



CENTER FOR
MACHINE PERCEPTION



CZECH TECHNICAL
UNIVERSITY IN PRAGUE

BACHELOR THESIS

USAC++ Improved Universal Framework for Random Sample Consensus

Maksym Ivashechkin

ivashmak@fel.cvut.cz

May 24, 2019

Thesis Advisor: prof. Ing. Jiří Matas Ph.D.

Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27 Prague 6, Czech Republic
fax +420 2 2435 7385, phone +420 2 2435 7637, www: <http://cmp.felk.cvut.cz>

I. Personal and study details

Student's name: **Ivashechkin Maksym** Personal ID number: **465693**
Faculty / Institute: **Faculty of Electrical Engineering**
Department / Institute: **Department of Cybernetics**
Study program: **Open Informatics**
Branch of study: **Computer and Information Science**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Improved USAC

Bachelor's thesis title in Czech:

USAC++

Guidelines:

USAC [1] was, at the time of its publication, a state-of-the-art implementation of RANSAC, a popular robust method for estimation of geometrical models. Recently, a new method of local optimization [2] has been published. Also, a potentially effective method for sample selection – NAPSAC [3] – has not been implemented and tested within USAC framework. The objective of the thesis is to obtain an improved version of USAC.

Therefore:

1. Review with the USAC method, download and test the code.
2. Review the available datasets with GT for two-view matching evaluation, e.g. the Oxford Mikolajczyk set [4], Lebeda dataset [5], EVD [6] and others.
3. Implement [2] and [3] in USAC-like framework.
4. (Optionally) propose and implement other improvements.
5. Evaluate the resulting USAC++ in terms of speed and accuracy.

Bibliography / sources:

- [1] R. Raguram, O. Chum, M. Pollefeys, J. Matas, and J-M. Frahm, - USAC: a Universal Framework for Random Sample Consensus - IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013.
- [2] D. Barath, J. Matas - Graph-Cut RANSAC - Computer Vision and Pattern Recognition (CVPR), 2018.
- [3] Darren R. Myatt, Philip H. S. Torr, Slawomir J. Nasuto, J. Mark Bishop, R. Craddock - Napsac: High noise, high dimensional robust estimation - British Machine Vision Conference (BMVC), 2002.
- [4] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, L. J. Van Gool - A Comparison of Affine Region Detectors - International Journal of Computer Vision, 2005.
- [5] K. Lebeda, J. Matas and O. Chum - Fixing the Locally Optimized RANSAC - BMVC, 2012.
- [6] D. Mishkin, J. Matas, M. Perdoch - MODS: Fast and robust method for two-view matching - Computer Vision and Image Understanding, 2015.
- [7] O. Chum, J. Matas - Matching with PROSAC - Progressive Sample Consensus. CVPR, 2005.

Name and workplace of bachelor's thesis supervisor:

prof. Ing. Jiří Matas, Ph.D., Visual Recognition Group, FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **02.01.2019** Deadline for bachelor thesis submission: **24.05.2019**

Assignment valid until: **30.09.2020**

prof. Ing. Jiří Matas, Ph.D.
Supervisor's signature

doc. Ing. Tomáš Svoboda, Ph.D.
Head of department's signature

prof. Ing. Pavel Ripka, CSc.
Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt

Student's signature

Author statement for undergraduate thesis

I declare that the presented work developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date

.....
signature

Acknowledgement

I would like to thank my supervisor Jiří Matas for his professional guidance and help throughout my thesis.

My sincere thanks also belong to Dániel Baráth for his big help in the implementation. I am greatly indebted to Nikolay Chumerin for his helpful advice and suggestions.

In additions, I am deeply grateful to my parents, relatives and friends who supported me at every bit and without whom it was impossible to finish my project.

Abstract

*Robust estimation is an important open problem, which has applications in many areas. One of the most popular robust estimation algorithm is **RAN**dom **SA**mple **C**onsensus, that is able to efficiently provide good estimation using even fairly contaminated data. Robust estimation of parametric models has been improving in many respects over the last decades. The proposed in this study universal framework **USAC++** for random sample and consensus includes novel superior (so far) methods for robust estimation. The most used **RANSAC**-based algorithms are reviewed, discussed and implemented in a more accurate, faster and simple way. The framework is written in **C++** and was tested on multiple view geometry problems and evaluated on different publicly available datasets.*

Keywords: RANSAC, robust estimation, local optimization, homography, epipolar geometry.

Abstrakt

*Robustní odhad je důležitý otevřený problém, který má aplikace ve mnoha oblastech. Jedním z nejpoužívanějších robustních algoritmů pro odhad je **RAN**dom **SA**mple **C**onsensus, který je schopen efektivně poskytnout dobrý odhad s využitím i poměrně kontaminovaných dat. V posledních desetiletích se ve mnoha ohledech zlepšil robustní odhad parametrických modelů. Navržený univerzální rámec **USAC++** pro náhodný výběr a konsenzus obsahuje v této studii (zatím) lepší metody pro robustní odhad. Nejpoužívanější algoritmy založené na **RANSACu** jsou přezkoumány, diskutovány a implementovány přesnějším, rychlejším a jednodušším způsobem. Rámec je napsán v jazyce **C++** a byl otestován na úlohách vícehledové geometrie a vyhodnocován na různých veřejně dostupných datových sádkách.*

Klíčová slova: RANSAC, robustní odhad, lokální optimalizace, homografie, epipolární geometrie.

Contents

1	Introduction	1
2	Background	3
2.1	Design	3
2.2	Two-View Geometry	4
3	Estimation problems	7
3.1	Line in 2D	7
3.2	Homography matrix (H)	7
3.3	Fundamental matrix (F)	8
3.4	Essential matrix (E)	8
4	Sampling	9
4.1	Random Sample Consensus	9
4.2	Progressive Sample Consensus	9
4.3	N adjacent points Sample Consensus	10
4.4	Progressive NAPSAC	11
4.5	Groups Sample Consensus	11
5	Model Evaluation	13
5.1	Error function	13
5.2	Score function	14
5.2.1	Binary Score	14
5.2.2	MSAC score	14
5.2.3	MLESAC score	14
5.3	Cost function	15
5.4	Evaluation	15
5.4.1	The $T_{d,d}$ test	16
5.4.2	The Bail-Out test	16
5.4.3	Sequential probability ratio test	16
5.4.4	Preemptive RANSAC	17
5.5	Marginalizing Sample Consensus	17
6	Degeneracy	19
6.1	Line in 2D	20
6.2	Homography matrix	20
6.3	Fundamental matrix	20
6.4	PLUNDER	20
6.5	DEGENSAC	21
6.6	QDEGSAC	21
6.7	Closure	21

7	Local Optimization	23
7.1	Locally Optimized RANSAC	23
7.1.1	Inner RANSAC	23
7.1.2	Iterative RANSAC	23
7.2	Fixing Locally Optimized RANSAC	23
7.3	Weighted Local optimization	24
7.4	Graph-Cut RANSAC	24
7.4.1	Energy minimisation	24
7.4.2	Graph Cut	25
7.4.3	Local Optimization	25
8	Termination criteria	27
8.1	RANSAC sampling	27
8.2	MAGSAC	28
8.3	PROSAC sampling	28
8.3.1	Non-randomness	28
8.3.2	Maximality	29
8.4	DEGENSAC	29
8.5	SPRT	29
9	Neighbors Searching	31
9.1	Radius Search	31
9.2	K Nearest Neighbors Search	31
9.3	Grid Search	31
10	Implementation	32
10.1	Randomness	32
10.2	Estimation	32
10.3	Sorting points	33
11	Experiments	35
11.1	Dense sort	35
11.2	Model Estimation	36
11.3	All	36
12	Conclusions	39
	Bibliography	41
13	CD content	44

Abbreviations

RANSAC	RAN dom SA mple Consensus.
PROSAC	PRO gressive SA mple Consensus.
NAPSAC	<i>N</i> A djacent P oints SA mple Consensus.
MLE SAC	MA ximum L ikelihood E stimation SA mple Consensus.
MAGSAC	MAr Ginalizing SA mple Consensus.
GC-RANSAC	G raph C ut RANSAC.
P-NAPSAC	P rogressive NAPSAC.
DEGENSAC	DEGEN nerate SA mple Consensus.
QDEGSAC	Quasi DE generate SA mple Consensus.
PLUNDER	P ick L east UNDE generate R andomly.
SPRT	S equential P robability R atio T est.
USAC	U niversal RANSAC.
MSAC	M -estimator SA mple Consensus.
IRLS	I terative R eweighted L east S quares.
DLT	D irect L inear T ransformation.
EVD	E xtrême V iew D ataset.
KNN	K N earest N eighbors.
SVD	S ingular V alue D ecomposition.
PCA	P rincipal C omponent A nalysis.
LSQ	L east SQ uares.
LMS	L east M edian S quares.
LO	L ocal O ptimization.
GT	G round T ruth.
i D	i -th D imension.
A, H, F, E	A ffine, H omography, F undamental, E ssential matrix.

1 Introduction

Parametric model estimation is an important task in many areas. The estimated model must have desired relation to the given data. The contamination level, large size and/or high dimensionality of the data often complicates estimation problem. The data could be contaminated by outliers – the unexpected, possibly incorrectly stored or measured data. Non-robust methods mostly could not find *good* estimation in very contaminated data and often fail, whereas the robust algorithms usually succeed.

The robust estimation is a difficult problem. Not only the result accuracy, but also computational costs play important role. Many estimation algorithms use different techniques. The quality of those methods could be evaluated by at least two important factors: *the solution confidence* and *required computational time*. These factors objectively describe the properties of the algorithm. One of the popular and widely used algorithm that provides right balance between solution confidence and computational expense is RANdom SAMple Consensus.

RANSAC is robust algorithm proposed by Fischler and Bolles [11]. The simple estimation procedure is "repeatedly generate-verify model of minimal random sample and save so-far-the-best one". The sample is subset of points used to estimate the model. The single outlier in a sample could completely destruct estimated model and this is one of the reasons why estimation fails for larger sample set. RANSAC tries to find randomly *good* samples to generate model and avoid outliers. The support of a model is i.e., a set of inliers (the points consistent with the model). The solution confidence is obtained using probability of drawing a *good* sample. RANSAC can be seen as an optimisation method too, because its goal is to find a model with the largest support.

Generally, RANSAC could be used in various problems requiring estimations, although, it is mostly popular in computer vision tasks. The examples of RANSAC application are image matching, statistics, geometric relation estimation (e.g., multiple view geometry), image rectification, image mosaicing [14], short / wide baseline stereo matching [30, 27], motion segmentation [30] etc. In image processing problems the data can be contaminated by noise or incorrectly detected points, thus robust estimation is worth using. As it is shown in figure 1.1a, RANSAC managed to find the correct line model despite the high fraction of outliers (95%). Another example is the panoramic image depicted in figure 1.1b, which was obtained by a RANSAC-based homography estimation.

The challenge of model estimation consists in not only the data contamination, but also in the size of the (contaminated) data. The computational time increases significantly with decrease of the inlier ratio. In the cases of high image noise and/or data degeneracy, the performance of the original RANSAC can be significantly affected. This lead to continuous improvements of the RANSAC in many respects over the last years. For example, the additional model verification tests save evaluation time, this idea was implemented in randomised RANSAC with $T_{d,d}$ [5], RANSAC with sequential probability ratio test (SPRT) [20], and RANSAC with Bail-out test [26]. The new sampling procedure based on quality of points, their spatial coherence or confidence in each point was proposed in PROSAC [6], (P-)NAPSAC [1, 22], EVSAC [12]. The local optimisation methods that refine so-far-the-best model were presented in LO-RANSAC [8],

1 Introduction

or GC-RANSAC [2]. The degeneracy handling was discussed in DEGENSAC [10] or QDEGSAC [13]. The new model cost function was suggested in MSAC [32], MLESAC [31] and MAGSAC [3]. There are many other RANSAC-based algorithms that have different improvements.

The first universal framework including all advanced RANSAC-like algorithms was Universal RANSAC proposed by Raguram et al. [28]. However, several new methods such as GC-RANSAC [2], P-NAPSAC [1] or MAGSAC [3] provide more accurate results. The proposed framework USAC++ can be considered as an improvement of the above mentioned Universal RANSAC, due to not only inclusion of new algorithms, but also better architecture and implementation.

In the next sections the following parts of the RANSAC-based algorithms are reviewed and discussed:

1. The design of USAC++ framework,
2. Sampling methods,
3. Verification and evaluation of estimated model,
4. Degenerate configurations,
5. Local optimisation,
6. Termination criteria.

In the end, the implementation details of the proposed framework USAC++ are provided. The RANSAC-based algorithms that included in the framework are evaluated on publicly available real-world datasets of several multiple view geometry estimation problems.

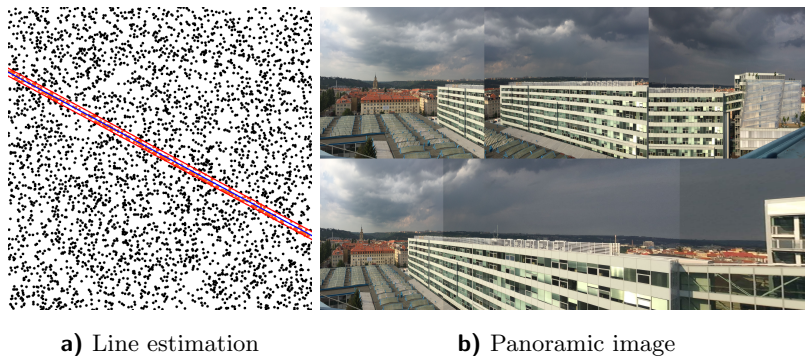


Figure 1.1 Example of line estimation in data with 95% of outliers. The second figure is example of panorama and rectification using homography.

