Universal Framework for Random Sample Consensus

Maksym Ivashechkin, Dániel Béla Baráth, Jiří Matas

Centre for Machine Perception, Dept. of Cybernetics, CTU Prague, Karlovo Naměstí 13, CZ 121 35
[ivashmak, baratdan, matas] @cmp.felk.cvut.cz

Abstract

After 6 years of the first publication of USAC [1] the original RANSAC algorithm by Fischler et. al. [2] is still popular and widely using. New advanced methods of sampling or local optimization were published. So we decided to make new version of USAC as most accurate, efficient and simple implementation that includes all best derivations of RANSAC.

1 Introduction

One of the reasons why RANSAC became so famous is its simplicity. The algorithm can be divided on several main parts. First is sampling procedure. Second one is model evaluation. And the last one is saving so far the best model and computing upper bound of iterations.
Now RANSAC has a lot improvements. For example many different Local Optimization parts or bunch of sampling methods. We will show the most interesting and popular ones and discuss their influence.

2 Solvers

2.1 Line in 2D (A)

Line in 2 dimensional subspace is the easiest and very popular estimation in many areas. The optimal model must cover with predefined threshold the biggest number of points. In non trivial cases (many outliers) Ransac is very robust and efficient algorithm.

2.2 Homography matrix (H)

Homography matrix estimation [10] is very popular in computer vision. And it also using in many image processing areas. The task is to find matrix of size 3×3 multiplying it by points of first correspondence and normalizing by third coordinate give us points on second correspondent image, similarly multiplying inverse H by points on second correspondence produces points on first correspondence.
For minimal sample estimation we use Direct Linear Transformation (DLT, 4-points algorithm). For non minimal estimation - DLT with normalizing transformation[11].

We applied Homography matrix estimation problem on HOMOGR dataset by K. Lebeda and EVD [14] dataset by D. Mishkin. The results are in Table 3.

2.3 Fundamental matrix (F)

The idea of fundamental matrix estimation is to find matrix 3×3 that represents epipolar lines on two correspondent images, which with predefined threshold covers the biggest number of points.
This matrix can be found by 7-points algorithm [10] (minimal sample). And non minimal model estimation - 8-points algorithm following normalizing transformation.

We applied F estimation on KUSVOD2 dataset by K. Lebeda and ADELAIDERMF dataset by H. Wong. The results of fundamental matrix estimation are in Table 3.

2.4 Essential matrix (E)

Essential matrix is similar to Fundamental matrix, but points correspond to the same 3D point in the scene.
For minimal model estimation we are using 5-points algorithm [12]. For non minimal model as for Fundamental matrix we use 8-points algorithm with normalizing transformation.

We applied Essential matrix fitting on Multi View dataset by C. Strecha [15]. The results of essential matrix estimation are in Table 3.

2.5 LSQ vs SVD vs PCA

We tried 3 similar approaches for $H - F$ estimation:

1. The simplest one is Least Squares (LSQ). We need to construct matrices A which represents desired linear mapping and b of desired values and then apply pseudo inverse $A^{+}b$. This is minimizing perpendicular distances among estimated and given values. Obviously this way is numerically and computationally very bad.
2. Principal Component Analysis (PCA) is much better than LSQ, one of the reasons is minimizing distances to the optimal subspace corresponding to the highest eigen value. The same problem is computing covariance matrix

by inner products of linear mapping which is inaccurate enough.
3. Singular Value Decomposition (SVD) is very popular algorithm which is used instead of PCA or spectral decomposition. Turns out it is the best way for model estimation with respect to error, though a bit slower than PCA,. It is not surprising, because it requires to construct matrix $N \times 9$ whereas for PCA the covariance matrix is 9×9.

We compare PCA and SVD by error and computational time in Table 1. So for most pairs of images PCA superiors SVD by time but losing in error.

Table 1 Comparing non minimal model estimation for $H-F$ problems with SVD and Eigen (with covariance matrix) calculation. 200 random runs for each images for "homogr" dataset. 100 random runs for "EVD" dataset. 200 random runs for each image for "adelaidermf" dataset. $\mathcal{E}$ is mean of averages error. $\mathcal{T}$ is mean of averages time.

