficient, Anytime, Multi-Model Fitting Algorithm. *ICCV*.

Barath, D.; Matas, J.; and Hajder, L. 2016. Multi-H: Efficient recovery of tangent planes in stereo images. In *BMVC*.

Barath, D.; Rozumnyi, D.; Eichhardt, I.; Hajder, L.; and Matas, J. 2023. Finding Geometric Models by Clustering in the Consensus Space. In *CVPR*.

Barinova, O.; Lempitsky, V.; Tretiak, E.; and Kohli, P. 2010. Geometric image parsing in man-made environments. In *ECCV*.

Brachmann, E.; Michel, F.; Krull, A.; Ying Yang, M.; Gumhold, S.; and Rother, C. 2016. Uncertainty-Driven 6D Pose Estimation of Objects and Scenes From a Single RGB Image. In *CVPR*.

Brachmann, E.; and Rother, C. 2018. Learning Less is More-6D Camera Localization via 3D Surface Regression. In *CVPR*.

Brachmann, E.; and Rother, C. 2019. Neural-Guided RANSAC: Learning Where to Sample Model Hypotheses. In *ICCV*.

Chin, T.-J.; Wang, H.; and Suter, D. 2009. Robust fitting of multiple structures: The statistical learning approach. In *ICCV*.

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*.

Denis, P.; Elder, J. H.; and Estrada, F. J. 2008. Efficient edge-based methods for estimating manhattan frames in urban imagery. In *ECCV*.

Farina, M.; Magri, L.; Menapace, W.; Ricci, E.; Golyanik, V.; and Arrigoni, F. 2023. Quantum Multi-Model Fitting. In *CVPR*.

Fischler, M. A.; and Bolles, R. C. 1981. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*.

Hartley, R.; and Zisserman, A. 2003. *Multiple view geometry in computer vision*. Cambridge university press.

Isack, H.; and Boykov, Y. 2012. Energy-based geometric multi-model fitting. *IJCV*.

Kluger, F.; Ackermann, H.; Brachmann, E.; Yang, M. Y.; and Rosenhahn, B. 2021. Cuboids revisited: Learning robust 3d shape fitting to single rgb images. In *CVPR*.

Kluger, F.; Ackermann, H.; Yang, M. Y.; and Rosenhahn, B. 2017. Deep learning for vanishing point detection using an inverse gnomonic projection. In *GCPR*.

Kluger, F.; Ackermann, H.; Yang, M. Y.; and Rosenhahn, B. 2020a. Temporally Consistent Horizon Lines. In *ICRA*.

Kluger, F.; Brachmann, E.; Ackermann, H.; Rother, C.; Yang, M. Y.; and Rosenhahn, B. 2020b. Consac: Robust multi-model fitting by conditional sample consensus. In *CVPR*.

Kuhn, H. W. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly*.

Lezama, J.; Grompone von Gioi, R.; Randall, G.; and Morel, J.-M. 2014. Finding vanishing points via point alignments in image primal and dual domains. In *CVPR*.

Lin, Y.; Wiersma, R.; Pintea, S. L.; Hildebrandt, K.; Eisemann, E.; and van Gemert, J. C. 2022. Deep vanishing point detection: Geometric priors make dataset variations vanish. In *CVPR*.

Liu, S.; Zhou, Y.; and Zhao, Y. 2021. Vapid: A rapid vanishing point detector via learned optimizers. In *ICCV*.

Lowe, D. G. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*.

Magri, L.; and Fusiello, A. 2014. T-linkage: A continuous relaxation of j-linkage for multi-model fitting. In *CVPR*.

Magri, L.; and Fusiello, A. 2015. Robust Multiple Model Fitting with Preference Analysis and Low-rank Approximation. In *BMVC*.

Magri, L.; and Fusiello, A. 2016. Multiple model fitting as a set coverage problem. In *CVPR*.

Magri, L.; and Fusiello, A. 2019. Fitting Multiple Heterogeneous Models by Multi-Class Cascaded T-Linkage. In *CVPR*.

Mur-Artal, R.; and Tardós, J. D. 2017. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *T-RO*.

Ozbay, B.; Camps, O.; and Sznaier, M. 2022. Fast Two-View Motion Segmentation Using Christoffel Polynomials. In *ECCV*.

Pham, T. T.; Chin, T.-J.; Schindler, K.; and Suter, D. 2014. Interacting geometric priors for robust multimodel fitting. *Transactions on Image Processing*.

Rosenhahn, B. 2023. Optimization of Sparsity-Constrained Neural Networks as a Mixed Integer Linear Program. *Journal of Optimization Theory and Applications*.

Schulman, J.; Heess, N.; Weber, T.; and Abbeel, P. 2015. Gradient estimation using stochastic computation graphs. *NeurIPS*.

Silberman, N.; Hoiem, D.; Kohli, P.; and Fergus, R. 2012. Indoor Segmentation and Support Inference from RGBD Images. In *ECCV*.

Simon, G.; Fond, A.; and Berger, M.-O. 2018. A-Contrario Horizon-First Vanishing Point Detection Using Second-Order Grouping Laws. In *ECCV*.

Tardif, J.-P. 2009. Non-iterative approach for fast and accurate vanishing point detection. In *ICCV*.

Toldo, R.; and Fusiello, A. 2008. Robust multiple structures estimation with j-linkage. In *ECCV*.

Tyree, S.; Tremblay, J.; To, T.; Cheng, J.; Mosier, T.; Smith, J.; and Birchfield, S. 2022. 6-DoF Pose Estimation of Household Objects for Robotic Manipulation: An Accessible Dataset and Benchmark. In *IROS*.

Vedaldi, A.; and Zisserman, A. 2012. Self-similar sketch. In *ECCV*.

Vincent, E.; and Laganiére, R. 2001. Detecting planar homographies in an image pair. In *ISPA*.

Von Gioi, R. G.; Jakubowicz, J.; Morel, J.-M.; and Randall, G. 2008. LSD: A fast line segment detector with a false detection control. *TPAMI*.

Wildenauer, H.; and Hanbury, A. 2012. Robust camera self-calibration from monocular images of Manhattan worlds. In *CVPR*.

Wong, H. S.; Chin, T.-J.; Yu, J.; and Suter, D. 2011. Dynamic and hierarchical multi-structure geometric model fitting. In *ICCV*.

Wu, J.; Zhang, L.; Liu, Y.; and Chen, K. 2021. Real-time vanishing point detector integrating under-parameterized ransac and hough transform. In *ICCV*.

Xu, Y.; Oh, S.; and Hoogs, A. 2013. A minimum error vanishing point detection approach for uncalibrated monocular images of man-made environments. In *CVPR*.

Zhai, M.; Workman, S.; and Jacobs, N. 2016. Detecting vanishing points using global image context in a non-manhattan world. In *CVPR*.

Zhou, Y.; Qi, H.; Huang, J.; and Ma, Y. 2019a. Neurvps: Neural vanishing point scanning via conic convolution. *NeurIPS*.

Zhou, Y.; Qi, H.; Zhai, Y.; Sun, Q.; Chen, Z.; Wei, L.-Y.; and Ma, Y. 2019b. Learning to reconstruct 3d manhattan wireframes from a single image. In *ICCV*.

Zou, C.; Colburn, A.; Shan, Q.; and Hoiem, D. 2018. Layoutnet: Reconstructing the 3d room layout from a single rgb image. In *CVPR*.