HAL open science

Automatic Threshold RanSaC Algorithms for Pose Estimation Tasks

Clément Riu, Vincent Nozick, Pascal Monasse

To cite this version:

Clément Riu, Vincent Nozick, Pascal Monasse. Automatic Threshold RanSaC Algorithms for Pose Estimation Tasks. Communications in Computer and Information Science, 2023, Communications in Computer and Information Science, 1815 (1815), pp.1-20. 10.1007/978-3-031-45725-8_1 . hal-04260593

HAL Id: hal-04260593

https://hal.science/hal-04260593v1

Submitted on 10 Jan 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Automatic Threshold RanSaC Algorithms for Pose Estimation Tasks

Clément Riu[0000−0002−5447−6149], Vincent Nozick[0000−0001−8792−7164], and Pascal Monasse[0000−0001−9167−7882]

Université Paris-Est, LIGM (UMR CNRS 8049), UGE, ENPC, F-77455 Marne-la-Vallée, France {clement.riu,pascal.monasse}@enpc.fr, vincent.nozick@univ-eiffel.fr

Abstract. When faced with data corrupted by both noise and outliers robust estimation algorithms like RanSaC are used, especially in the field of Multi-View Stereo (MVS) or Structure-from-Motion (SfM). To find the best model fitting the data, from numerous minimal samples it evaluates models and rates them according to their number of inliers. The classification as inlier depends on a user-set threshold that should be tailored to the noise level of the data. This requires the knowledge of this level, which is rarely available. The few existing adaptive threshold algorithms solve this problem by estimating the value of the threshold while computing the best model. However, it is hard to obtain ground-truth for MVS and SfM tasks and usually test datasets are based on the output on some state of the art algorithm, which prevents the objective evaluation of new algorithms. We propose a new method to generate artificial datasets based on true data to get realistic and measurable results. We use this method to benchmark different automatic RanSaC algorithms and find out how they compare to each other and identify each algorithm's strengths and weaknesses. This study reveals uses cases for each method and the possible tradeoffs between performance and execution time.

Keywords: Perspective from n Points, Multi-View Stereo, Structure-From-Motion, RanSaC, Semi-synthetic Dataset, Benchmark

1 Introduction

To fit a model to noisy data in the presence of outliers, a standard regression method can easily fail and the RANdom SAmple Consensus algorithm (RanSaC) [9] was proposed as a solution. This algorithm try to discriminate inliers from outliers and estimate a model simultaneously. To do so, it compares residuals of datapoints to a user-defined threshold given a model. This model is estimated from a random minimal sample of data and gets scored based on the number of inliers it yields, its consensus. The higher this consensus, the most likely the right model has been selected. However, this method can fail if the outlier ratio is too high and most importantly, it requires *a priori* knowledge of

the noise level of the inliers, or at least a good estimate of this value. Methods we call *adaptive*, *automatic* or *threshold-free* propose a solution to this issue by changing the quality criteria of models to some other measure that does not depend only on the inlier number and can estimate the inlier/outlier threshold.

Our first contribution is a semi-artificial data generation methodology that creates realistic and controlled data to precisely compare algorithms and see strengths and weaknesses of each. Usually, novel algorithms test on both real world data and artificial toy problems. However, in Multi-View Stereo (MVS) and Structure-from-Motion (SfM) it is expensive to get ground-truth and often relies on some algorithm that might introduce uncontrolled noise or bias. For instance, the KITTI [10] and the 7 scenes datasets [33] require calibrating an active sensor (LiDAR) to the passive one (camera). Another example is [6] that uses uncorrected matches as inliers. Artificial settings like the one proposed in [5], a plane estimation in 3D space, or [1], a random homography estimation between two 2D spaces, lack realism and do not give enough confidence in the generalization power of these experiments. Our solution presented in Section 5 answers that issue by using artificial matches based on real data.

Our second contribution is an extensive benchmark of different state-of-the-art adaptive RanSaC methods across different tasks. Through this benchmark we aim at comparing the different algorithms in a single setting; all algorithm have been integrated in a common pipeline so that the impact of the implementation of other elements can be removed. Contrary to previous RanSaC comparative studies like [4] we focus on adaptive methods and use our novel data generation technique to get more accurate results. It is also used to test the quality of the data generation methodology by testing it across multiple source datasets and comparing change and similarities in behavior depending on the setting.