			Confidence 95%	
			GC+SVD	GC+PCA
homogr, H	№1	$\mathcal{E}$	**.499** ± .003	.501 ± .003
		$\mathcal{T}$	.593 ± .114	**.407** ± .064
	№2	$\mathcal{E}$	**.431** ± .204	.569 ± .187
		$\mathcal{T}$	.633 ± .167	**.367** ± .085
	№3	$\mathcal{E}$	**.402** ± 1.00	.598 ± .900
		$\mathcal{T}$	.503 ± .166	**.497** ± .134
Total number of images in dataset is 12. PCA is faster at 12 and more accurate in 4 images.				
EVD, H	№1	$\mathcal{E}$	**.493** ± .051	.507 ± .059
		$\mathcal{T}$	.503 ± .048	**.497** ± .081
	№2	$\mathcal{E}$	**.413** ± .832	.587 ± 2.39
		$\mathcal{T}$	**.478** ± .003	.522 ± .045
	№3	$\mathcal{E}$	.506 ± .039	**.494** ± .038
		$\mathcal{T}$	.735 ± .516	**.265** ± .046
Total number of images in dataset is 15. PCA is faster at 8 and more accurate in 7 images.				
adelaidermf, F	№1	$\mathcal{E}$	**.467** ± .213	.533 ± .272
		$\mathcal{T}$	.500 ± .144	**.500** ± .092
	№2	$\mathcal{E}$	.642 ± 1.24	**.358** ± 1.98
		$\mathcal{T}$	.540 ± .166	**.460** ± .202
	№3	$\mathcal{E}$	.526 ± .021	**.474** ± .017
		$\mathcal{T}$	**.480** ± .104	.520 ± .115
Total number of images in dataset is 15. PCA is faster at 14 and more accurate in 6 images.				

3 Sampling

3.1 Uniform

For uniform sampling we have the modern version of Fisher–Yates shuffle. We need to allocate an array of size number of points N and fill it from 0 to N-1 and define variable $max = N$ which represents maximum random range. Take at random index of array from range $[0; max)$, return value of array by this random index, decrease max by 1 and swap value of array at random index with value of array at index max. This algorithms produces unbiased permutation with 100% uniform distribution. The main advantage is that all values are unique. Cons of this method is expensive memory allocation for big number of points.

3.2 Progressive Sample Consensus (PROSAC)

Progressive Sample Consensus described in [3] by Chum et. al. is the most efficient sampling based on known quality function for every point. Every sample is taken from the top-ranked correspondences (in descending order).

In our experiments we used quality function as was mentioned in Prosac paper to ratio of the distances in the SIFT space of the best and second match with threshold 0.8 recommended in [13]. Also it is possible to use density sort defined by sum of distances of the nearest neighbors. This simple way, which does not require any extra information about points, works well for $A-H-F$ estimation. Although this quality sort could be completely wrong for unexpected distribution of points.

3.3 N adjacent points Sample Consensus (NAPSAC)

NAPSAC [4] is one of the well-known sampling procedure based on simple observation that inliers are usually very closed to each other. The first step is choosing initial point from all points uniformly at random and take required minimal sample size under hyper-sphere centered at chosen point of some predefined radius r. If there are not enough points then take another initial point. Thus the probability of being inlier within d-dimensional manifold in D-dimensional subspace is αr^d and for outlier is βr^D.

3.4 Progressive NAPSAC

One of the biggest advantage of NAPSAC is good inlier classification in high dimension. In the [4] was mentioned:

"... required iterations (of RANSAC) increases exponentially with dimensionality." And according to results [4] NAPSAC for dimension starting from 4-5 is significantly better than RANSAC.

But for our practical problem of $A-H-F-E$ estimation we do not need so big dimension. Or what if the distribution of points is not normal (Gaussian) and points are not closed to each other? We would like introduce the new method of sampling that similar to NAPSAC, but can handle those problems.

Algorithm 1 Progressive Napsac.

```
1: Define initial radius $r_k, \quad k = 0$.
2: Choose initial point uniformly at random.
3: If there are not enough points under hyper-sphere then
   choose another initial point.
4: Take minimal sample size under hyper-sphere of current
   radius $r_k$.
5: Gradually increase current radius (following predefined
   conditions) $r_{k+1} = r_k + \epsilon, \quad \epsilon > 0$.
6: Repeat step 1-5.
```

It is clear that Progressive NAPSAC will converges to RANSAC-like sampling, because for bigger k the radius r_k will increase. This method is generalization of NAPSAC. For example if the problem is solvable by NAPSAC then Progressive NAPSAC can solve it too, because it starts like NAPSAC.