2 Background

2.1 Design

The simplicity is one of the greatest features of RANSAC. Briefly the algorithm could be summarized in few steps: draw sample of minimal number, generate model, evaluate model, store so-far-the-best model.

Another good property of RANSAC is the small number of input parameters. The main arguments are points, threshold and confidence solution. The confidence is desired probability (closed to 100%) for getting *good* result. The threshold value is allowed tolerance of point error to model e.g., for computer vision estimation problems the threshold is measured in pixels. The threshold determines if point is inlier or outlier.

Algorithm 1 Random Sample Consensus.

Input: \mathcal{P}, τ, p – points, threshold, solution confidence.

Output: $\hat{\theta}^*$ – the best model of RANSAC.

```
1:  $inliers\_count^*, max\_iters := 0, \infty$ 
2: while  $iter < max\_iters$  do
3:    $\mathcal{S} := random\_sample(\mathcal{P})$  ▷ Draw minimal random sample
4:    $\hat{\theta} := estimate(\mathcal{S})$  ▷ Estimate model with sample  $\mathcal{S}$ .
5:    $inliers\_count := get\_inliers\_count(\hat{\theta}, \tau)$  ▷ Get number of inliers.
6:   if  $inliers\_count > inliers\_count^*$  then
7:      $\hat{\theta}^* := \hat{\theta}$  ▷ Save so-far-the-best model and score
8:      $inliers\_count^* := inliers\_count$ 
9:      $max\_iters := get\_max\_iterations(inliers\_count^*, p)$ 
10:   $iter := iter + 1$ 
```

The extended RANSAC algorithm has following abstract functions: **Sampling, Estimation, Verification, Evaluation, Degeneracy, Local Optimization, Termination Criteria**. The extended version was designed by exploring various of RANSAC-like algorithms so the each RANSAC-based method could be derived from the extended algorithm. For example, sampling does not need to be at random, or score does not need to be sum of inliers. Note, that not every step in algorithm 2 is necessary. The input parameters could be also extended w.r.t. used methods.

Algorithm 2 Extended Sample Consensus.

Input: \mathcal{P} – points; $\{\tau, p, \dots\}$ – set of input parameters.
Output: $\hat{\theta}^*$ – the best found model.

```

1:  $score^* := -\infty$ 
2: while termination criteria  $\mathcal{T}$  not met do
3:    $\mathcal{S} := sample(\mathcal{P})$  ▷ Draw minimal sample
4:   if !  $good\_sample(\mathcal{S})$  then ▷ Verify sample set.
5:     continue
6:    $\hat{\Theta} := estimate(\mathcal{S})$ . ▷ Estimate set of models with sample  $\mathcal{S}$ .
7:   for  $\hat{\theta} \in \hat{\Theta}$  do ▷ For each model in estimated model set.
8:     if !  $good\_model(\hat{\theta})$  then ▷ Verify model.
9:       continue
10:     $score := evaluate(\hat{\theta}, \tau)$  ▷ Get model score.
11:    if  $better(score, score^*)$  then ▷ Check if new score is better
12:      if  $degenerate(\hat{\theta})$  then ▷ Verify model  $\hat{\theta}$  on degeneracy.
13:        continue or try to 'fix' degenerate model
14:         $\hat{\theta}_{LO} := refine(\hat{\theta})$  ▷ Local optimization of model
15:         $score_{LO} := evaluate(\hat{\theta}_{LO}, \tau)$ .
16:        if  $better(score_{LO}, score)$  then
17:           $\hat{\theta}^* := \hat{\theta}_{LO}$  ▷ Save so-far-the-best model and score.
18:           $score^* := score_{LO}$ 
19:        else
20:           $\hat{\theta}^* := \hat{\theta}$ 
21:           $score^* := score$ 
22:           $\mathcal{T} := termination(\hat{\theta}^*, p)$  ▷ Update termination criteria
23:  $\hat{\theta}^* := estimate(\mathcal{I}_{\hat{\theta}^*})$  ▷ Refine the best found model by estimation on all inliers.

```

Note, that arguments of some functions in algorithm 2 were skipped for the sake of readability.

2.2 Two-View Geometry

The two-view geometry estimation problems are widely used in computer vision. The cameras usually used to provide the views. Every camera has extrinsic and intrinsic parameters. Consider a pinhole camera, which parameters could be easily written in matrix form.

The intrinsic matrix entails information about principal point offset (e.g. half of width and height), skew and focal lengths of camera. The aim of \mathbf{K} is mapping points from image-pixel 2D coordinate to camera 3D coordinate system and back (\mathbf{K} is invertible).

The extrinsic matrix includes rotation \mathbf{R} (defined by pitch, roll and yaw angles) and translation \mathbf{t} (3D position) of camera location in the world. It describes how world is transformed relative to the camera. Extrinsic model $[\mathbf{R} \ \mathbf{t}]$ maps world 3D point to camera coordinates. And inverse transformation from camera to the world is also possible.

Extrinsic and intrinsic matrices altogether allow to project world point $(x_w, y_w, z_w)^\top$ to the image coordinates. Then two points $\mathbf{x} = (x, y, 1)^\top$ and $\mathbf{x}' = (x', y', 1)^\top$ corre-

spend if two cameras map the same world point:

$$\left\{ \begin{array}{l} d \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{K} [\mathbf{R} \mathbf{t}] \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \\ d' \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \mathbf{K}' [\mathbf{R}' \mathbf{t}'] \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \end{array} \right.$$

d is distance from 3D world point to camera origin and \mathbf{x}, \mathbf{x}' are points in homogeneous coordinates.

In general, $\mathbf{K}, \mathbf{R}, \mathbf{t}$ remain to be unknown and usually denoted as ground Truth parameters, because they lead to solution of estimation problems.

3 Estimation problems

In this section are presented some of the most popular computer vision / algebraic estimation problems.

3.1 Line in 2D

Line estimation in two dimensional subspace is the most popular example of RANSAC. The best line must cover the biggest number of points within predefined threshold.

The real world examples are approximation of data points with linear trend. In the figure 3.1 all points are e.g., some measurements. Suppose, those points must have linear trend, although the red ones do not agree with assumption and could be viewed as outliers. RANSAC can robustly estimate the desired linear trend (line).

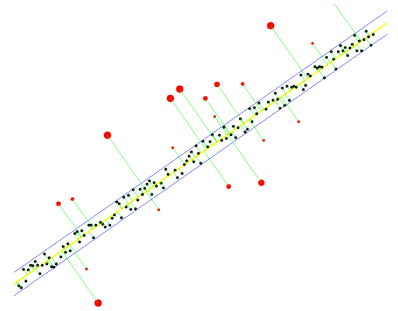


Figure 3.1 Example of linear trend estimation. Black points are inliers, red – outliers.

3.2 Homography matrix (H)

$$\mathbf{x}' \sim H \mathbf{x}$$

The homography 3×3 matrix H project 2D planar points in homogeneous coordinates from position on the first view (image) to the same position on the second view and backward (H is inverse). Unlike affine transformation the homography preserves also the perspective projection.



Figure 3.2 Homography matrix estimation

For estimation is using Direct Linear Transformation (DLT) algorithm [15]. The homography matrix has 8 degrees of freedom thus four points are required for estimation (each point gives 2 equations). In [15] was investigated that origin and scale of image coordinate may have negative influence on DLT algorithm and make it unstable, especially when the set of equations is overdetermined. In [15] was suggested to use normalization of points so that the average distance of a point \mathbf{x} from the origin is equal to $\sqrt{2}$. Then searching (planar) homography is

$$H = T'^{-1} \tilde{H} T, \text{ where}$$

T', T stand for normalization matrices ($\tilde{\mathbf{x}} = T\mathbf{x}$)

The homography estimation has various application. For example, image rectification, image stitching (mosaicing, panorama see 1.1b), image matching etc. The problem of estimation is to find the homography with the biggest support – i.e., number of correctly (within threshold) projected points.

In the figure 3.2 are two corresponded images. The green points are given. The red points on the first image are projected points from the second image using homography matrix and similarly red points on the second image are projected points from the first image. The given points on the left building are mismatched – outliers, their projections are bad. The black lines show the found matches – when the distance from projected and given point is less than threshold.

3.3 Fundamental matrix (F)

$$\mathbf{x}'^\top F \mathbf{x} = 0$$

$$F = \mathbf{K}_2^{-\top} \mathbf{R}_{1,2} [\mathbf{t}_{1,2}]_\times \mathbf{K}_1^{-1} \quad (3.1)$$

The limitation of homography transformation is necessity of points on plane, which means that features out of plane could not be matched correctly. The idea of fundamental 3×3 matrix F is generating (epipolar) lines in image coordinates that go through two correspondent points. The points on line in this case could be considered as matches. Then $F\mathbf{x}$ is line on the second image, and $\mathbf{x}'^\top F$ is line on the first image.

Fundamental matrix has seven degrees of freedom, because points are homogeneous ($F_{33} = 1$) and fundamental matrix also has zero determinant¹ constraint. F could be estimated with seven-points algorithm [15] with up to three solutions. Or by eight-points algorithm with perfect (i.e. noise-free) points. The normalization of points as in homography case is also recommended in [15]: $F = T'^\top \tilde{F} T$

In the figure 3.3 the black lines show again matches by the criterion if two corresponded points lie within threshold on epipolar line. However, the matches must not necessary be real correspondences, because even incorrectly detected pairs by coincidence could lie on the same line.



Figure 3.3 Fundamental matrix estimation

3.4 Essential matrix (E)

$$\mathbf{y}'^\top E \mathbf{y} = 0$$

$$E = \mathbf{R}_{1,2} [\mathbf{t}_{1,2}]_\times = \mathbf{K}_2^\top F \mathbf{K}_1 \quad (3.2)$$

The idea of essential 3×3 matrix E is similar to fundamental matrix, but the epipolar lines go through normalized (by third coordinate) points \mathbf{y}, \mathbf{y}' corresponding to the 3D camera scene. E has five degrees of freedom: translation and rotation has by 3 unknowns, but essential matrix also has also zero determinant constraint. Thus minimal model could be estimation using five-points algorithm [24] with are up to ten solutions.

The examples of epipolar geometry application is image rectification or constraining data points in images by epipolar lines to detect wrong correspondences.

From equations 3.1 and 3.2 fundamental and essential matrices can be found from camera parameters, where $\mathbf{R}_{1,2}, \mathbf{t}_{1,2}$ stand for rotation and translation between cameras.

¹ F, E have rank 2, because were constructed by multiplication of skew symmetric matrix $[\cdot]_\times$, $\text{rank}(AB) \leq \min(\text{rank}A, \text{rank}B)$.

4 Sampling

The key of robustness in RANSAC over other estimation algorithms (e.g. Least Squares Fitting) is sampling procedure. In presence of noise or high outlier percentage the non-robust iterative minimisation methods could not find the best model and usually converge to local minimum instead of global. This happens because computation is based on large or entire subset of points. Whereas RANSAC makes estimation on minimal number of sample required. The main question is: *what points should be chosen*. In this chapter we describe the most popular samplers.

The simple observation about estimation problem is:

The model is good if it was generated by inliers.

But what is actually inlier? In the statement above *inliers* mean the subset of points that *lie* under ground Truth model and equivalently generate it. Neither *inliers* nor GT model we never know as no estimation is needed then. Also by (estimated) inliers we may consider points that are closed to be *inliers*. In some context virtual inliers are points that in current state could be viewed as inliers.

4.1 Random Sample Consensus

The main idea of original RANSAC proposed by Fischler et al. [11] is random sampling. Independently in every iteration needed minimum number m from all N points are taken uniformly at random for estimation. Assume choosing samples without repetition $\binom{N}{m}$ to avoid degeneracy.

So, RANSAC tries to find inliers randomly to make a correct estimation.

4.2 Progressive Sample Consensus

The more advanced and efficient (progressive) sampling was presented by Chum and Matas [6]. The assumption of PROSAC is data ordering by quality function $q(\mathbf{x})$ which assigns a given point \mathbf{x} the score. Thereby points are descendingly ordered from the highest quality, that also implies probability of being inlier

$$\forall i, j \in [1, N] : i < j \Rightarrow q(\mathbf{x}_i) \geq q(\mathbf{x}_j) \Rightarrow P_{\mathbf{x}_i \in \mathcal{I}} \geq P_{\mathbf{x}_j \in \mathcal{I}}$$

Intuitively, by this assumption every sample should be drawn from the top-ranked points. Authors of PROSAC derived and explained algorithm in [6] how to do sampling correctly, which is briefly summarised in this section.