RanSaC algorithms can be used in a variety of tasks but we focus our experiments around Multi-View Stereo and Structure-From-Motion tasks, namely homography estimation, fundamental and essential matrix fitting and the Perspective from n Points (PnP) problem. Those tasks are the core steps in 3D reconstruction pipelines, like in [32, 25], and as such our benchmark compares the performance on these tasks. Improving the RanSaC algorithm should not only be able to provide a better model, but also better inliers, which are useful for the pipeline overall quality. Finally, datasets like [15] used to train 3D deep-learning estimators are created using those traditional methods, and thus improving the quality of them could help improving neural networks training.

This paper is an extension of a conference paper [31]. Compared to the latter, we introduced a new test case, the PnP problem. As we wrote, extending our method to this problem is the necessary next step to see whether we can improve software like ColMap [32] that is still state-of-the-art. We also propose a new version of Fast-AC-RanSaC, an automatic RanSaC algorithm that was used in the first paper and deemed unsatisfactory. We added the recent MAGSAC++ [2] to the benchmark, an improvement over MAGSAC [1] that was included in the first paper. For both MAGSAC and MAGSAC++, we also integrated their implementation in the same framework as all others, mainly to use more stable

model estimators. Indeed, our previous benchmark indicated a high instability in MAGSAC result, with failure to estimate good models even in some easy cases, due to the numerical instability of the minimal solvers chosen by the authors.

Section 2 reviews the automatic RanSaC methods and those included in the benchmark are detailed in Section 3, Section 4 presents the data generation methodology and how to apply it to the different test cases, Section 5 presents the benchmark data, parameters, results and analysis and Section 6 concludes.

2 RanSaC Methods

2.1 Notation

Table 1. Notations. This table originates from the original paper [31].

Definition	Description
$k \in \mathbb{N}_{>0}$	dimension of data points
$\mathcal{S} = \mathbb{R}^k$	space of data points
$\mathcal{P} \subset \mathcal{S}$	set of input data points
$d \in \mathbb{N}_{>0}$	degrees of freedom of a model
$\Theta \subset \mathbb{R}^d$	space of model parameters
$\theta \in \Theta$	parameter vector of a model
$s \in \mathbb{N}_{>0}$	data sample size
$Sa : 2^{\mathcal{S}} \to \mathcal{S}^s$	sampling function
$F : \mathcal{S}^s \to \Theta$	fitting function
$p : [0,1] \to [0,1]$	proba. of sampling inliers only
$D : \mathcal{S} \times \Theta \to \mathbb{R}$	point-model residual function
$\sigma \in \mathbb{R}$	inlier threshold
$I : \Theta \times \mathbb{R} \to 2^{\mathcal{P}}$	inlier selector function
$\mathcal{I} = I(\theta, \sigma) \subset \mathcal{P}$	set of model inliers
$Q : 2^{\mathcal{P}} \times \Theta \to \mathbb{R}$	model quality function

The notations we use are the same as the original paper and are summarized in Table 1. The input of RanSaC algorithms are datapoint of dimension k, usually matches between two images where $k = 4$, or matches between putative 3D points and an image, where $k = 5$. We write $\mathcal{S} = \mathbb{R}^k$, $k > 0$, the ambient space of the points and $\mathcal{P} \subset \mathcal{S}$ the set of available input.

$\mathcal{P}$ contains both points originating from the true model, that we call inliers, even though they might not fit perfectly the model due to noise, and points that have no relation to the model, called outliers, that the algorithm will try to discriminate. The unknown proportion of inliers in the dataset is noted $\epsilon \in [0, 1]$.

A model can be estimated on a sample of size d, the degree of freedom of the model, obtained with a sampler Sa that proposes s datapoints. Sa is usually a uniform sampler in $\mathcal{P}^s$ and thus the probability of drawing an uncontaminated

sample is $p(\epsilon) = \epsilon^s$; however some algorithms presented in Section 2.2 do not use a uniform sampler. This sample is passed to a fitting function, or estimator F, which gives a solution in $\Theta \subset \mathbb{R}^d$, the parameter space. $d = 8$ for homographies, 7 for fundamental matrices because of rank 2 constraint, 5 for essential matrices, and 6 for PnP.

For a model θ, it is possible to compute its quality using the function Q. This function usually depends on the selected inliers of the model $\mathcal{I} = I(\theta)$, where I is the selector function, for example $Q(\mathcal{I}, \theta) = |\mathcal{I}|$ the number of inliers for standard RanSaC. The classic selector of RanSaC $I(\theta, \sigma) = \{P \in \mathcal{P} : D(P, \theta) < \sigma\}$ depends on the inlier/outlier threshold σ and the computation of all residuals of datapoints $P \in \mathcal{P}$ with error function $D(P, \theta)$.

D varies with the model, I varies with the algorithm as the standard definition increases with σ and as such is not appropriate for an algorithm that would estimate σ and θ simultaneously. Another usual parameter of the presented algorithms are the confidence they have regarding some errors, like the type II error—the risk of missing a valid solution because of early termination—for classic RanSaC.