3.5 PROSAC with NAPSAC

We are also considering the mix of PROSAC and NAPSAC as interesting combination of two useful sampling. The proposed algorithm is:

Algorithm 2 Prosac with Napsac.

```
1: Select a point by PROSAC.
2: Select the rest of the required points from the selected
   points's neighborhood with PROSAC using the score
   of the neighbours.
3: If the neighborhood is not big enough for minimal sam-
   ple size then select a new initial point by PROSAC
   (step (1)).
4: Repeat step 1-3.
```

4 Local Optimization

4.1 Graph Cut

The most important local optimization in our implementation is Graph Cut Ransac [7]. Formulated as energy minimization this local optimization superiors many other optimizations by error and time. According to our results it is the most accurate in Table 3 for every dataset and has the smallest average error.

4.2 Locally Optimized Ransac

Locally Optimized Ransac [5] is reliable and simple defined local optimization. We can divide it by inner and iterative parts. In the inner LO we take non minimal sample size from inliers of so far the best model and estimate non minimal model. The iterative part estimate non minimal model of all inliers iteratively decreasing threshold.

4.3 Fixing Locally Optimized Ransac

This method was presented by Lebeda et. al. [8] as improving Locally Optimized Ransac. The important thing was noticed, citation:
"For a large number of inliers, this procedure (author means Iterative Ransac) can dominate the execution time, even if executed only once." (Lebeda et. al.)

Our experiments also prove it. Sampling of some not big (Lebeda used $7 * minimal\ sample\ size$) number of inliers in Iterative Ransac sped up computational time several times, but on another hand the accuracy a little decreased, which is understandable, because in iterative part estimation based not on all inliers.

5 Model Evaluation

5.1 Score

The simplest way to evaluate model is define the binary quality function, e.g:

$$score\ (X, Model) = \sum_{\mathbf{x} \in X} f(\mathbf{x},\ Model)$$

$$f(\mathbf{x}) = \begin{cases} 1, & d(\mathbf{x},\ Model) < threshold \\ 0, & otherwise \end{cases}$$

Where $d(\mathbf{x},\ Model)$ represents error (distance) function of point $\mathbf{x}$ to evaluating $Model$. For our solvers it is comparing sum of inliers of the best model and current model.

The better way of model evaluation was shown in MLESAC [9] by Torr et. al.

5.1.1 Error function

1. For line in 2d the error is distance from point to line.
2. For Homography matrix estimation error is average of distances from point on 1 correspondence to estimated point on 2 correspondence and from point on 2 correspondence to estimated point on 1 correspondence.
3. For Fundamental matrix estimation we used Sampson distance.
4. For Essential matrix estimation it is average of distances from point to epipolar line on 1 correspondence and from point on 2 correspondence to same epipolar line.

5.2 Sequential probability ratio test

Another powerful method for model evaluation is its verification with Wald's sequential probability ratio test (SPRT) described in Randomized Ransac [6] by Chum et. al. The idea is that we do not need to calculate errors for all points, but using decision threshold we can predict if model is good or bad.

Bad thing about SPRT is choosing initial ϵ_0 (probability of any randomly chosen data point is consistent with a 'good' model) and δ_0 (probability of a data point being consistent with a 'bad' model). We can not obtain these probabilities for completely unknown tasks and even lower bound could not make the right prediction. The example when SPRT can fail is two correspondent images with very small inlier ratio, so after applying SPRT every model will be bad, because any point chosen at random is outlier with high probability.
One of the suggested method in [6] is obtaining δ_0 by geometric considerations, i.e. as a fraction of the area that supports a hypothesised model.
In our implementation we start skipping bad models after some predefined limit of iterations, so even if SPRT decides that model is bad we still finish its evaluation to get full score. Thus even initial prediction of δ and ϵ was wrong it will stabilize after those iterations.

6 Termination criteria

6.1 Uniform sampling

As soon as USAC can have Prosac or Uniform sampler with SPR test or not we can combine termination conditions.
For uniform independent sampling Fischler et. el. derived upper bound iteration:

$1 - p = (1 - w^n)^k$

$k = log_{1-w^n}(1 - p)$

$k = \frac{ln\ 1-p}{ln\ 1-w^n}$

Which means that in k step probability of getting at least one outlier is p (in our experiments set to 1% and 5%).

6.2 Prosac sampling

For sorted points by quality function $q(\mathbf{x})$ there are two conditions:
1. *maximality* the probability that I_{n*}(number of inliers) out of $n*$ (stopping length) data points are by chance inliers to an arbitrary incorrect model is smaller than 5% ([3]).
2. $non - randomness$ the probability that a solution with more than I_{n*} inliers in U_{n*} (a set of $n*$ data points with the highest quality) exists and was not found after k samples is smaller than 5% ([3]).