First of all we need to know the sample range of the most quality points, which will be *progressively* increased by number of iterations (drawn samples). Thus we need to define subset \mathcal{U}_l of size $l \in [m, N]$ highly ranked points and stopping length l^* – maximum size of $\mathcal{U}_l, l \leq l^*$. Also denote T_l, T_{l+1} as average number of samples of size m which consist of points only from $\mathcal{U}_l, \mathcal{U}_{l+1}$ resp. and are drawn by RANSAC. $T_{l+1} - T_l$ is again number of samples that contain one point \mathbf{x}_{l+1} from $\mathcal{U}_{l+1} = \mathcal{U}_l \cup \mathbf{x}_{l+1}$ and $m - 1$ points from \mathcal{U}_l . Because $T_{l+1} - T_l$ is not integer define:

$$T'_{l+1} - T'_l = \lceil T_{l+1} - T_l \rceil \quad \text{if } l = m \text{ then } T'_l = 1$$

So sampling procedure in PROSAC is drawing $T'_{l+1} - T'_l$ samples that contains \mathbf{x}_l and $m - 1$ points from \mathcal{U}_l at random. By implementation reasons the growth function $g(t)$ that defines size of \mathcal{U} should be also introduced as

$$g(t) = \min \{l \mid T'_l \geq t\}$$

It says that subset \mathcal{U}_l of top-ranked points has minimal size l such that RANSAC's average number of samples containing points from \mathcal{U}_l is bigger than t iterations of PROSAC.

Now, sample set of PROSAC in t -th iterations is:

$$\mathcal{S}_t = \mathbf{x}_{g(t)} \cup \mathcal{S}'_t, \quad \mathcal{S}'_t \subset \mathcal{U}_{g(t)-1}, \quad |\mathcal{S}'_t| = m - 1$$

From all above PROSAC converges (with the worst case) to RANSAC. *Termination length l^* will be described in termination criteria section.*

4.3 N adjacent points Sample Consensus

It was observed that inliers are usually closed to each other. In the figure 4.1 black points are inliers and gray points are outliers. Inliers in the figure are closed, whereas outliers are not.

In NAPSAC by Myatt et al. [22] was proposed a new local sampling with assumptions that inliers are tend to be closer than outliers. (Without loss of generality) assume that inliers are perturbed by Gaussian noise and outliers are uniformly distributed. In D -dimensional subspace exist local structures of d -dimensional ($d \leq D$) manifold. In this case number of inliers within hypersphere of radius r centered in point of manifold \mathbf{x} is proportional to r^d and number of outliers is proportional to r^D . Then roughly probability of inlier inside this hypersphere is bigger for smaller radius

$$P_{\mathbf{x}' \in \mathcal{I}_{\mathbf{x}}^{(r,d)}} \approx \frac{\alpha r^d}{\alpha r^d + \beta r^D} = \frac{1}{1 + \frac{\beta}{\alpha} r^{D-d}} \xrightarrow{r \rightarrow 0} 1$$

in the figure 4.1 points inside green circle are all inliers, but points inside red circle include outliers too.

NAPSAC's sampling procedure is next:

1. Select initial point \mathbf{x}_0 (uniformly) at random from all points.
2. Select $m - 1$ points at random inside hypersphere of radius r with centre in \mathbf{x}_0 .
3. If not enough points for minimal sample size m then choose another initial point.

So, outliers will be chosen less than inliers, because by assumption they do not have big neighborhood. By experiments in [22] NAPSAC is much better than RANSAC in high noise and high dimension subspace.

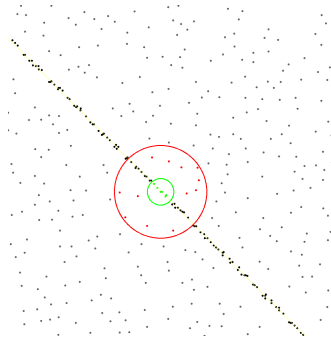


Figure 4.1 Line estimation. Inliers (black) points are closer than outliers (gray).

Nevertheless, NAPSAC has considerable disadvantages. The worst case of NAPSAC would be if inliers are not closed, e.g. uniformly distributed. Or for practical estimation problems, as image matching, NAPSAC could select ill-samples that lead to degenerate configurations. For example, closed points generates bad homography, or points from same plane produce incorrect epipolar geometry.

4.4 Progressive NAPSAC

One of the biggest benefit of NAPSAC over RANSAC is performance in high dimension subspace: *"... required iterations (of RANSAC) increases exponentially with dimensionality."* [22]. But disadvantages as unexpected distribution of inliers or trend to degeneracy for $H - F - E$ estimations make NAPSAC in those cases even worse than RANSAC. Also, for some other practical problems as rigid motion the global sampling could be worth than local.

Barath et al. in [1] proposed more general sampling that starts locally as NAPSAC and progressively converges to global RANSAC sampling. The goal is to keep positive sides of NAPSAC as finding local structures earlier and avoid cons as degeneracy and local addiction.

The first step of algorithm in [1] is to choose initial point \mathbf{x} from all points e.g. at random. Then comes progressive sampling based on neighborhood system $\mathcal{N} \subseteq \mathcal{P} \times \mathcal{P}$ (denote $\mathcal{N}_{i,j}$ as point \mathbf{x}_i has neighbor \mathbf{x}_j). Rest of sample of size $m-1$ are drawn from the closest neighbors progressively including the farthest ones. By this it reminds PROSAC, but because every point may belong to different local structure, in P-NAPSAC growth function, iteration counter and subset of top-ranked points are given independently for each point \mathbf{x}_p . The subset of l closest neighbors of point \mathbf{x}_p is denoted as $\mathcal{U}_{p,l}$. Then quality function represents distance from point \mathbf{x}_p to its neighbors, so neighbors that closer points has bigger 'quality':

$$i < j \in [1, l] \Rightarrow \|\mathbf{u}_i - \mathbf{x}_p\|_2 \leq \|\mathbf{u}_j - \mathbf{x}_p\|_2, \quad \mathbf{u}_i, \mathbf{u}_j \in \mathcal{U}_{p,l} \subseteq \mathcal{N}_p$$

Growth function $g(t_p)$ is the same as in PROSAC with exception that T'_l is average number of samples of size $m-1$, because the first point is chosen from all points. Iteration number t_p increases for point \mathbf{x}_p if \mathbf{x}_p is initial \mathbf{x}_{p_0} or the closest neighborhood of \mathbf{x}_{p_0} contains \mathbf{x}_p and vice versa.

Finally, sample of p -th point in iteration t_p is

$$\mathcal{S}_{p,t_p} = \{\mathbf{x}_{p_0}, \mathbf{u}_{g(t_p)}\} \cup \mathcal{S}'_{p,t_p}, \quad \mathcal{S}'_{p,t_p} \subseteq \mathcal{U}_{p,g(t_p)-1}, \quad |\mathcal{S}'_{p,t_p}| = m-2$$

Within neighborhood points are drawn at random.

4.5 Groups Sample Consensus

GroupSAC proposed by Ni et al. [23] is another sampling method which assumes that points (correspondences) could be divided into groups. Recommended and tested grouping in [23] was by optical flow based clustering or image segmentation.

The main idea of GroupSAC is under assumption that some groups may contain big inlier ratio and other groups contain mostly outliers. For example, for homography estimation the groups with bigger number of inliers are planes.

The sampling strategy reminds both PROSAC and NAPSAC, because it starts sampling from smaller groups and those ones that have big number of points and progressively converges to global RANSAC sampling. Groups could be viewed as local

4 Sampling

structures too. It was shown in [23] that smaller groups and groups with high number of correspondences have bigger inlier ratio, so it is more likely to draw correct sample earlier.

According to the experiments in [23], GroupSAC could be even more efficient than PROSAC. However, the evident disadvantage of this method is grouping of data. Because sometimes it may require a prior knowledge of the estimation problem or construction and searching for the groups needs a time too.

5 Model Evaluation

The evaluation procedure should be careful part of each algorithm, because it basically classifies model as *good* with big support and *bad* with small support. Obviously, the mismatched classification must be avoided. The RANSAC considers all models with so-far-the-biggest support as *good*. The model evaluation consist with three computation steps provided by:

1. the error function – gives the point its error distance w.r.t. evaluated model.
2. the score function – gives the point score based on its error.
3. the cost function – gives the cost of model based on point scores.

5.1 Error function

Denote error function $\varepsilon : \mathbb{R}^n \rightarrow \mathbb{R}$ as fitness of (concatenated correspondence) point dimension n to the model θ and $d : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$ distance function of $\|\cdot\|_2$ between inhomogeneous points of dimension m .

Affine and projective geometry

Consider 2D transformation (i.e. isometry, scale, rotation, affine, homography etc.) represented by 3×3 matrix M such that exists inverse transformation by M^{-1} . Then reprojection error is calculated as geometric distance between measured and estimated point. In [15] was recommended to use symmetric transfer error which is sum of reprojection distances of forward and backward mapping:

$$\varepsilon([\mathbf{x}\mathbf{x}']) = d(\mathbf{x}', \underbrace{M\mathbf{x}}_{\hat{\mathbf{x}}'}) + d(\mathbf{x}, \underbrace{M^{-1}\mathbf{x}'}_{\hat{\mathbf{x}}}) \quad (5.1)$$

However, vectors $(\mathbf{x}' - \hat{\mathbf{x}}')$, $(\mathbf{x} - \hat{\mathbf{x}})$ (denote already homogeneous points) in most cases appears to be collinear and computationally it is very inefficient to calculate both errors. The spent time for two errors computation is not worth a little rise in accuracy. The calculation of one reprojection error is enough.

Epipolar geometry

Continue with Two-View geometry, denote epipolar lines, which are generated by $M_{3 \times 3}$, here M is for example F fundamental or E essential matrix. The quality of (epipolar) lines $M\mathbf{x}$ and $\mathbf{x}'^T M$ can be measured by geometric distance from given points to them. Then error is sum of distances from both points to estimated lines:

$$\varepsilon([\mathbf{x}\mathbf{x}']) = \frac{|\mathbf{x}'^T M\mathbf{x}|}{\|((M\mathbf{x})_1, (M\mathbf{x})_2)\|_2} + \frac{|\mathbf{x}'^T M\mathbf{x}|}{\|((\mathbf{x}'^T M)_1, (\mathbf{x}'^T M)_2)\|_2} \quad (5.2)$$

Another, even more used quality of M is Sampson distance (first-order geometric error):

$$\varepsilon([\mathbf{x}\mathbf{x}']) = \frac{(\mathbf{x}'^T M\mathbf{x})^2}{(M\mathbf{x})_1^2 + (M\mathbf{x})_2^2 + (\mathbf{x}'^T M)_1^2 + (\mathbf{x}'^T M)_2^2} \quad (5.3)$$

5.2 Score function

Denote score function $s(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$ of point \mathbf{x} . RANSAC could have unlimited number of score functions using various combinations of kernel, threshold and error distances. Here we describe the most used ones.

5.2.1 Binary Score

RANSAC assumes that inlier is the point which error to estimated model is smaller than allowed boundary τ . Then binary score is the clearest way for evaluation: 1 - Inlier, 0 - Outlier:

$$s_{RSC}(\mathbf{x}) = \begin{cases} 1, & \varepsilon^2(\mathbf{x}) < \tau^2 \\ 0, & otherwise \end{cases} \quad (5.4)$$

5.2.2 MSAC score

The biggest problem of binary score is that geometrically more accurate model may have same number of inliers as less accurate. Although M-estimator is a whole class of estimators, in [32] Torr et al. presented score as truncated quadratic function. Experimentally and intuitively score defined using error distance is better than binary score, because it also carries information about error distance of inlier.

$$s_{MSC}(\mathbf{x}) = \begin{cases} \varepsilon^2(\mathbf{x}), & \varepsilon^2(\mathbf{x}) < \tau^2 \\ \tau^2, & otherwise \end{cases} \quad (5.5)$$

5.2.3 MLESAC score

Another way of computing score was presented in [31] by Torr et al. The assumption holds that noise in image is Gaussian with 0 mean and uniform standard deviation σ . Hence using maximum likelihood estimation the probability density function of presence of noise in data could be defined. The expected error distance of inlier has normal distribution and error of outlier is distributed uniformly. For the reason that features could be mismatched then fairly to use probability of error as mixture of normal and uniform distributions. Denote γ - mixture parameter and $\langle -\frac{v}{2}; \frac{v}{2} \rangle$ is pixel range of outlier.

In general σ, v and γ are unknown, but it is possible to make estimation of them. For instance, in [31] was proposed to obtain mixture parameter γ with Expectation-Maximisation algorithm treating inlier-outlier as missing data and making suitable expectation.

$$P_{\varepsilon_{inlier}}(\mathbf{x}) = \frac{e^{-\frac{\varepsilon^2(\mathbf{x})}{2\sigma^2}}}{\sqrt{2\pi}\sigma}, \quad P_{\varepsilon_{outlier}}(\mathbf{x}) = \frac{1}{v} \quad (5.6)$$

$$P_{\varepsilon}(\mathbf{x}) = \gamma P_{\varepsilon_{inlier}}(\mathbf{x}) + (1 - \gamma) P_{\varepsilon_{outlier}}(\mathbf{x}) \quad (5.7)$$

For sake of minimisation problem, MLESAC minimise sum of negative logarithmic likelihood and score in this case is:

$$s_{MLSC}(\mathbf{x}) = -\log P_{\varepsilon}(\mathbf{x}) \quad (5.8)$$

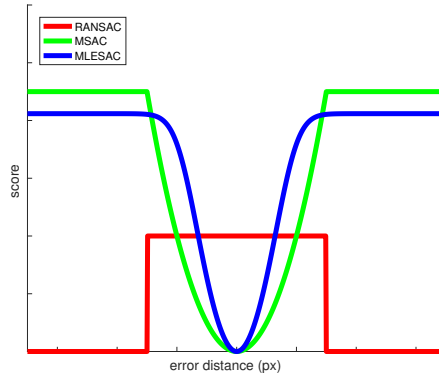


Figure 5.1 Score functions.