2.2 History of RanSaC Algorithms

Before presenting novel adaptative algorithms, algorithm 1 presents the pseudocode for the generic RanSaC from [9]. Most of the improvements have largely similar structure and mostly change a combination of Sa, I, Q. A few can propose widely different methods.

Input: $\mathcal{P}$, Sa, F, p, Q, σ
Input: confidence against type II error β, it_{max} (or min. inlier rate ϵ)
Output: Best model parameters and set of inliers
$\mathcal{I}_{max} = \emptyset$, $q_{max} = 0$
$it = 0$ (and $it_{max} = \frac{ln(1-\beta)}{ln(1-p(\epsilon))}$ if ϵ is input)
while $it \leq it_{max}$ **do**
  $sample = Sa(\mathcal{P})$, $\theta = F(sample)$
  $\mathcal{I} = I(\theta, \sigma)$, $q = Q(\mathcal{I}, \theta)$
  if $q > q_{max}$ **then**
    $q_{max} = q, \theta_{max} = \theta, \mathcal{I}_{max} = \mathcal{I}$
    $\epsilon = |\mathcal{I}|/|\mathcal{P}|, it_{max} = \frac{ln(1-\beta)}{ln(1-p(\epsilon))}$
  $it = it + 1$
return $\theta_{max}, \mathcal{I}_{max}$

Algorithm 1: RanSaC algorithm. Algorithm originated from [31]

Most authors, when designing a new RanSaC method, need to make assumption about the data or the models. They can be about the distribution of inliers or outliers, assuming some distribution of the points or their residuals, or a cross-model-consistency where patterns repeat across different models

Table 2. Overview of robust fitting methods with adaptive inlier criteria. Bracketed terms in the "Consistency assumption" column specify where the assumption is made.

Ref.	Consistency assumption	Optimized function
[9]	Bounded residuals (inliers)	Inlier set size
[20]	Gaussian residual distribution (inliers)	Scale estimate
[35]	Gaussian (inliers) Minimal residual density (transition)	Inlier/threshold
[8]	Gaussian (inliers) Residuals correlate (cross-model)	Scale estimate
[3]	Low parameter variance (cross-model)	Variance
[22, 21]	Problem specific (outlier)	Number of false alarms
[5]	Uniform (outliers)	Likelihood
[7]	Data-driven (outliers)	False discovery rate
[1]	Uniform (inliers) - Uniform (outliers)	Deviation
[2]	Uniform (inliers) - Uniform (outliers)	Deviation
[28]	Problem specific (outliers)	Greedy number of false alarms
[12]		Graph cuts energy
[34]	Residuals correlate (cross-model)	Inlier cluster merging cost
[18]	Residuals correlate (cross-model)	Residual cluster merging cost
[19]	Residuals correlate (cross-model)	Factorization error of residual matrix

and can be related to detect valid models. Table 2 regroups such methods and summarizes the assumption and quantity measured by the quality function for the presented algorithms. For example RanSaC [9] is inlier consistent: all inlier residuals are counted and the quantity optimized is the number of inliers.

Different hypotheses can be used for the inlier consistent methods. For example, MUSE [20] estimates the noise level of all inliers set based on inlier residuals following a Gaussian distribution. This noise level is used as the quality measure of the model, smaller being better. [35] works with any unimodal distribution, though a Gaussian distribution is used in practice. With a threshold on the residual density, the inlier to inlier-threshold ratio gives the quality of a model.

Outlier consistent methods include A Contrario RanSaC (AC-RanSaC) [21, 22] (first named ORSA in [23]) which assumes a background distribution for residual if the input is random, and checks the likeliness of getting a model. To do this, it tests all possible thresholds and ensures that a given model and threshold are selected only if the probability that they occur by chance is lower than some user defined confidence. The likelihood-ratio test [5] can also be used to assess the quality of a model against a uniform distribution of outlier residuals.

MAGSAC [1] and its newest improvement MAGSAC++ [2] make assumptions both on the inliers being uniformly distributed and the outliers being uniformly distributed on a different space. The quality function is based on the likelihood of a model given this hypothesis but the main improvement proposed is a refinement step to weigh the potential inliers instead of using a computed threshold.

StarSac [3] introduces cross-model-consistency by estimating multiple models for a given threshold and ranking the thresholds based on the variance over the parameters of the models. [8] proposes a mix between inlier and cross-model assumptions using a scale estimate as quality function. A weighted median absolute deviation is computed by increasing the weight and probability of a point to be inlier when it validates a sampled model.