6.3 SPRT

Termination conditions in [6] says *"If the probability of missing a set of inliers larger than the largest support found so far falls under a predefined threshold 5% then algorithm ends"*. Assumption for this condition is that any model evaluation is independent and any point was chosen uniformly at random for every test.

In our experiments we found that it is sufficient to shuffle all indexes of points in the beginning and for every new test start model verification with point where previous test ended. It is much faster than take at random point for every SPRT iteration.

So in USAC we can mix Fischler's termination conditions with SPRT or Prosac termination criteria with SPRT, because each data point for model evaluation is choosing at random for sorted or uniform sampling. However there is no need to combine SPRT with Prosac, because Prosac shown to be much more efficient and termination conditions for Prosac is more suitable.

7 Neighbors Searching

For Napsac sampler and Graph-Cut local optimization it is quite important to find 'good' neighbors for better estimation. We tried next methods:

7.1 K Nearest Neighbors with Nanoflann

K nearest neighbors (KNN) using Nanoflann by Jose Luis Blanco-Claraco et. al. is more efficient than Flann KNN search. But KNN search has one big disadvantage: if point does not have adjacent points under some hyper sphere than KNN search returns K far neighbors of this point and it has negative influence on GC-Ransac.

7.2 Grid Searching

We would like represent Grid Nearest Neighbors search by defining next rule:
Assume we have 2 correspondent images for H-F estimation. We can uniformly divide 2 images by cells and get grid. *Points are neighbors if and only if they are in the same cell in both correspondent images.*

This implementation has almost linear complexity $O(n)$ as we put n points to cells with Hash table and after checking those cells. In this case point does not necessary have neighbors.

Our experiments in table 2 shows that GC-Ransac with Grid searching is faster and more accurate than KNN.
The best cell size appeared to be 50 pixels after exhaustive experimentation. Because with smaller cell size points have not enough neighbors, on another hand with bigger cell size points have more neighbors which are too far.

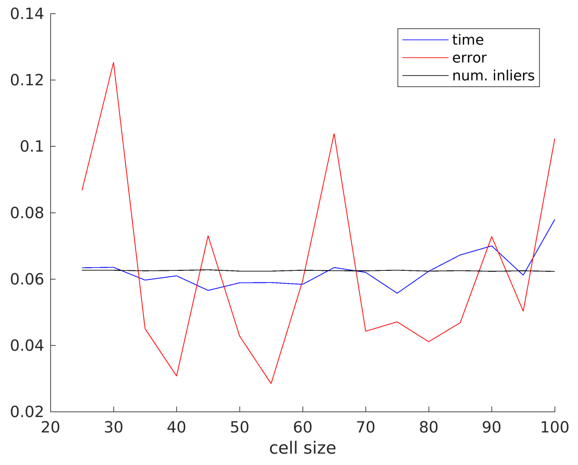

Figure 1: Dependence of error, time and num. of inliers to cell size (sum to 1). Tested with uniform sampling and Graph Cut LO. First graph is Homography matrix fitting from Lebeda "Homogr" dataset (12 images were chosen), every image have run 200 times and mean of averages was calculated.

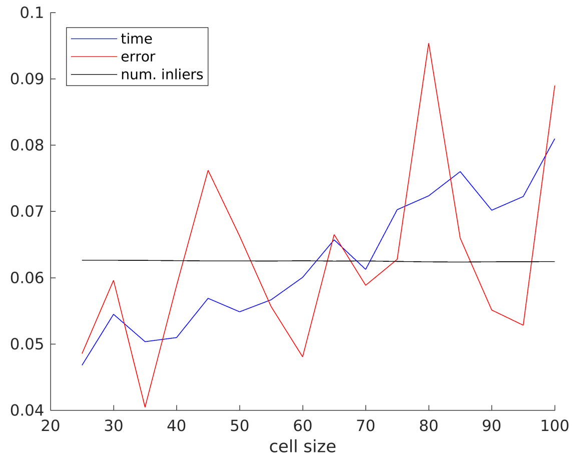

Figure 2: Analogically F matrix fitting was tested from "Adelaidermf" dataset (15 images were chosen). Every image was run 100 times.