Note, that MLESAC and MSAC as well are classes of estimators. MLE approach for Gaussian noise is only expected example and MLESAC could derive for other distributions of noise too.

According to the experiments in [31], MLESAC is slightly better than MSAC and more accurate up to 10% than RANSAC.

Note, that figure 5.1 shows only score (not loss) of point \mathbf{x} which has error distance $\varepsilon^2(\mathbf{x})$. So, RANSAC maximises a (binary) score, while MSAC and MLESAC minimise.

5.3 Cost function

For most RANSAC-like algorithms the cost C of model θ is defined as sum of scores over points \mathcal{P} to model θ :

$$C(\theta) = \sum_{\mathbf{x} \in \mathcal{P}} s(\mathbf{x}) \quad (5.9)$$

5.4 Evaluation

In standard RANSAC cost of model is calculated over all points and it implies time-consuming evaluation process for large number of data size N . While time required to calculate minimal model is constant. Evidently, for big N time to qualify i.e. incorrect model much exceeds time for its estimation. The average time of running standard RANSAC is:

$$\bar{t}_{RANSAC} = k (t_S + t_\Theta + \overline{|\Theta_S|} t_{eval}) \quad (5.10)$$

Where k is number of samples drawn, t_S is time to generate sample, t_Θ is time to compute models, $\overline{|\Theta_S|}$ is average number of models per sample and t_{eval} is time to evaluate the model. So time to generate sample and compute model is not significant, whereas evaluation of model takes the most running time.

It was proposed several methods to reduce evaluation time by rejecting *bad* models.

Note, by *good* model is meant that it was computed by inliers and *bad* model was generated by contaminated sample, but occasionally happens that outlier or quasi-degenerate data might produce correct model, e.g. in epipolar geometry. However, in computation this fact was ignored.

5.4.1 The $T_{d,d}$ test

The idea of $T_{d,d}$ test proposed by Chum and Matas in [5] is verification if all d from d points randomly chosen from whole data set are inliers to current hypothesized model. If they are inliers then comes further evaluation of model on the rest points, otherwise new model is generated. The average time of evaluation with this verification test is:

$$\bar{t}_{eval} = P_{S \subseteq \mathcal{I}} (\alpha N + (1 - \alpha) \bar{t}_\alpha) + (1 - P_{S \subseteq \mathcal{I}}) (\beta N + (1 - \beta) \bar{t}_\beta), \quad \alpha = \epsilon^d, \beta = \delta^d \quad (5.11)$$

α is the probability that all d random points are inliers (ϵ is inlier ratio) and β is the probability that d points consistent with 'random' model. Similarly $\bar{t}_\alpha, \bar{t}_\beta$ is average time spent in verification of d points.

In [5] was derived the optimal number of points for verification $d^* = 1$.

5.4.2 The Bail-Out test

The concept of Bail-Out test presented by Capel in [26] is similar to $T_{d,d}$ test. Again, the decision about model rejection is based on subset of d random points. The main factor is inlier ratio ϵ_d of d -points, if $\epsilon_d < \epsilon^*$ then it is very unlikely that tested model is *good*.

The odds if current model, that was evaluated on d points, has bigger support I (number of inliers) than so-far-the-best model is

$$P(I > I^*) = \sum_{I=I^*}^N P(I|I_d, d, N) \quad (5.12)$$

Desirably, $P(I > I^*)$ should be high, so if it below threshold (e.g. 1%) then model is *bad*. Calculation of this probability is quite inefficient, so in [26] was suggested to use approximation of I_d . The recommended ones are hypergeometric, binomial (low bound for small d) and normal for large d .

$$I_d \sim \mathcal{N}(\mu, \sigma^2) \quad (5.13)$$

The minimal support $I_d^{\min} = \lfloor d \hat{\epsilon} - z_{conf} \sigma \rfloor$ for each d makes the decision if model is *bad*.

Input: θ – model to evaluate.

Output: Score of θ , e.g. I – number of inliers.

- 1: $I := 0, d := 0$
 - 2: **for** $\mathbf{x} \in$ randomized points set \mathcal{P} **do**
 - 3: $d := d + 1$
 - 4: **if** \mathbf{x} is inlier **then**
 - 5: $I := I + 1$
 - 6: **if** $I < I_d^{\min}$ **then**
 - 7: Model is *bad*. Terminate verification.
-

5.4.3 Sequential probability ratio test

The more advance method of evaluation similar to $T_{d,d}$ or Bail-Out test was presented in [20] by Matas and Chum. It includes Wald's Sequential Probability Ratio test to decide if model is *good* or *bad*. So, the errors of two types are: 1) rejecting a *good* model

and 2) acceptance of *bad* one, but in RANSAC only error of the first kind could be observed. The goal is again to minimise time of evaluation.

Denote α as probability of rejecting *good* estimation H_g , and H_b is label for *bad* model. Then consistency of j -th point to *good* model is based on likelihood ratio:

$$\lambda_j = \prod_{i=1}^j \frac{P(\mathbf{x}_i|H_b)}{P(\mathbf{x}_i|H_g)} \quad (5.14)$$

$P(\mathbf{x}_i|H_g), P(\mathbf{x}_i|H_b)$ are conditional probabilities of belonging point \mathbf{x}_i to *good* and *bad* model resp. $P(\mathbf{x}_i|H_g)$ could be approximated as inlier ratio ϵ among all points, and $P(\mathbf{x}_i|H_b) = \delta$, where δ is the probability of random event (parameter of Bernoulli distribution). The initial probabilities δ_0 and ϵ_0 are difficult to estimate as it requires the knowledge of problem beforehand. Although in [20] was recommended to obtain δ by geometric considerations, i.e. as a fraction of the area that supports a hypothesised model. And ϵ_0 could be guessed as the lowest bound of inlier ratio.

The verification test must be on randomized data points. If for j the likelihood ratio λ_j is bigger than decision threshold A then model is *bad*, otherwise increment j and continue testing. After the test the *good* model is accepted or *bad* is denied.

The probabilities ϵ and δ are changing during the algorithm. The lower bound of $\hat{\epsilon}$ is inlier ratio of the biggest support so far and $\hat{\delta}$ could be estimated as the average fraction of consistent data points in rejected model. If model is accepted and has the biggest support then $\epsilon \leftarrow \hat{\epsilon}$ and $\delta \leftarrow \hat{\delta}$. If tested hypothesis is rejected and $\hat{\delta}$ differs than δ more than 5% then $\delta \leftarrow \hat{\delta}$.

In [7] was experimented performance of RANSAC with $T_{d,d}$, Bail-Out and SPRT. Among three of them the best one w.r.t. time seemed to be RANSAC with SPRT.

5.4.4 Preemptive RANSAC

Completely new way of fixed-time evaluation was proposed by Nister in [25]. Denote positive (exponentially) decreasing function $f(i), i = 1..N$ which defines how many models should be verified in i -th step. Only $f(1)$ hypothesis are generated in the beginning. For each i only $f(i)$ so-far-the-best models are evaluated based on score of i points. The procedure stops if $i > N$ or $f(i) = 1$. In any case the best model is chosen.

Because the algorithm has almost constant time of running it became popular for real-time problems as live structure or motion estimation, or where fixed computational time is required.

However, the limitation of preemptive RANSAC as was mentioned in [20] is "... *fixed number of models is evaluated, which is equivalent to an a priori assumption that the fraction of inliers is known*". Basically, for cases with small number of inliers the method would rather fail, whereas standard RANSAC tests more samples and has bigger chances to find a good model.

5.5 Marginalizing Sample Consensus

The previous methods and score functions in this section depends on very important user defined threshold value τ . If threshold is too big or low, the estimation result will change significantly. For example, big threshold value could tolerate outliers. Or if data points are fairly perturbed by noise than estimation is inaccurate and small threshold is very pessimistic in this case. Also threshold should depend on the resolution of

image, because for computer vision estimation problems threshold is usually measured in pixels. The recommended threshold in [31] is based on standard deviation of image noise $\tau = 1.96\sigma$, but σ in most cases unknown.

Barath et al. in [3] proposed novel RANSAC-like algorithm that does not require threshold τ . Unlike RANSAC that decides if point is inlier by threshold MAGSAC marginalise the likelihood of point being inlier over standard deviation of noise. In this case MAGSAC needs the upper bound of noise level σ_{\max} , which could be fairly large (e.g. 10 px [3]).

Assume that standard deviation of noise is uniformly distributed $\sigma \sim \mathcal{U}(0, \sigma_{\max})$, because no prior information is given. The distribution of inliers and outliers is uniform: inlier $\sim \mathcal{U}(0, \sigma)$, outlier $\sim \mathcal{U}(0, v)$ (v is e.g. image range, recall MLESAC above). Suppose that the residuals of inliers in every dimension of ρ -dimensional subspace are independent and normally distributed with zero mean and standard deviation of image noise $\mathcal{N}(0, \sigma^2)$. Then fractions $\frac{\varepsilon^2}{\sigma^2}$ have chi-squared distribution $\mathcal{X}^2(\rho)$ with ρ degrees of freedom.

The distribution of inlier residuals in this case is given by density function $g(\varepsilon | \sigma)$ and distribution of outlier residuals is uniform within interval $[0; v]$.

Using the maximum likelihood estimation in MAGSAC was derived the likelihood of model θ for std. dev. of noise σ : $L(\theta, \mathcal{P} | \sigma)$. The cost function in MAGSAC is based on the likelihood of model marginalised over the σ .

6 Degeneracy

"The state or quality of being degenerate."

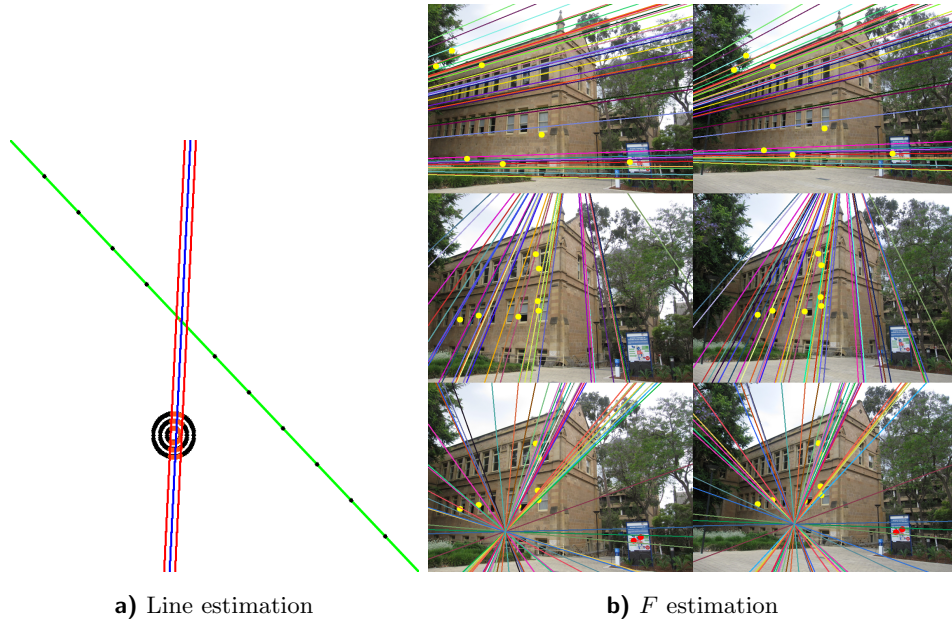


Figure 6.1 In a) is example of degeneracy in line estimation. The red line has the biggest found support – points within threshold, but those points that make circles are outliers. The desired green line lie on inliers. In b) is example of degeneracy in fundamental matrix estimation from ADELAIDERMF dataset. The first pair of images has correct epipolar geometry, the epipolar geometry on the second pair is degenerate and has big support. The epipolar geometry on the third pair is 'fixed' second one using plane-and-parallax algorithm.

Originally RANSAC considers the model with biggest found support as so-far-the-best. Intuitively, model with smaller score is worse, but this assumption is not completely true. By contradiction, could be shown that exists model with less score which is better.

In [15] Hartley et al. defined term *degenerate* as follows:

"A situation where a configuration does not determine a unique solution for a particular class of transformation is termed degenerate."

Similarly, *degenerate data* – points that provide not enough constraints to compute correct relation [13], which, basically means that degenerate data imply degenerate configuration. So, the stronger assumption for so-far-the-best model, slightly modified from [4] is:

A model consistent with a sample that is not contaminated by outliers or degenerate data and does not contain a degenerate configuration has larger support.

The estimation problems have different degenerate configuration and data:

6.1 Line in 2D

This is simplest case to show degeneracy which can occurred in RANSAC. In the figure 6.1a estimated line has the biggest found number of inliers, but obviously, desired model must lie on the points with linear trend. The cause of degeneracy in this case is estimation by sample containing outliers.

Degenerate configuration for line is estimation from two exactly the same points. Because, then line has arbitrary orientation and by Hartley's et al. definition there is no unique solution.