In this paper only single model methods are considered; however, many threshold-free methods exist for such situations. Some still follow previous hypotheses, like [28] that uses multiple AC-RanSaC and merges results according to some criteria. Using cross-model-consistency, J-Linkage [34] and T-Linkage [18] cluster inliers and models to detect the good set of models. [12] uses graph cuts on an energy that depends on the total residual and number of models to compute a valid set of solutions.

3 Adaptative RanSaC Algorithms

We selected for our benchmark eight algorithms among those presented in Section 2.2, first RanSaC [9] which will be used as baseline, then MUSE [20], StaRSaC [3], A-Contrario RanSaC (AC-RanSaC) [22], Likelihood Ratio Test (LRT) [5], Marginalizing Sample Consensus (MAGSAC) [1] and two supposed improvements Fast-AC-RanSaC[26] and MAGSAC++ [2]. These algorithms where chosen as they are threshold-free methods and used for SfM and MVS tasks. We excluded multi-model specific methods like [12, 34, 18] as we concentrate our benchmark around the classification performance of the algorithms.

For the baseline, we used the results of RanSaC with a fixed σ in pixel. This baseline will be evaluated at two different thresholds to present the performance of non-adaptative methods even when the threshold is somewhat well chosen. The classic stopping formula to compute the number of iterations it_{max} of algorithm 1 has been changed to have confidence β that at least $n = 5$ good samples have been drawn. The new formula is:

$$it_{max} = \frac{\log(1-\beta)}{\log(1-\epsilon^s)} + \frac{-\log\left(\sum_{i=0}^{n-1}\left(\frac{epsilon^s}{1-epsilon^s}\right)^i\right)}{\log(1-\epsilon^s)} \tag{1}$$

where the first fraction is the usual stopping criteria of RanSaC and the second one is a positive value that increases the required number of iterations to reach this new confidence criterion. Its implementation was adapted from [24].

MUSE [20] is an adaptation of Least Median of Squares [14] that uses scale estimates as objective function to rank models using the standard iterative sampling of minimal samples of RanSaC. The article claims that the new objective function is more robust to higher outlier ratios. This algorithm does not include a stopping criteria when confidence β is reached, thus we added the usual RanSaC one as it adapts seamlessly to the framework. Implementation was adapted from https://github.com/vxl/vxl.

Likelihood Ratio Test (LRT) [5] introduces control over both the type I and type II errors. The type II error confidence β impacts the stopping criteria similarly to RanSaC. However, an early bailout strategy is implemented to reduce the number of residuals to compute when a model might not beat the so-far-the-best one, which requires a new parameter γ to control the increased risk. This early bailout strategy shifts the value of the stopping criteria:

$$it_{max} = \frac{\log(1-\beta)}{\log(1-\underline{\epsilon}^s \times \gamma)} \tag{2}$$

where $\underline{\epsilon}$ is the minimal value of the possible future inlier ratios to find a better model. The control over the type I error comes from the quality function of a model which is the likelihood for the dataset to be non-random at proposed σ.

$$Q = L(\epsilon, \sigma) = 2|\mathcal{P}| \left(\epsilon \log \frac{\epsilon}{p_\sigma} + (1-\epsilon) \log \frac{1-\epsilon}{1-p_\sigma} \right), \tag{3}$$

with inlier ratio $\epsilon = k(\sigma)/|\mathcal{P}|$ and σ spanning a predefined list $\{\sigma_{min}, \ldots, \sigma_{max}\}$. We reimplemented this algorithm in [30].

StaRSaC [3] proposes the most intuitive solution to remove thresholds: simply launch RanSaC at different threshold values and select the best performing one according to a well chosen quality function Q. Q is defined as the variance over parameters. It is computed by launching $n_{starsac}$ RanSaC for each tested thresholds and computing the variance of the parameters of the estimated models $\theta_i(\sigma)$ for this threshold. A threshold that leads to less variance in model parameters should be an appropriate threshold.

$$Q(\sigma) = -\frac{1}{n_{starsac}} \left(\sum_{i=0}^{n_{starsac}} \left(\bar{\theta}_i(\sigma) - \theta_i(\sigma) \right)^2 \right) \tag{4}$$

where $\bar{\theta}_i(\sigma) = \frac{1}{n_{starsac}} \sum_{i=0}^{n_{starsac}} \theta_i(\sigma)$ is the mean of estimated parameters. To reduce runtime we reduced the range of tested thresholds σ to a smaller range around possible values. The selected range and the step size is the same we used for LRT defined above. We reimplemented this algorithm ourselves.