8 Implementation

One of the biggest advantages of our proposal is that we can present simple (in understanding sense) code written in C++. We do not use any extra libraries only well-known OpenCV and OpenMP. Code is compatible with C++ 11 version. The source can be easily derived to any project and extended by any new local optimization or sampling method. Every component such as solver or sampler has abstract definition and every sub component is derived and has own class. Also our code is able to have parallel evaluation which is much faster for big number of points.

Table 2 Comparing GC with KNN and Grid Searching. There was 5 neighbors for H and 8 neighbors for F. 200 random runs for each image in adelaidermf and homogr dataset, 500 runs in kusvod2 and 100 runs in EVD dataset. $\mathcal{E}_{wc}$ is mean of worst case errors.

			Confidence 95%	
			GC+KNN	GC+Grid
kusvod2	F	$\mathcal{E}$	.558	**.442**
		$\mathcal{E}_{wc}$	.557	**.443**
		$\mathcal{T}$	.509	**.491**
Grid is better than KNN in 11/16 cases by $\mathcal{E}$				
Grid is better than KNN in 11/16 cases by $\mathcal{E}_{wc}$				
Grid is better than KNN in 9/16 cases by $\mathcal{T}$				
adelaidermf	F	$\mathcal{E}$	.506	**.494**
		$\mathcal{E}_{wc}$	.516	**.484**
		$\mathcal{T}$	**.499**	.501
Grid is better than KNN in 7/15 cases by $\mathcal{E}$				
Grid is better than KNN in 8/15 cases by $\mathcal{E}_{wc}$				
Grid is better than KNN in 10/15 cases by $\mathcal{T}$				
homogr	H	$\mathcal{E}$	.542	**.458**
		$\mathcal{E}_{wc}$	.530	**.470**
		$\mathcal{T}$	**.488**	.512
Grid is better than KNN in 8/12 cases by $\mathcal{E}$				
Grid is better than KNN in 10/12 cases by $\mathcal{E}_{wc}$				
Grid is better than KNN in 7/12 cases by $\mathcal{T}$				
EVD	H	$\mathcal{E}$	**.482**	.518
		$\mathcal{E}_{wc}$	**.438**	.562
		$\mathcal{T}$	.502	**.498**
Grid is better than KNN in 9/15 cases by $\mathcal{E}$				
Grid is better than KNN in 9/15 cases by $\mathcal{E}_{wc}$				
Grid is better than KNN in 8/15 cases by $\mathcal{T}$				

9 Experiments

We tested RANSAC, PROSAC, Locally Optimized RANSAC, Fixing Locally Optimized RANSAC, RANSAC with SPRT, Graph-Cut RANSAC and Graph-Cut RANSAC with SPRT on KUSVOD2, ADELAIDERMF, HOMOGR and EVD datasets in Table 3.
Clearly the best algorithm by error is Graph-Cut RANSAC. Usually the most efficient is PROSAC and it is reasonable. On the second place is again Graph-Cut and Fixing Locally Optimized RANSAC.
The smallest worst case error has Graph-Cut RANSAC and Locally Optimized RANSAC.
By results seems that PROSAC is not reliable, but it is not completely true, because PROSAC due to its complex termination criteria sometimes returns very quickly bad solution.
EVD dataset is quite tough for SPRT, because there are a lot of outliers in the images.

Table 3 RSC means Ransac with Uniform Sampler. PSC means Prosac. LO is the number of local optimization iterations. $\mathcal{E}$ is the geometric error of the estimated model w.r.t. the manually selected inliers. Similarly $\mathcal{E}_{wc}$ is the worst case error. $\mathcal{T}$ is the mean processing time. $\mathcal{S}$ is the required number of samples (iterations). $\mathcal{F}$ is number of failed estimation (estimation was failed if there more than 50% of ground truth inliers that are not lying under threshold). All values in each row are sum to 1. Each image have run 100 times and mean of all averages was calculated. # means number of images that were tested within dataset.