6.2 Homography matrix

In [15] Hartley et al. showed that for homography matrix estimation using DLT algorithm require four points, where every triple of them are not collinear.

If three points are collinear on one correspondent image and not collinear on second, it means that H does not exist, because homography preserves collinearity. If three points are collinear on both correspondence then in DLT algorithm eight equations are linearly dependent and no unique solution could be found (degenerate configuration). So, degenerate data are collinear points.

6.3 Fundamental matrix

In [15] Hartley et al. classified degeneracies that could happen in F -estimation as follow:

- i Estimation from $n \geq 8$ perfect (noise-free) points in general position using 8-points algorithm gives unique non-degenerate solution.
- ii Estimation from $n = 7$ correspondences using 7-points algorithm generates 1-3 solutions, some of them may be (non-)degenerate.
- iii Estimation from $n \geq 6$ perfect points related by homography $\mathbf{x}'_i \sim H\mathbf{x}_i$ produces family of solutions.

In [15] was also mentioned, that degeneracy occurs when all points lie on a same plane: *"In this case, all the points plus the two camera centres lie on a ruled quadric surface, namely the degenerate quadric consisting of two planes – the plane through the points, plus a plane passing through the two camera centres."*

Generally, in Two View Geometry Unaffected by Dominant Plane proposed by Chum et al. [10] has been shown and proved that 5 or more points from 7 that are inliers and lying on dominant plane produce wrong epipolar geometry that might have big score.

In [10, 15] degeneracy was causing by H -degenerate configuration. Because epipolar geometry in this case is consisted with homography: $\mathbf{x}'_i \sim H\mathbf{x}_i$ satisfy $\mathbf{x}'_i F \mathbf{x}_i = 0$.

6.4 PLUNDER

Pick Least Undegenerate Randomly algorithm described in [33] by Torr et al. was the first algorithm that tried to find degeneracy for different estimation problems in RANSAC. The method is divided on sampling and model selection parts. In sampling phase, sample selected at each iteration, determines the model that will gain support from them. And in model selection part the most appropriate model is picked to describe the data.

6.5 DEGENSAC

In [10] a novel part of RANSAC algorithm for detecting and fixing degeneracy in fundamental matrix estimation was presented. Method checks so-far-the-best F^* model of RANSAC on H -degenerate sample test. If 5 or more points are related by homography then (optionally) new H homography matrix is computed and also new fundamental matrix F^H is estimated by plane-and-parallax algorithm [15]. If F^H has bigger support than F^* then $F^* \leftarrow F^H$.

Briefly, H -degenerate sample test attempts to estimate plane homography from fundamental matrix. According to [15], 3 points and F is enough to find homography. If constructed in this way homography is consistent with at least 5 sample points then F is degenerate.

In the figure 6.1b yellow (7) points were used for minimal estimation. The first epipolar geometry is correct and has the biggest support, only 4 points are on the dominant plane. The second one is degenerate, because all points lying on the same plane. The third one was fixed using plane-and-parallax algorithm, two additional random points marked red colour generated new fundamental matrix that has almost twice bigger support than degenerated.

6.6 QDEGSAC

A more universal approach to estimate different models on (quasi-)degenerate data was proposed in [13] by Frahm and Pollefeys.

One of the ideas of QDEGSAC is that mapping (represented by matrix L), constructed from degenerate data, for model θ estimation (e.g. $L\theta = 0$) does not constrained enough. Which means L does not have a full rank required for unique estimation. Authors also mentioned that proposed algorithm could be interpreted as a robust measurement of rank L .

QDEGSAC unlike DEGENSAC is not component of RANSAC (however includes it) and is divided on 3 parts. In the first phase RANSAC estimates a full model assuming that data are non-degenerate. Output is inliers and outliers. In model selection part the robust rank detection is performed on generated constraints from inliers. In the last model completion phase the outliers are tested for non-degenerate inliers to compute the correct relation.

6.7 Closure

In conclusion of this chapter, degeneracy is quite important problem in RANSAC: *"degenerate data are likely to lead to a numerically ill-conditioned estimation"* [15] which one might have even large support. It is worth to have additional tests and constraints for sampling or estimation parts for sake of evaluation time savings. For example, some non time-expensive tests (i.e, test on collinearity) could be applied for every sample.

7 Local Optimization

Local optimization (LO) is definitely the most improving part of original RANSAC. The basic idea of LO in RANSAC-like algorithms is updating so-far-the-best model during the run (locally) with non minimal estimation using different techniques.

7.1 Locally Optimized RANSAC

Locally Optimized RANSAC is the first LO method proposed by Chum and Matas [8]. Authors explained general idea by observation that, experimentally, generating model from minimal number of sample leads to poor estimation in the presence of noise. On another side, the main reason for minimal estimation is that every next point in the sample exponentially decreasing the probability of being inlier ($(\frac{|Z|}{N})^m \xrightarrow{m \rightarrow \infty} 0$). As a result one outlier ruins estimated model. Hence, sample that are drawn from inliers produce geometrically more precise model, though size of sample should not be too big due to time complexity (in [8] suggested number is $\min(\frac{|Z|}{2}, 12)$ for homography and $\min(\frac{|Z|}{2}, 14)$ for epipolar geometry).

LO-RANSAC could be divided into Inner and Iterative parts which could be run both and separately:

7.1.1 Inner RANSAC

Inner phase executes when so-far-the-best model was found and there are enough inliers for non minimal estimation. Sample of predefined size are drawn (e.g. at random) from inliers consistent with so-far-the-best model and new model is estimated. If new model has bigger support than so-far-the-best model then new model is stored. This procedure repeats k times (e.g. $k = 10$ in [8]).

7.1.2 Iterative RANSAC

Iterative part may run independently as Inner RANSAC or after it.

Anyway, the first step is to find data points of the new threshold $\tau' := K\tau$, K is positive threshold multiplier. Those points could be viewed as (virtual) inliers consistent with so-far-the-best model and τ' . Then iteratively, new model is estimated with all inliers, and data points of new model and reduced threshold τ' are generating again. This procedure repeats until new threshold is not equal to original $\tau' \neq \tau$.

7.2 Fixing Locally Optimized RANSAC

Iterative RANSAC from previous section has one disadvantage that was discussed by Lebeda et al. in [18]

“For a large number of inliers, this procedure can dominate the execution time, even if executed only once.” [18]

7 Local Optimization

So, this is actually mean that all-inliers estimation in iterative part as was proposed in [8] is very time-consuming for big set of points. Furthermore those 'inliers' correspond to threshold $K\tau$ which means that for bigger K the procedure slows down too. The proposed solution in [18] is sampling from inliers inside Iterative part as well as in Inner RANSAC. The suggested in [18] sample size is $7m$ (m is minimal sample size).

This improvement sped up computational time several times, but on another hand the accuracy (not significantly) decreased, which is understandable, because estimation now is based on the subset of inliers.

7.3 Weighted Local optimization

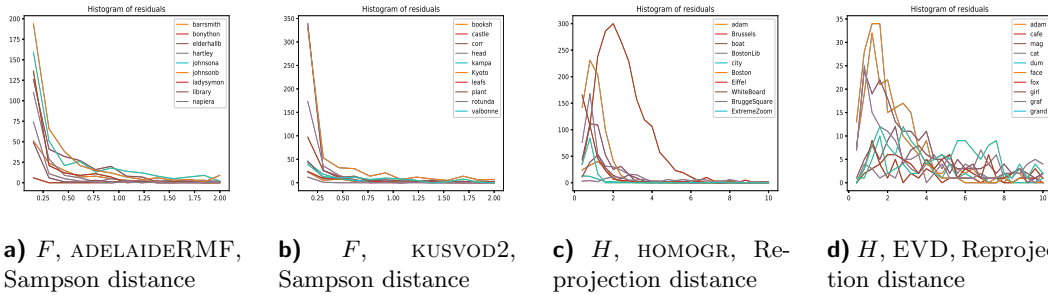


Figure 7.1 Histograms of point errors to the ground truth models. x -axis is threshold, y -axis is number of points in corresponded bins.

A lot of algorithms use weighting function to improve results. The weights usually seem like additional information that could improve estimation. For the vision problems it is very practical, because it is expected some noise or outliers in data. Intuitively, (correctly) weighted data points should refine estimation model.

In MAGSAC [3] by Barath et al. was proposed the σ -consensus algorithm to improve the *good* model of RANSAC by weighted least-squares. Weights of points are based on the likelihood of being inlier marginalised over the standard deviation of image noise σ .

In figures 7.1a - 7.1d are histograms of point errors. For fundamental matrix residuals (Sampson distance) of ground truth inliers are mostly in the error range $[0; 1]$ so the rest points out of this range are outliers. Similarly, for the homography matrix, the (reprojection) errors of inliers are within range $[0; 3]$. Since MAGSAC does not require inlier-outlier threshold, the outliers could be avoided by correct weighting using the right distribution of inlier residuals.

7.4 Graph-Cut RANSAC

7.4.1 Energy minimisation

Graph-Cut RANSAC proposed by Barath and Matas [2] was presented as the most geometrically accurate LO among state-of-the-art techniques. The idea was formulated as energy minimisation. Assume labeling L s.t. $\forall i \in 1, \dots, |\mathcal{P}| : L_i = 1$ if point i is inlier, 0 otherwise, $|\mathcal{P}|$ is number of points. For example, the binary loss is 1 if the classification (inlier-outlier) is wrong w.r.t. error distance ε and threshold τ , and loss is 0 if classification is correct. So far this definition reminds plain RANSAC, but recalling to MLESAC [31] the binary loss function was changed in GC-RANSAC to continuous

loss using Gaussian kernel $K(\varepsilon, \tau) = e^{-\frac{\varepsilon^2}{2\tau^2}}$. Then unary energy term $E_K(L)$ is:

$$E_K(L, \varepsilon, \tau) = \sum_{i=1}^{|\mathcal{P}|} \begin{cases} 1 - K(\varepsilon_i, \tau) & L_i = 1 \\ K(\varepsilon_i, \tau) & L_i = 0 \end{cases} \quad (7.1)$$

Remark, $(1 - K(\varepsilon, \tau)) \xrightarrow{\varepsilon \rightarrow \{0, \infty\}} \{0, 1\}$ and $K(\varepsilon, \tau) \xrightarrow{\varepsilon \rightarrow \{\infty, 0\}} \{0, 1\}$ and classification loss is similar to binary.

The energy minimisation could be more accurate by considering also the spatial coherence of points. In other words, it is assumed that close points are more likely belong to the same model. Therefore, considering the point proximities in the energy minimisation leads to improvement in accuracy.

In [2], Barath et al. modified the Potts model, which penalizes neighbors having different labels. The pair-wise energy term for each neighbor pair (edge) from neighborhood graph \mathcal{G} is

$$E_S(L) = \sum_{(p,q) \in \text{if}\mathcal{G}\mathcal{E}} \begin{cases} 1 & L_p \neq L_q \\ \frac{1}{2}(K(\varepsilon_p, \tau) + K(\varepsilon_q, \tau)) & \text{if } p, q \text{ are outliers} \\ 1 - \frac{1}{2}(K(\varepsilon_p, \tau) + K(\varepsilon_q, \tau)) & \text{if } p, q \text{ are inliers} \end{cases} \quad (7.2)$$

The total energy is calculated then

$$E(L) = E_K(L) + \lambda E_S(L), \quad (7.3)$$

where λ is spatial coherence parameter balancing the energy terms. The recommended value in [2] is 0.1 .

7.4.2 Graph Cut

The optimal labeling L^* that minimise 7.3 is found by using the source-target graph-cut algorithm. The construction of graph $\mathcal{G} = (\mathcal{P}, \mathcal{E})$ using unary and pair-wise energy terms is in details discussed in [2, 16]. Edges \mathcal{E} are represented by neighborhood system $\mathcal{N} \subseteq \mathcal{P} \times \mathcal{P}$ (\mathbf{x}_i has neighbors $\{\mathbf{x}' \mid \mathbf{x}' \in \mathcal{P}\}$).

7.4.3 Local Optimization

The local optimization part of GC-RANSAC can be briefly summarized in following pseudo code:

The sample size for estimation is $7m$ (suggested in [2]). The quality of model (function *better*) could be either standard binary or MSAC score or the recommended support function in [2] is based on kernel K

$$C(\theta) = \sum_{i=1}^{|\mathcal{P}|} K(\varepsilon_i, \tau), \quad \varepsilon_i \text{ is error distance from model } \theta \text{ to point } i. \quad (7.4)$$

Algorithm 3 GC-RANSAC

Input: $\hat{\theta}^*$ – so-far-the-best model of RANSAC, \mathcal{N} – neighborhood of points**Output:** Updated $\hat{\theta}^*$

```

1: changed  $\leftarrow$  1
2: while changed do
3:   changed  $\leftarrow$  0
4:    $\mathcal{G} \leftarrow \text{build}(\hat{\theta}^*, \mathcal{N})$  ▷ Build neighborhood graph.
5:    $L \leftarrow \text{graph\_cut}(\mathcal{G})$  ▷ Get optimal labeling using graph-cut algorithm.
6:   for  $i = 1, \dots, n$  do ▷  $n$  is maximum number of iterations
7:      $\mathcal{S} \leftarrow \text{sample}(L)$  ▷ Get sample from inliers using given labeling.
8:      $\hat{\theta}_{GC} \leftarrow \text{estimate}(\mathcal{S})$  ▷ Estimate new model.
9:     if better ( $\hat{\theta}_{GC}, \hat{\theta}^*$ ) then
10:        $\hat{\theta}^* \leftarrow \hat{\theta}_{GC}$  ▷ Update so-far-the-best model
11:       changed  $\leftarrow$  1

```

8 Termination criteria

Termination is essential part of every algorithm. In RANSAC stopping criteria are based on probabilities of finding a *good* model. We will describe in this section the termination conditions of different RANSAC-like algorithms. It is worth to note that termination is quite fragile phase, because any additional component or even small changes in algorithm structure could ruin the whole idea and assumptions of completion process.