AC-RanSaC [21, 22] estimates the quality of a given model by considering all residuals as a potential threshold value. The best model will be the one with lowest Number of False Alarm (NFA), a measure of the type I Error.

$$Q = -NFA(\theta, \sigma) \sim \binom{|\mathcal{P}|}{k(\sigma)} \binom{k(\sigma)}{s} p_\sigma^{k(\sigma)-s}, \tag{5}$$

with $k(\sigma) = |I(\theta, \sigma)|$ the number of inliers at threshold σ and p_σ the relative area of the image zone defining inliers at σ. To accelerate computation the residuals are sorted in order to easily compute the NFA for each of them. AC-RanSaC always reserves the last 10% of the maximum number of iterations to improve the model by reducing the input dataset $\mathcal{P}$ to the so-far-the-best inlier set. The parameters of this method are σ_{max}, the maximum value of residual for which the

NFA is computed and the upper bound of the NFA to consider the run successful NFA_{max}. However, the first parameter can technically be set to infinity with little increase in runtime if a good value cannot be guessed and a good value for the second is $NFA_{max} = 1$. Implementation was adapted from [24].

The Fast-AC-RanSaC we used in this paper is an improvement over our previous attempt in the original paper used in OpenMVG [26]. The main difference with traditional AC-RanSaC is the use of a histogram to classify errors and thus remove the need for a sorting step which slows the algorithm as discussed in Section 5.4. The residuals are thus just dispatched in n_{bin} in $\mathcal{O}(n)$ and the Number of False Alarms is computed at each value separating those bins $B_i, \ \forall i \in [n_{bin}]$:

$$NFA(\theta, B_i) \sim \binom{|\mathcal{P}|}{k(B_i)} \binom{k(B_i)}{s} p_\sigma^{k(B_i)-s}, \tag{6}$$

Our implementation was adapted from OpenMVG [26].

MAGSAC [1] makes hypotheses about the distribution of inliers and outliers, that they are uniform, and derives the likelihood of the model given these hypotheses as quality function. However, the main contribution of MAGSAC is to remove the need for an inlier/outlier threshold by introducing the σ-consensus method. The weights depend on various models estimated for different residual segmentations. This post-processing weighs a set of pseudo-inliers to fit a model with more confidence. It still requires parameters for the pseudo-inlier threshold σ_{max}, a reference threshold σ_{ref} and the number of segmentations to compute the weights of pseudo-inliers. The pseudo-inlier threshold σ_{max} has very low impact on the result. Implementation was adapted from [1].

MAGSAC++ [2] is a modification of the MAGSAC [1] algorithm. The main idea behind this algorithm is to remove entirely the need for inlier/outlier threshold by estimating a model with a weighted least square estimation. There is a threshold σ_{max} that determines the maximum residual a point can have to be considered inlier and thus have a weight in the estimation. MAGSAC++ uses a reweighted least square instead of multiple least square fittings. The computed weights are changed compared to the previous MAGSAC as well as the quality function, but the assumptions made on the inlier distribution and noise distribution are the same. Implementation was adapted from [2].

4 Data generation methodology

4.1 Models and estimators

Four different problems are considered in this benchmark: homography estimation, fundamental matrix and essential matrix estimation, and the PnP problem. These estimation problems are all core tasks of Structure from Motion and Multi-View Reconstruction. All models except fundamental and essential matrix estimation require a different processing but the pipeline remains the same: a feature extraction and matching, a RanSaC step to select inliers from the matches and select the best parameters of the model thanks to a minimal estimator, and,

usually, a refinement step at the end. Thus, to study the pros and cons of different threshold-free RanSaC algorithms we can use the same methodology for all models with little adaptation.

The homography estimation problem consists in estimating the projection from one image to the other originating from a rotation of the camera or a any movement around a flat scene. Correspondences are established between the two images and, as homographies maintain alignment it is possible to compute the deformation between these points. The fundamental and essential matrix estimation problems both consist in finding the relative pose of two cameras. The difference lies with the *a priori* knowledge of the intrinsic parameters of the camera for the essential matrix. Similarly to homography estimation, correspondences between the two images are established and then, using the epipolar constraint, it is possible to find the matrix.

The Perspective from n Points (PnP) problem consists in estimating the pose of a camera from correspondences between 3D points and their 2D image projection. From the correspondences, the position and rotation of the camera is estimated assuming the intrinsic camera parameters are known. This step is used to iteratively add views to an initial two-view estimation and obtain a complete reconstruction of a scene. The estimator used to compute the pose is the EPnPalgorithm [13]. This algorithm proposes an $\mathcal{O}(n)$ non-iterative solution, compared to previous solutions that where either iterative and/or $\mathcal{O}(n^2)$ at least. The EPnPalgorithm rewrites the coordinates of the correspondences in the coordinate system of four control points. These control points are chosen to form a basis with the center of the data and its principal directions. Then, from expressing the projection from 2D to 3D in this new coordinate system, the camera pose can be computed as the right sum of null vectors of a derived matrix.