			Confidence 95%						
			RSC	PSC	LO	FLO	SPRT	GC	GC+SPRT
kusvod2 (#16)	F	LO	.000	.000	.200	.213	.000	.628	.558
		$\mathcal{E}$	.136	.094	.173	.153	.153	**.110**	.182
		$\mathcal{E}_{wc}$	**.089**	.211	.140	.143	.105	.130	.182
		$\mathcal{T}$	.137	**.093**	.171	.151	.125	.157	.167
		$\mathcal{S}$	.168	**.115**	.134	.139	.142	.151	.151
		$\mathcal{F}$	**.105**	.211	.143	.137	.138	.123	.143
adelaideRMF (#15)	F	LO	.000	.000	.116	.116	.000	.412	.357
		$\mathcal{E}$	.343	.250	.089	.086	.087	**.071**	.075
		$\mathcal{E}_{wc}$	.260	.591	.029	.031	**.027**	.029	.033
		$\mathcal{T}$	.077	**.026**	.276	.136	.071	.216	.196
		$\mathcal{S}$	.299	**.030**	.037	.035	.031	.106	.184
		$\mathcal{F}$	0.400	0.200	**.000**	**.000**	.300	**.000**	.100
homogr (#12)	H	LO	.000	.000	.121	.000	.121	.424	.333
		$\mathcal{E}$	.153	.180	**.122**	.133	.156	.133	.123
		$\mathcal{E}_{wc}$	.132	.297	**.089**	.119	.138	.117	.110
		$\mathcal{T}$	.431	.084	.149	.077	.115	**.072**	.074
		$\mathcal{S}$	.439	.050	.019	.017	.428	**.011**	.038
		$\mathcal{F}$	**.000**	.544	**.000**	.018	.380	.018	.041
EVD (#15)	H	LO	.000	.000	.147	.082	.000	.606	.166
		$\mathcal{E}$	.070	.071	.093	.106	.231	**.051**	.378
		$\mathcal{E}_{wc}$	.068	.301	.092	.217	.091	**.044**	.188
		$\mathcal{T}$	.195	.148	.191	.188	.047	.190	**.041**
		$\mathcal{S}$	.157	**.114**	.147	.141	.158	.144	.141
		$\mathcal{F}$	.165	.151	.139	.135	.160	**.113**	.137
strecha	E	LO	.000	.000	-	-	.000	-	-
		$\mathcal{E}$	-	-	-	-	-	-	-
		$\mathcal{E}_{wc}$	-	-	-	-	-	-	-
		$\mathcal{T}$	-	-	-	-	-	-	-
		$\mathcal{S}$	-	-	-	-	-	-	-
		$\mathcal{F}$	-	-	-	-	-	-	-

10 Conclusions

So for now the most reliable and efficient is Graph-Cut RANSAC (possibly with SPRT).

References

[1] R. Raguram, O. Chum, M. Pollefeys, J. Matas, and J.-M. Frahm. USAC: a universal framework for random sample consensus. Transactions on Pattern Analysis and Machine Intelligence, 2013. 2, 9

[2] M. A. Fischler and R. C. Bolles, ”Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” Communications of the ACM, vol. 24, no. 6, pp. 381–395, 1981

[3] O. Chum and J. Matas, ”Matching with PROSAC - progressive sample consensus,” in Computer Vision and Pattern Recognition (CVPR), 2005.

[4] D. R. Myatt, P. H. S. Torr, S. J. Nasuto, J. M. Bishop, and R. Craddock, ”NAPSAC: high noise, high dimensional robust estimation,” in British Machine Vision Conference (BMVC), 2002, pp. 458–467

[5] O. Chum, J. Matas, and J. Kittler. Locally optimized ransac. In Joint Pattern Recognition Symposium. Springer, 2003. 1, 2, 4

[6] J. Matas and O. Chum, ”Randomized RANSAC with sequential probability ratio test,” in International Conference on Computer Vision (ICCV), vol. 2, Oct. 2005, pp. 1727–1732 Vol. 2

[7] Graph-Cut Ransac, D. Baráth, J. Matas, 2018 IEEE CVF Conference on Computer Vision and Pattern Recognition, 6733-6741

[8] K. Lebeda, J. Matas, and O. Chum. Fixing the locally optimized ransac. In BMVC. Citeseer, 2012. 2, 3, 4

[9] P. H. S. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. Computer Vision and Image Understanding, 2000. 1, 3

[10] Multiple View Geometry in Computer Vision Second Edition, R. Hartley and A. Zisserman, Cambridge University Press, March 2004.

[11] Homography Estimation, E. Dubrofsky

[12] D. Nister, "An Efficient Solution to the Five-Point Relative Pose Problem," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 26, no. 6, pp. 756-777, June 2004.

[13] D. Lowe. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2):91–110, 2004

[14] D. Mishkin, J. Matas, and M. Perdoch. MODS: Fast and robust method for two-view matching. Computer Vision and Image Understanding, 2015. 1

[15] C. Strecha, R. Fransens, and L. Van Gool. Wide-baseline stereo from multiple views: a probabilistic account. In Conference on Computer Vision and Pattern Recognition. IEEE, 2004. 6