8.1 RANSAC sampling

Termination criterion used in RANSAC is about confidence that model was estimated by inliers. In previous section we have shown that sampling is uniformly at random without replacement $\binom{N}{m}$, so probability to draw sample \mathcal{S} of size m from inliers of size $|\mathcal{I}|$ and total number of points N is

$$P_{\mathcal{S} \subseteq \mathcal{I}} = \frac{\binom{|\mathcal{I}|}{m}}{\binom{N}{m}} = \prod_{i=0}^{m-1} \frac{|\mathcal{I}| - i}{N - i} \quad (8.1)$$

In other words, $w = \frac{|\mathcal{I}|}{N}$ is inlier ratio. Roughly, probability of picking inlier at random is w . Hence, the joint probability that independently at random m points were chosen to be inliers is

$$P_{\mathcal{S} \subseteq \mathcal{I}} \approx \prod_{i=1}^m w = w^m \quad (8.2)$$

and $(1 - P_{\mathcal{S} \subseteq \mathcal{I}})$ is the odds that from m points at least one is outlier. Desirably, the probability $(1 - P_{\mathcal{S} \subseteq \mathcal{I}})$ should be as less as possible. Thus in k steps of random, independent and uniform sampling, the probability of having at least one outlier in sample of size m should be closed to zero $(1 - P_{\mathcal{S} \subseteq \mathcal{I}})^k < \eta \rightarrow 0\%$. Usually p is denoted as desired probability of getting useful result, so $1 - \eta = p \rightarrow 100\%$ and final equation is:

$$(1 - P_{\mathcal{S} \subseteq \mathcal{I}})^k = 1 - p \quad (8.3)$$

$$k^* = \frac{\log(1 - p)}{\log(1 - P_{\mathcal{S} \subseteq \mathcal{I}})} \quad (8.4)$$

Note, that $P_{\mathcal{S} \subseteq \mathcal{I}}$ is only lower bound probability of *good* sample, because real number of inliers $|\mathcal{I}^{(GT)}|$ is unknown. The used way to obtain tentative number of inliers is counting points under threshold. Another way to approximate probability of *good* sample is using some data distribution.

In [1], a relaxation of $P_{\mathcal{S} \subseteq \mathcal{I}}$ was proposed. Although, this method is more suitable for (P-)NAPSAC sampling, where we assume local structures and closed points are likely to be inliers. By positive constant number γ the RANSAC termination criterion can

be relaxed as follows

$$k^* = \frac{\log(1-p)}{\log(1-(w+\gamma)^m)}, \quad 0 \leq \gamma < 1-w \quad (8.5)$$

To the experiments in [1], a suitable value of γ^* is 0.1 leading to significantly fewer iterations with no noticeable deterioration in accuracy.

8.2 MAGSAC

Since MAGSAC [3] does not need an inlier-threshold, a new termination condition was proposed in [3]. Let us denote the expected number of samples to draw (see equations above) by RANSAC as $k(\tau)$ given a manually set threshold τ . Then number of iterations in MAGSAC is calculated marginalizing $k(\tau)$ over the noise level σ as follows:

$$k_{MAGSAC}^* = \frac{1}{\sigma_{max}} \int_0^{\sigma_{max}} k(\sigma) d\sigma \approx \frac{1}{\sigma_{max}} \sum_{i=1}^K \frac{(\sigma_i - \sigma_{i-1}) \log(1-p)}{\log(1 - \frac{|\mathcal{I}(\sigma_i)|}{|\mathcal{P}|})} \quad (8.6)$$

The integral can be replaced by the sum over the K smallest sigmas, $\sigma_1 \leq \sigma_2 \leq \dots \leq \sigma_K \leq \sigma_{max} \leq \dots \leq \sigma_{|\mathcal{P}|}$. However, to avoid the expensive computation over K smallest std. dev. of noise in [3] was suggested to uniformly divide interval $[\sigma_1; \sigma_{max}]$ into $d \ll K$ partitions. The used d in MAGSAC is 10.

8.3 PROSAC sampling

In sampling section was described procedure of choosing samples from subset of top-ranked points \mathcal{U}_l of size l . Let us recall also termination length l^* , that determines maximal range of sampling, denote \mathcal{I}_{l^*} as subset of inliers within \mathcal{U}_{l^*} . So, PROSAC terminates if two next conditions for $|\mathcal{I}_{l^*}|$ are satisfied:

8.3.1 Non-randomness

non-randomness – the probability that $|\mathcal{I}_{l^*}|$ out of l^* data points are by chance inliers to an arbitrary incorrect model is smaller than Φ (e.g. 5%) [6].

By the growth function, the subset \mathcal{U}_l of l points with the highest quality is progressively enlarged by points with lower probability of being inlier. This causes that sample by chance may consistent with outliers. The distribution that determines the size i of those top-ranked random "inliers" within \mathcal{U}_l is binomial (denote $P_l^{Rnd}(i)$) with parameter β – the probability of a random point being consistent with a contaminated model.

As a result the minimal size of inliers within \mathcal{U}_l is calculated so the probability of such size being random is smaller than threshold Φ .

$$\forall l : |\mathcal{I}_l^{min}| = \min \{j \mid \sum_{i=j}^l P_l^{Rnd}(i) < \Phi\} \quad (8.7)$$

Finally, the condition which is satisfied usually first is $|\mathcal{I}_{l^*}| \geq |\mathcal{I}_{l^*}^{min}|$

8.3.2 Maximality

maximality – the probability that a solution with more than $|\mathcal{I}_l^*|$ inliers in \mathcal{U}_l^* exists and was not found after k samples is smaller than η_0 (e.g. 5%) [6].

This criterion reminds RANSAC's termination condition, though difference is that sampling is not from all points but inside the range l

$$P_{S \subseteq \mathcal{I}_l} = \frac{\binom{|\mathcal{I}_l|}{m}}{\binom{l}{m}} \quad (8.8)$$

and similarly $(1 - P_{S \subseteq \mathcal{I}_l^*})^k$ must fall under threshold η_0 .

The stopping length l^* is chosen to maximise confidence solution subject to non-randomness constraint.

8.4 DEGENSAC

Let us recall F -estimation with 7-points algorithm. The probability that all 7 points are inliers or *good sample* is $P_{7/7} = w^7$ by RANSAC sampling. But some of the points could be H -degenerate and probability of inlier can not be simply predicted as inlier ratio. Chum et al. in [10] derived new formula of calculating this probability including fraction of homography consistent inliers w_H :

$$P_{7/7}^{(F)} = \sum_{i=0}^5 \binom{7}{i} w_H^i (w - w_H)^{7-i} \quad (8.9)$$

So, $P_{7/7}^{(F)} = P_{7/7}$ if $w_H = 0$, otherwise $P_{7/7}^{(F)} < P_{7/7}$. For now the confidence of finding good solution is even less than in RANSAC, but we could use plane-and-parallax algorithm to recover degenerate estimation. In this case at least 5 from 7 points must be H -degenerate, this probability is $P_{5/7}^{(H)} = \sum_{i=5}^7 \binom{7}{i} w_H^i (1 - w_H)^{7-i}$. However, probabilities $P_{5/7}^{(H)}$ and $P_{7/7}^{(F)}$ include correspondences that are both *good* and H -degenerate with odds $Pr_{7/7}^{(F)} \cap Pr_{5/7}^{(H)} = \binom{7}{5} w_H^5 (w - w_H)^2$. The final probability that epipolar geometry is recovered by drawing a single sample is:

$$P = P_{7/7}^{(F)} + P_{5/7}^{(H)} - (Pr_{7/7}^{(F)} \cap Pr_{5/7}^{(H)}) \quad (8.10)$$

Now, $P \geq P_{7/7}$ and DEGENSAC terminates earlier than RANSAC with the worst case when $w_H = 0$.

Nevertheless, authors of DEGENSAC computed probability $P_{7/7}$ to be confident in any solution.

8.5 SPRT

To decide if tested hypothesis is *bad* in [20] was introduced decision threshold A . The main properties of A is minimising average time of running, hence if A is too big time for evaluation would increase, on another side if A is small then probability of rejecting

a *good* model would rise. Denote, the time of evaluation dependent on A as

$$t(A) = \frac{1}{P_{S \subseteq \mathcal{I}}(1 - \alpha)} \left(t_{\Theta} + |\Theta_S| \frac{\log A}{C} \right) \quad (8.11)$$

Where $\alpha \approx \frac{1}{A}$ is the probability of rejecting *good* model and $\frac{\log A}{C}$ is average number of checked data points while testing a *bad* model. The optimal $A^* = \arg \min_{A'} t(A')$ minimises verification time. After derivation of equation above to reach minimum, exist two solutions and using numerical analysis the optimal A^* could be found (see [20]).

The termination condition for RANSAC with SPRT is "*The algorithm is terminated, when the probability of missing a set of inliers larger than the largest support found so far falls under a predefined threshold η_0 (e.g. 5%) [20]*". Recall the odds α of rejecting a *bad* model. For i -th sequence of SPRT the probability of α w.r.t. ϵ_i (probability that point is consistent with *good* model) and δ_i (probability that point is consistent with *good* model) is

$$\alpha_i = A^{-h_i}, \quad \epsilon \left(\frac{\delta_i}{\epsilon_i} \right)^{h_i} + (1 - \epsilon) \left(\frac{1 - \delta_i}{1 - \epsilon_i} \right)^{h_i} = 1, \quad \epsilon_i < \epsilon \approx \frac{|\mathcal{I}|}{N} \quad (8.12)$$

The solutions of second equation above are two, the first one is for $h_i = 0$ and second could be found numerically. The probability of having at least one outlier and not rejecting a *good* model in k steps is

$$\eta_i = (1 - P_{S \subseteq \mathcal{I}}(1 - \alpha_i))^{k_i} \quad (8.13)$$

It is worth to notice, that in standard RANSAC α is zero.

Denote k_i number of samples processed during i -th likelihood ratio test from sequence of r independent tests in RANSAC. Then the probability η is given

$$\eta(r) = \prod_{i=1}^r \eta_i \quad (8.14)$$

And upper bound number of iteration for r -th test is

$$k_r^* = \frac{\log \eta_0 - \log(\eta(r-1))}{\log\left(1 - \frac{P_{S \subseteq \mathcal{I}}}{A_r}\right)} \quad (8.15)$$

9 Neighbors Searching

In the previous sections we note that spatial coherence and as a result neighborhood of points has influence on estimation problems. For instance, P-NAPSAC and NAPSAC outstrips RANSAC when inliers are closed by assumption. Or GC-RANSAC is better than LO-RANSAC because by one of the reasons that cooperates with neighbors of points.

There is three options to obtain neighborhood

9.1 Radius Search

Points within hypersphere of radius r centered in target point \mathbf{x} are neighbors of \mathbf{x} . This is the standard approach to find neighborhood of point, which also suits for NAPSAC sampling.

9.2 K Nearest Neighbors Search

The implementation benefit of KNN over Radius-Search is the fixed number of neighbors for every point. However the neighbors may not be adjacent, moreover could be even fairly distant from the target point.

9.3 Grid Search

Non-trivial implementation of previous methods requires construction of k -dimensional trees, but for two-view $A - H - F - E$ estimation problems it is unnecessary. The point could be viewed as four-dimensional after concatenating of two 2D corresponded points.

The advanced method of searching neighbors of correspondent points is tiling the data set. Suppose, we have (uniformly) divided grids of two images then

Two points are neighbors if they are in the same cell in both correspondent images.

Using hash tables this implementation is much simpler and faster than KNN or Radius-Search. The complexity is linear $\mathcal{O}(N)$ for data size N , while k - d tree has $\mathcal{O}(N \log N)$ complexity.

10 Implementation

USAC++ framework is written in C++ language. Each component (sampling, local optimization, termination criteria etc.) has an abstract class. Thus new parts could be easily added to the framework. In the implementation of framework for the sake of efficiency was avoided using complex data types and large usage of memory. The main used library is OpenCV. The minor libraries are Graph Cut Optimization (GCO) [16], Nanoflann (for k -nearest neighbors search) and OpenMP (for parallel evaluation).

By having an abstract score function RANSAC is possible to change e.g., to Least Median Squares estimator so the score is median of residuals.

The main parts of USAC++ framework are RANSAC, PROSAC, NAPSAC, P-NAPSAC sampling. The local optimisation methods are Inner RANSAC, (Fixing) Iterative RANSAC, GC-RANSAC. The implemented abstract solvers are for homography, fundamental, essential, affine matrices and line in 2D estimation. The abstract quality and score classes are binary-RANSAC, MSAC, MLESAC and MAGSAC. The termination criteria are for RANSAC, PROSAC and MAGSAC, for DEGENSAC in [10] was used same termination condition as for RANSAC. There are also many other things in the framework such as degeneracy test and plane-and-parallax algorithm for fundamental matrix or oriented epipolar constraint which benefits was discussed by Chum et al. in [9].