4.2 Semi-artificial data generation method

Our proposed benchmark relies on semi-artificial data. Indeed, fully synthetic data are easy to control and generate but can be unrealistic and highly impacted by the generation choices. On the other hand, using data extracted from photographs is important to make sure the algorithm will succeed in real-life scenarios but it can be hard to compute reliable metrics when the ground-truth is hard to obtain. Thus it is hard to generalize results obtained on limited datasets with unknown noise and unknown outliers. Our solution uses real images models and data extracted from these to initialize our artificial dataset, giving us control over inlier noise and outlier distribution of synthetic data and the capacity to compute metrics reliably while retaining a realistic setup.

The pipeline used to generate a semi-artificial dataset can be used on any available dataset. If the dataset does not provide the matches, those can be computed using an *ad-hoc* algorithm like SIFT [17]. From the matches, containing both unknown inliers and outliers, a first model is estimated using a RanSaC algorithm. We chose AC-RanSaC [21, 22] at arbitrary precision and up to 10 000

iterations; our early experiments did not show any impact of the chosen algorithm and AC-RanSaC appeared to be the most stable one while its relative slowness was not an issue for this step of the pipeline. This first step creates an initial model and an inlier/outlier split. The model is considered "ground-truth": it is not the true model associated to the input data but it is realistic and thus can be used to generate new semi-artificial data. The outliers are discarded and the inliers are corrected to make "ground-truth" inliers using the "ground-truth" model.

This step creates a known model which can then be used to generate noisy inliers and true outliers in a known quantity and controlled distribution. To create noisy inliers, uniform noise is added to the "ground-truth" inliers. For matches, it is sufficient to add noise to one of the points, as this gives better control over the value of and distribution of the noise, for example avoiding cancelling if the same noise is added on both elements of the match. The choice of uniform noise for inliers has two reasons: no differences were observed in the initial tests between Gaussian and uniform noise and it would be extremely complicated to evaluate the true distribution of noise coming from SIFT matches or such. Given these noisy inliers, we have the true inlier/outlier threshold, by simply taking the maximum added noise.

Once our noisy inliers are generated, the outliers can be added. To create an outlier match, first a point is drawn at random on one side of the matches. Then, its projection is computed on the other side. This gives us an "inlier region" associated to the first point. A random point can thus be drawn outside this zone to generate a true outlier. However, drawing a match uniformly in pixel space outside the "inlier region" results in a poor distribution of errors as most matches end up very far from the model and thus do not offer a significant challenge to the RanSaC algorithms. A better method is to draw uniformly the error of the match thus giving a good range of errors for the outliers and a realistic behavior for the RanSaC algorithms. The number of generated outliers varies during the benchmark in order to obtain a desired outlier ratio. However, to avoid slowing down the benchmark with situations where it is required to generate 9 times more outliers than inliers to get a 90% outlier ratio, the maximum number of matches is thresholded. If need be, inliers and outliers are removed in order to keep the desired ratio and less than 4000 matches.

The computation of an inlier or of the "inlier region" varies depending of the studied model. For homography estimation, inlier points in the first image are mapped into the second using the ground truth homography (figure 1) to create "ground-truth" inliers. Then inlier noise is added by a uniform 2D perturbation in $[-\sigma_{noise}, \sigma_{noise}]^2$ where the maximum inlier noise σ_{noise} varies during the benchmark. Once the inliers are drawn, to create outliers, a random point is drawn in the first image, then its "inlier region" in the second image is defined by a circle of radius σ_{noise} centered around the projection of the first point. The matching point is then drawn in a random, uniform, direction at a randomly, uniformly drawn distance from the "inlier region" (figure 3).

For fundamental and essential matrices, the inlier points on the second image are simply projected orthogonaly on the epipolar line computed with the "ground-truth" matrix associated to their matches in the first image to create "ground-truth" inliers (figure 2). Then the inlier noise is added in the second image perpendicularly to the epipolar line by drawing an uniform noise in $[-\sigma_{noise}, \sigma_{noise}]$. To compute the "inlier region" in the second image of a point from the first image to create an outlier, the zone of width $2\sigma_{noise}$ around the epipolar line is defined. Then, a random position is drawn along this line and from this direction, a random error perpendicular to the line (figure 4).

For the PnP problem, the 3D points are projected on the 2D image to create the "ground-truth" inliers. The noise is added on the 2D image by a uniform 2D perturbation in $[-\sigma_{noise}, \sigma_{noise}]^2$. To create outliers, a bounding box is defined around the inlier 3D points as

$$\begin{aligned} &[r_{aug} \min_{\forall i \in [n]}(x_i), r_{aug} \max_{\forall i \in [n]}(x_i)] \times \\ &[r_{aug} \min_{\forall i \in [n]}(y_i), r_{aug} \max_{\forall i \in [n]}(y_i)] \times \\ &[r_{aug} \min_{\forall i \in [n]}(z_i), r_{aug} \max_{\forall i \in [n]}(z_i)] \end{aligned} \tag{7}$$

where $r_{aug} = 1.1$ is a small factor to increase the range of outliers around the inliers. 3D points are then drawn in this bounding box. Then, following the same principle as for homographies, the 3D point is projected on the 2D image to create the "inlier region" and the outlier match is created by a a random, uniform, direction at a randomly, uniformly drawn distance from this zone.

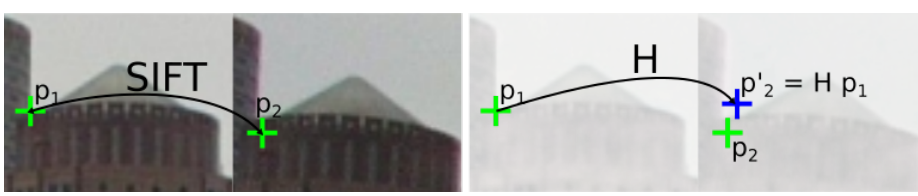

Fig. 1. From an imperfect match (p_1, p_2) considered inlier by AC-RanSaC, the "perfect match" (p_1, p'_2) is constructed such that $p'_2 = Hp_1$ using a realistic homography H given by AC-RanSaC. Figure originates from original paper [31]
.

5 Benchmark and Results

5.1 Performance Measures

Thanks to the semi-artificial data generation method, we have access to the label of the matches and thus can compute precision and recall to evaluate the

Fig. 2. From an imperfect match (p_1, p_2) considered inlier by AC-RanSaC, the "perfect match" (p_1, p_2') is constructed using p_2' the orthogonal projection of p_2 on the epipolar line $\mathcal{L}_1 = Fp_1$ where F is a realistic fundamental matrix given by AC-RanSac. This does not guarantee that p_2' represents the same physical point as p_1, but that some 3D point at possibly different depth projects exactly at p_1 and p_2'. Figure originates from original paper [31]
.

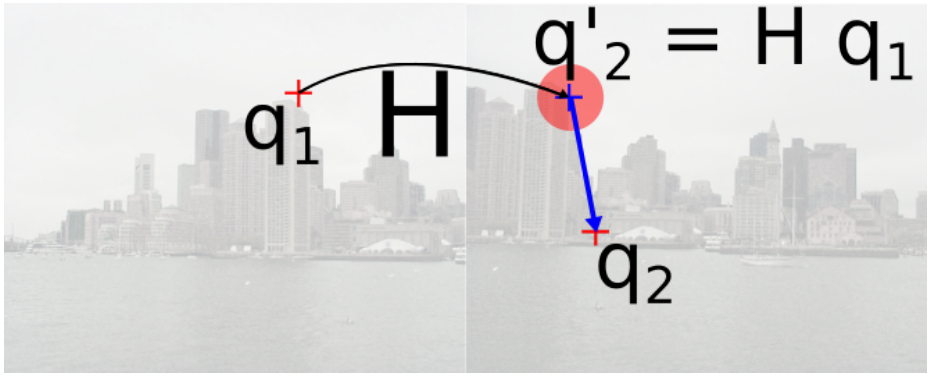

Fig. 3. A random point q_1 is drawn from the left image. Using the ground truth model H, its perfect match $q_2' = Hq_1$ is computed. Then a direction and a distance to q_2' are drawn uniformly in order to create q_2 so that it remains in the image and out of the inlier zone (marked in red) defined by the inlier noise level. Figure originates from original paper [31]
.

performance of the methods. Precision is computed as the number of correctly classified inliers over the number of detected inliers while recall its the same number over the true number of inliers. In the first paper we observed that, baring RanSaC, no algorithms presented compromise between precision and recall metrics depending on the dataset parameters. Thus the F1-Score is a good alternative to observe the results of the algorithms in a synthetic manner.

MAGSAC and MAGSAC++ do not provide an inlier/outlier classification but a weight on pseudo-inliers. As such, we introduce three metrics: `Magsac-P`, `Magsac-R` and `Magsac-W`. The first is the highest recall MAGSAC would get to obtain the same precision as AC-RanSaC. The second is the highest precision MAGSAC would get tot obtain the same recall as AC-RanSaC. The last is a weighted version of precision and recall.