10.1 Randomness

For RANSAC is very important to be random, but standard random functions are not reliable, hence we implemented pseudo-random Fisher-Yates shuffle. This method generate uniformly distributed unbiased permutation and unique random subset is easily extracted. Although, it requires allocation of array of points size.

Similarly, shuffled array is using for randomized evaluation e.g. for RANSAC with sequential probability ratio test.

in the figure 10.1 is histogram of values from C++ function `random()`. The total number of values equals to the random range, so desirably each bin should have only one value (red line). The entropy of `random()` for tested data size is 9.34 and maximum entropy is 9.90.

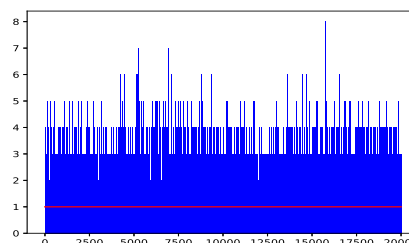


Figure 10.1 Histogram of random values.

10.2 Estimation

The common step to obtain any of $A-H-F-E$ matrices is solving the set of equations generated from sample. However, overdetermination or complication of equations leads to the optimisation problem. Denote matrix $L \in \mathbb{R}^{m \times n}$ ($L' \in \mathbb{R}^{m \times n'}$) that represents

desired mapping to obtain the model θ , and $b \in \mathbb{R}^n$ is a vector of given values. Then minimisation task is

$$\theta^* = \arg \min_{\theta} \|L\theta - b\|_2 \quad (10.1)$$

Or

$$\theta^* = \arg \min_{\theta'} \|L'\theta'\|_2, \text{ s.t. } \|\theta'\|_2 = 1 \quad (10.2)$$

There are similar approaches to find the optimal solution, although each of them lead to different result w.r.t. computational time and accuracy (see experiments).

1. Least Squares

It could be shown that solution of 10.1 that minimises the sum of residuals is

$$\theta^* = (L^\top L)^{-1} L^\top b \quad (\text{assume } L \text{ has full column rank})$$

However numerically the inner product is expensive and inaccurate.

2. Principal Component Analysis

In 10.2 θ' minimise perpendicular distances. The optimal solution could be found using PCA. Then θ^* is eigen vector corresponded to the highest eigen value of $L'^\top L'$. Similarly the inner product is still necessary to calculate.

3. Singular Value Decomposition

SVD is very popular numerical method for nullspace extraction and as a result could be used to solve 10.2. The optimal eigen vector from PCA could be decomposed from L' without explicit calculation of $L'^\top L'$.

4. QR factorisation

QR factorisation is universal method that solves 10.1 and 10.2. The matrix L could be decomposed to orthogonal $Q \in \mathbb{R}^{m \times m}$ and upper triangular $R \in \mathbb{R}^{m \times n}$.

The solution of least squares problem then is $\theta^* = R^{-1} Q^\top b$

Let the $Q' \in \mathbb{R}^{n' \times n'}$ is orthogonal matrix obtained by QR-decomposition of L'^\top . Suppose L' has rank r . Then the last $n' - r$ columns are orthogonal nullspace of L' and the optimal θ^* could be extracted from them.

In [24] was suggested to use QR decomposition for epipolar geometry estimation.

10.3 Sorting points

In PROSAC [6] the used ordering of correspondences was based on SIFT descriptors[19]. The top-ranked points have the smallest ratio of distances of the best and second match. Basically, the ordering is about confidence that two detected points are really correspond to each other. However, such information of any feature detector is often remain unknown.

Another way to order points by their quality is dense sorting. It does not require knowledge of descriptor's scores but assumption that closed points are detected correct. Denote $\mathcal{N}_{\mathbf{x},r}, \mathcal{N}'_{\mathbf{x}',r}$ as the closest neighborhood of corresponded points \mathbf{x}, \mathbf{x}' within radius r . Intuitively, if all neighbors of \mathbf{x} correspond to neighbors of \mathbf{x}' then these two points and their neighbors are very likely to be correctly matched. For bigger neighborhood this likelihood is increasing, unless all correspondences inside neighborhood are mismatched. Then ordering of points based on feature density could be for instance, following:

$$i < j \Rightarrow |\mathcal{N}_{\mathbf{x}_i,r} \cap \mathcal{N}'_{\mathbf{x}'_i,r}| \geq |\mathcal{N}_{\mathbf{x}_j,r} \cap \mathcal{N}'_{\mathbf{x}'_j,r}| \quad (10.3)$$

10 Implementation

Point \mathbf{x}_i has higher quality if the number of corresponded neighbors of $[\mathbf{x}\mathbf{x}']_i$ is bigger than $[\mathbf{x}\mathbf{x}']_j$'s. Or density of correspondences could be measured in four dimensional subspace (concatenation of two 2D points). Then concatenated point has bigger quality, if sum of distances to the k closest neighbors is smaller.

$$i < j \Rightarrow \sum_{t=1}^k \|\mathbf{x}_i \mathbf{x}'_i - [\mathbf{x}_{i,t} \mathbf{x}'_{i,t}]\|_2 < \sum_{t=1}^k \|\mathbf{x}_j \mathbf{x}'_j - [\mathbf{x}_{j,t} \mathbf{x}'_{j,t}]\|_2 \quad (10.4)$$

The biggest disadvantage of proposed sorting is that points without the closest neighbors may be wrongly considered as incorrect matches.

It worth to note that presented sort is also compatible with NAPSAC assumption that inliers are tend to be closer. The inliers have in this case bigger neighborhood rather than outlier hence a bigger quality. In the figure 1.1a or 4.1 the inliers distributed closer than outliers. PROSAC with density sort found correct line significantly faster than RANSAC or NAPSAC.

11 Experiments

11.1 Dense sort

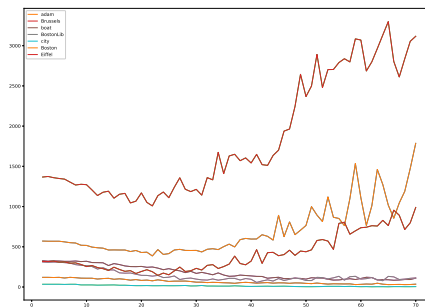
To evaluate points sort by their density the neighborhood of points must be obtained. It could be efficiently done with Grid search. The main parameter is the size of cell in images. To find the optimal cell size was randomly chosen different image pairs with ground truth annotation. The loss function is based on the number of same neighbors within cell size of point on the first and second correspondent images. The loss is 1 if point is correct match but has not closed neighbors. If the match is incorrect then loss is number of the nearest neighbors. The total loss L of assumption that *better* point has bigger same neighborhood of the correspondent pair is

$$L(\mathcal{P}, \mathcal{N}, r) = \sum_{(\mathbf{x}, \mathbf{x}') \in \mathcal{P}} \begin{cases} 1 & |\mathcal{N}_{\mathbf{x}, r} \cap \mathcal{N}'_{\mathbf{x}', r}| = 0 \wedge (\mathbf{x}, \mathbf{x}') \text{ is correct match} \\ |\mathcal{N}_{\mathbf{x}, r} \cap \mathcal{N}'_{\mathbf{x}', r}| & |\mathcal{N}_{\mathbf{x}, r} \cap \mathcal{N}'_{\mathbf{x}', r}| \geq 1 \wedge (\mathbf{x}, \mathbf{x}') \text{ is incorrect match} \\ 0 & \text{otherwise} \end{cases} \quad (11.1)$$

$\mathcal{N}_{\mathbf{x}, r}$ is neighborhood of point \mathbf{x} from points set \mathcal{P} of radius r (cell size). $|\mathcal{N}_{\mathbf{x}, r} \cap \mathcal{N}'_{\mathbf{x}', r}|$ is the number of same neighbors of \mathbf{x} and \mathbf{x}' .

The optimal cell size is 42. The figure 11.1a is example how loss is changing to the cell size.

In the figure 11.1b is example of sort of one image pair. The green points are correct matches, red – incorrect. The first line is classification of top-ranked points using SIFT score. The next lines show points classification by density of the different cell size.



a) Optimal cell size



b) Classification of points by SIFT and Dense sort

Figure 11.1 Dense sort

11.2 Model Estimation

For estimation the geometric relation (e.g., homography or epipolar geometry) the least-squares problem must be solved. The minimisation of perpendicular distances 10.2 is using more often than vertical ones 10.1. The task then is to find the optimal nullspace. This could be done by PCA, SVD or QR. To determine which method is better we evaluated them w.r.t. error and time on different data size (see table 11.1). The QR decomposition, not surprisingly, is more accurate for epipolar geometry for minimal sample size. However, the PCA for non minimal estimation is much faster and a bit more accurate than SVD or QR and furthermore use much less memory allocation. The estimation of minimal sample size could be done by SVD for homography and QR for epipolar geometry.

To obtain (full) SVD and PCA was used OpenCV library, for (full) QR-decomposition was used Householder transformation from Eigen library.

			PCA	QRf	SVDf
min size	H	\mathcal{T}	12.4158 \pm 5.05	106.1188 \pm 22.06	10.8416 \pm 3.53
		\mathcal{E}	0.0142 \pm 0.09	0.0000 \pm 0.00	0.0000 \pm 0.00
	F	\mathcal{T}	14.8020 \pm 4.54	86.4059 \pm 8.26	10.0495 \pm 0.43
		\mathcal{E}	0.3418 \pm 0.72	0.0001 \pm 0.00	0.0004 \pm 0.00
	E	\mathcal{T}	16.1485 \pm 5.65	64.4356 \pm 18.12	10.4356 \pm 2.51
		\mathcal{E}	0.2051 \pm 0.43	0.0002 \pm 0.00	0.0011 \pm 0.00
size=20	H	\mathcal{T}	25.9109 \pm 4.33	313.5049 \pm 29.99	30.8218 \pm 4.01
		\mathcal{E}	0.3477 \pm 0.13	2.3733 \pm 5.32	0.4372 \pm 0.13
	F, E	\mathcal{T}	15.0297 \pm 6.31	235.52 \pm 14.88	33.0099 \pm 4.42
		\mathcal{E}	0.0442 \pm 0.26	1.2198 \pm 5.18	0.0068 \pm 0.00
size=1500	H	\mathcal{T}	139.6931 \pm 2.68	11556.5938 \pm 333.79	94160.5781 \pm 5631.14
		\mathcal{E}	4.3206 \pm 0.06	24.7286 \pm 35.58	5.0085 \pm 0.45
	F, E	\mathcal{T}	85.0495 \pm 2.74	4939.7524 \pm 161.87	12030.0986 \pm 5339.82
		\mathcal{E}	0.0813 \pm 0.00	2.6306 \pm 4.26	0.0805 \pm 0.0014

Table 11.1 The nullspace estimation time and error for homography matrix and epipolar geometry. 100 random samples of minimal size (4 – homography, 7 – fundamental, 5 – essential matrix), size of 20 and 1500 were used for estimation. \mathcal{T} is average estimation time (microseconds). \mathcal{E} is average estimation error. The error from 10.2 is sum of perpendicular distances of estimated nullspace to data matrix. For minimal sample size for epipolar geometry the error is average of sums, because for fundamental and essential matrix two and four resp. nullspaces were calculated.

11.3 All

The RANSAC, PROSAC, fixing locally optimised RANSAC, Graph-Cut RANSAC, Progressive NAPSAC and RANSAC with SPRT were tested on homography and epipolar geometry estimation problems. The data images were taken from publicly available real-world datasets: Extreme View Dataset EVD [21], HOMOGR and KUSVOD2 from Lebeda [17], ADELAIDERMF from Wong and STRECHA [29].

All optimal parameters for algorithms that were discussed in previous sections were used in experiments. The evaluation of algorithms was unbiased and equal. The degeneracy test and epipolar constraint were applied for epipolar geometry estimation. The test on collinearity was applied for homography estimation. The threshold for each

algorithm was set to 2 pixels.

The quality score for PROSAC is the ratio of the first and second match based on SIFT detector and FGINN ratio [21] for EVD dataset. PROSAC with sorted points by density was also tested.

The most accurate algorithm from the tested ones is GC-RANSAC, the average error is much less on every tested dataset on each estimation problem.

PROSAC is the fastest algorithm almost on all datasets. However the error especially for STRECHA dataset is quite big, which could be explained by sensitivity to the given scores. Similarly, the density sort does not need to be suitable for all images, it is easy to shown that for images with distant points such sorting would fail.

The cause of big error for EVD is very small inlier ratio in some image pairs. The maximum limit of iterations was set to 5000, which is too small for some pairs.

For RANSAC with SPRT is important not to overestimate the initial probabilities δ_0 and ϵ_0 (belonging point to *bad* and *good* model resp.). To avoid this problem SPRT starts verification only after the fixed number of drawn samples so the initial parameters δ_0, ϵ_0 could be approximated.

			Confidence 95%						
			RSC	PSC	PSCd	FLO	SPRT	GC	P-NSC
1, #14	F	$\bar{\mathcal{E}}$	2.66	3.49	5.36	1.29	3.01	0.25	4.03
		$\bar{\mathcal{T}}$	8.66	1.38	0.49	7.77	3.8	6.63	5.69
		$\bar{\mathcal{S}}$	166.84	18	9.92	138.38	150.23	96.46	121.8
		$\bar{\mathcal{F}}$	0	0	0	0	framework for 0	0	0
2, #15	F	$\bar{\mathcal{E}}$	1.11	2.28	4.2	1.15	1.09	1.04	1.33
		$\bar{\mathcal{T}}$	1.55	4.81	8.17	1.16	1.1	2.13	2.57
		$\bar{\mathcal{S}}$	30.7	358.29	358.35	19.58	36.11	21.52	32.76
		$\bar{\mathcal{F}}$	0	0	0	0	0	0	0
3, #12	H	$\bar{\mathcal{E}}$	1.29	1.16	2.54	1.19	1.29	1.07	1.08
		$\bar{\mathcal{T}}$	1.05	0.28	0.24	1.27	0.66	2.4	1.94
		$\bar{\mathcal{S}}$	53	7.25	6.83	33.79	44.79	35.25	25.91
		$\bar{\mathcal{F}}$	0	0	8.33	0	0	0	0
4, #15	H	$\bar{\mathcal{E}}$	50.6	49.91	47.79	49.28	59.09	45.6	52.45
		$\bar{\mathcal{T}}$	74.23	15.94	27.17	71.69	20.75	74.23	71.59
		$\bar{\mathcal{S}}$	3457.66	827.33	1157.7	3254.73	3595.6	3287.13	2900.33
		$\bar{\mathcal{F}}$	29.73	15.53	23.46	21.6	46.4	22.06	28.73
5, #20	E	$\bar{\mathcal{E}}$	2.82	12.19	7.36	0.73	2.74	0.3	2.52
		$\bar{\mathcal{T}}$	33.97	10.34	8.26	18.21	19.5	20.67	23.89
		$\bar{\mathcal{S}}$	60.75	19.45	15.65	28.7	51.8	29.55	36.25
		$\bar{\mathcal{F}}$	0	12.75	11.75	0	0	0	0

Table 11.2 The tested datasets are 1 – KUSVOD2, 2 – ADELAIDERMF, 3 – HOMOGR, 4 – EVD and 5 – STRECHA. # is the number of image pairs tested. The algorithms RSC – RANSAC, PSC – PROSAC with SIFT ratio score with and FGINN ratio score for EVD dataset, PSCd – PROSAC with dense sort, FLO – fixing Locally Optimized RANSAC, SPRT – RANSAC with sequential probability ratio test, GC – Graph-Cut RANSAC. The score function for each method is truncated error distance (MSAC score). Each image pair was run 100 times. $\bar{\mathcal{E}}$ is average error (px) of all image pairs average errors to ground truth inliers in dataset. Similarly $\bar{\mathcal{T}}$ is average time (milliseconds), $\bar{\mathcal{S}}$ is average number of main RANSAC drawn samples, $\bar{\mathcal{F}}$ is average number of fails.



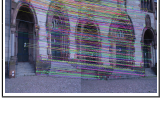
		Confidence 95%						
		RSC	PSC	PSCd	FLO	SPRT	GC	P-NSC
	$\bar{\mathcal{E}}$	1.3 ± 0.0	1.4 ± 0.4	1.6 ± 0.2	0.88 ± 0.3	1.2 ± 0.3	0.8 ± 0.4	0.8 ± 0.2
	$\bar{\mathcal{T}}$	0.9 ± 0.2	0.5 ± 0.1	0.8 ± 0.2	1.6 ± 0.5	1.1 ± 0.15	2.8 ± 0.7	3.1 ± 0.3
	$\bar{\mathcal{S}}$	34.2 ± 1.0	14.8 ± 1.1	23.2 ± 6.2	30.3 ± 5.2	69.9 ± 5.7	29.9 ± 9.5	19.1 ± 6.1
	$\bar{\mathcal{E}}$	2.5 ± 2.1	7.9 ± 5.1	9.8 ± 6.7	1.8 ± 1.7	6.5 ± 6.3	0.7 ± 0.2	4.2 ± 2.3
	$\bar{\mathcal{T}}$	7.1 ± 3.9	0.8 ± 0.2	0.9 ± 0.4	6.2 ± 1.6	5.1 ± 1.7	7.8 ± 1.7	8.1 ± 3.3
	$\bar{\mathcal{S}}$	113.7 ± 65.2	5.6 ± 2.7	4.6 ± 2.9	59.6 ± 46.3	229 ± 123	37.8 ± 26.8	69.7 ± 39.6
	$\bar{\mathcal{E}}$	2.3 ± 0.6	4.3 ± 0.9	3.1 ± 0.9	1.3 ± 1.1	2.4 ± 0.6	0.35 ± 0	2.4 ± 0.6
	$\bar{\mathcal{T}}$	17.8 ± 7.9	4.9 ± 2.1	4.6 ± 1.1	16.7 ± 5.9	13.2 ± 5.2	18.3 ± 5.5	13.7 ± 5.2
	$\bar{\mathcal{S}}$	39.7 ± 17.2	12.1 ± 5.33	10 ± 2.2	33.5 ± 13	36.6 ± 19.8	33.3 ± 29.5	26.4 ± 13.5

Table 11.3 Similarly to previous table, some randomly chosen correspondent pairs from HO-MOGR (H), KUSVOD2 (F) and STRECHA (E) datasets were tested on the same algorithms. The number of runs for each image pair is 1000.

12 Conclusions

The universal framework for Random Sample Consensus USAC++ consists of following algorithms: RANSAC, MSAC, MLESAC, DEGENSAC, PROSAC, (F)LO-RANSAC, (P-)NAPSAC, MAGSAC, GC-RANSAC, RANSAC with SPRT.

PROSAC – is so-far the most efficient algorithm that should be used when quality score for each data point is given. For image matching problem the recommended quality is ratio of the best and the second match of descriptors. However, the such score is often unknown, but could be approximated using the density of features.

In the presence of noise, the model estimated from minimal sample, has poor quality. The Locally Optimised RANSAC significantly improves the so-far-the-best model of RANSAC using non minimal sample estimation.

GC-RANSAC is so-far the most accurate method that considering the spatial coherence of points minimise energy loss by s-t graph-cut algorithms. By experiments it has the smallest errors among other algorithms.

If the inliers are distributed closely and the outliers are not, then the local structures with high inlier ratio can exist. The benefit of NAPSAC sampling is in finding *good* local structures earlier than by global-RANSAC sampling. However, the global sampling also has *good* properties for some estimation problems. NAPSAC with progressively growing neighborhood is an optimal choice preserving the benefits of local and global sampling. The efficient Grid neighbors search for the two view correspondence problem could be used to find the neighborhood of points.

The verification steps save a lot of time required for model evaluation, thus sequentially probability ratio test, collinearity tests, degeneracy tests and oriented constraints for epipolar geometry are included in the framework.

The score function plays important role in model evaluation. The binary-RANSAC score is not accurate. MSAC that use truncated error distance or MLESAC considering the likelihood of error improve the estimation.

Inlier/outlier threshold is important input parameter for all previous algorithms, whereas for MAGSAC threshold is not required. The method use weighted least squares to find estimation based on the likelihood of point being inlier.

The framework USAC++ was evaluated in terms of speed and accuracy on some two-view geometry estimation problems on different annotated real-world datasets. The most accurate result provide GC-RANSAC and FLO-RANSAC, the fastest algorithms are PROSAC and RANSAC with SPRT. The RANSAC-based algorithms were thoroughly implemented in C++. The main parts of RANSAC such as i.e, sampling or evaluation has abstract class in implementation so framework could be extended by adding new algorithms.

The robust estimation is an important open problem. Different algorithms use different techniques based on some assumptions to provide good estimation. Individually, not each method could desirably with respect to time and accuracy solve all types of estimation problems. The proposed universal framework USAC++ includes so-far-the most used, accurate and efficient methods that could be used for various estimation tasks.

Bibliography

- [1] D. Baráth, M. Ivashechkin, and J. Matas. Progressive napsac: sampling from gradually growing neighborhood. 2019. 1, 2, 11, 27, 28
- [2] D. Baráth and J. Matas. Graph-cut ransac. *IEEE CVF Conference on Computer Vision and Pattern Recognition*, pages 6733–6741, 2018. 2, 24, 25
- [3] D. Baráth, J. Noskova, and J. Matas. Magsac: marginalizing sample consensus. 06 2019. 2, 18, 24, 28
- [4] O. Chum. Two-view geometry estimation by random sample and consensus. PhD dissertation. Czech Technical University in Prague, 2005. 19
- [5] O. Chum and J. Matas. Randomized ransac with t d,d test. In *Image and Vision computing*, pages 448–457, 2002. 1, 16
- [6] O. Chum and J. Matas. Matching with PROSAC-progressive sample consensus. In *Computer Vision and Pattern Recognition*. IEEE, 2005. 1, 9, 28, 29, 33
- [7] O. Chum and J. Matas. Optimal randomized ransac. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(8):1472–1482, August 2008. 17
- [8] O. Chum, J. Matas, and J. Kittler. Locally optimized ransac. In *Joint Pattern Recognition Symposium*. Springer, 2003. 1, 23, 24
- [9] O. Chum, T. Werner, and J. Matas. Epipolar geometry estimation via ransac benefits from the oriented epipolar constraint. In *International Conference on Pattern Recognition*, 2004. 32
- [10] O. Chum, T. Werner, and J. Matas. Two-view geometry estimation unaffected by a dominant plane. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 772–779, 2005. 2, 20, 21, 29, 32
- [11] 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*, 1981. 1, 9
- [12] V. Fragoso, P. Sen, S. Rodriguez, and M. Turk. Evsac: Accelerating hypotheses generation by modeling matching scores with extreme value theory. In *2013 IEEE International Conference on Computer Vision (ICCV)*, pages 2472–2479, 2013. 1
- [13] J. M. Frahm and M. Pollefeys. RANSAC for (quasi-)degenerate data (QDEGSAC). In *CVPR (1)*, pages 453–460. IEEE Computer Society, 2006. 2, 19, 21
- [14] D. Ghosh and N. Kaabouch. A survey on image mosaicing techniques. *J. Vis. Comun. Image Represent.*, pages 1–11, 2016. 1
- [15] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 7, 8, 13, 19, 20, 21

- [16] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? In *Proceedings of the 7th European Conference on Computer Vision-Part III*, ECCV '02, pages 65–81, Berlin, Heidelberg, 2002. Springer-Verlag. 25, 32
- [17] K. Lebeda. Robust sample consensus. In Master Thesis. Czech Technical University in Prague, 2013. 36
- [18] K. Lebeda, J. Matas, and O. Chum. Fixing the locally optimized ransac. In *British Machine Vision Conference*. Citeseer, 2012. 23, 24
- [19] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004. 33
- [20] J. Matas and O. Chum. Randomized ransac with sequential probability ratio test. *International Conference on Computer Vision*, page 1727–1732, 2005. 1, 16, 17, 29, 30
- [21] D. Mishkin, J. Matas, and M. Perdoch. MODS: Fast and robust method for two-view matching. *Computer Vision and Image Understanding*, 2015. 36, 37
- [22] D. R. Myatt, P. H. S. Torr, S. J. Nasuto, J. M. Bishop, and R. Craddock. Napsac: high noise, high dimensional robust estimation. In *In BMVC02*, pages 458–467, 2002. 1, 10, 11
- [23] K. Ni, H. Jin, and F. Dellaert. Groupsac: Efficient consensus in the presence of groupings. *2009 IEEE 12th International Conference on Computer Vision*, pages 2193–2200, 2009. 11, 12
- [24] D. Nistér. An efficient solution to the five-point relative pose problem. *Transactions on Pattern Analysis and Machine Intelligence*, pages 756–770, 2004. 8, 33
- [25] D. Nistér. Preemptive ransac for live structure and motion estimation. In *ICCV*, pages 199–206. IEEE Computer Society, 2003. 17
- [26] D. P. Capel. An effective bail-out test for ransac consensus scoring. 01 2005. 1, 16
- [27] P. Pritchett and A. Zisserman. Wide baseline stereo matching. In *Proceedings of the Sixth International Conference on Computer Vision*, ICCV '98. IEEE Computer Society, 1998. 1
- [28] 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
- [29] 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. 36
- [30] P. H. S. Torr and D. W. Murray. Outlier detection and motion segmentation. pages 432–443, 1995. 1
- [31] P. H. S. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 2000. 2, 14, 15, 18, 24

- [32] P.H.S. Torr and A. Zisserman. Robust computation and parametrization of multiple view relations. In *ICCV*, pages 727–732, 1998. 2, 14
- [33] P.H.S. Torr, A Zisserman, and S.J. Maybank. Robust detection of degenerate configurations for the fundamental matrix. In *IEEE International Conference on Computer Vision*, pages 1037 – 1042, 07 1995. 20

13 CD content

```
Ransac
├── usac
│   ├── estimator
│   ├── sampler
│   ├── degeneracy
│   ├── local_optimization
│   ├── quality
│   ├── ransac
│   ├── termination_criteria
│   ├── utils
│   │   ├── math
│   │   └── neighbors_search
│   └── random_generator
├── dataset
│   ├── homography
│   ├── adelaidermf
│   ├── EVD
│   ├── kusvod2
│   └── strecha
├── detector
├── reader
├── generator
├── helper
├── include
│   └── gco; nanoflann
├── test
└── thesis
```