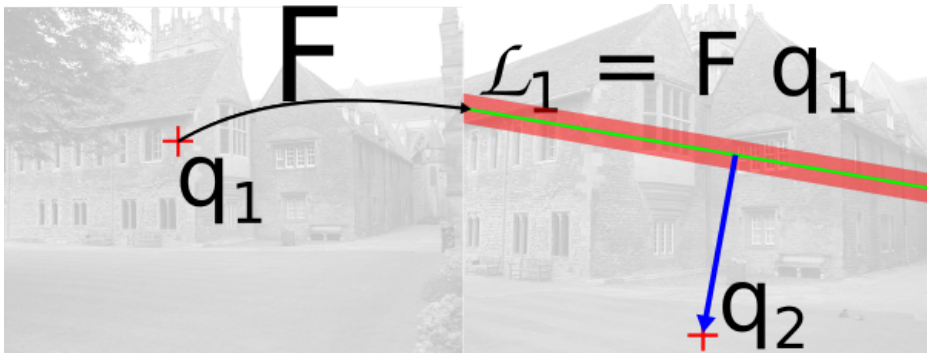

Fig. 4. A random point q_1 is drawn from the left image. Using the ground truth model F, the epipolar line $\mathcal{L}_1 = Fq_1$ is computed. Then position on this line and a distance to $\mathcal{L}_1$ are drawn uniformly in order to create q_2 so that it remains in the image and out of the inlier zone (marked in red) defined by the inlier noise level. Figure originates from original paper [31]
.

The metrics are computed over different values of inlier noise levels and outliers to observe the classification performance and its evolution for the different algorithms. We also report runtime to evaluate its evolution across the different test cases for the different models.

5.2 Parameters

This section details the different elements related to the experiments: the datasets from which we extracted the input data for the generator, the solvers used for each task and the parameters of the generator and the different tested algorithms.

Datasets: The images used come from USAC [29] — 10 image pairs for homographies estimation, 11 for fundamental matrix estimation and 6 for essential matrix estimation — Multi-H — 24 image pairs for fundamental matrix estimation —, kusvod2 — 16 image pairs for fundamental matrix estimation —, homogr — 16 image pairs for homographies estimation —[1] and megadepth [16] — 16 images for the PnP problem were used.

Solvers: As minimal solvers F, we use the standard 4-point estimator for homography and 7-point estimator for fundamental matrix [11], the 5-point estimator for essential matrix [27] and the P3P algorithm for the PnP problem. For non-minimal solvers, least-square evaluation was used for two-view geometry problems and the EPNP[13] algorithm for PnP. For MAGSAC and MAGSAC++

[1] Multi-H, kusvod2 and homogr can be found at http://cmp.felk.cvut.cz/data/geometry2view/

weighted version of the non minimal solvers are required and we adapted the EPNP algorithm to work in a weighted case.

Data generation parameters: To average results across multiple runs we generate N_{gen} different datasets with same inlier noise σ_{noise} and outlier ratio $1-\epsilon$ parameter for each image pair or 3D-2D matches. Each dataset is then evaluated N_{run} times. $N_{gen} = N_{run} = 5$ for a total of 25 different runs on which to compute the metrics. For inlier noise, we chose values in pixels that represent meaningful values, ranging from no noise (0) to 3 pixels by increments of 0.1. The outlier noise varied in $[0, 0.9]$ by increments of 0.1.

Algorithms hyperparameters: When possible we extracted the parameters of the tested algorithms from their original publications, otherwise we chose value after some initials tests. Those values are summarized in table 3.

Table 3. Hyperparameters of the tested algorithms, see section 3 for definitions.

Parameter name	Value	Algorithms using it
Success confidence β	0.99	RanSaC, StarSac, MUSE, LRT, AC-RanSaC, Fast-AC-RanSaC
Inlier search cutoff σ_{max}	16 pixels	AC-RanSaC, Fast-AC-RanSaC, StarSaC and LRT
NFA maximum value NFA_{max}	1	AC-RanSaC, Fast-AC-RanSaC
Expected type I error α	0.01	LRT
Increase in type II error from early bailout γ	0.05	LRT
Number of data partitions p	10	MAGSAC, MAGSAC++
Pseudo-inlier threshold σ_{max}	10	MAGSAC, MAGSAC++
Reference threshold σ_{ref}	1	MAGSAC, MAGSAC++

5.3 Results

We exclude StaRSaC from the presented results as it extremely slow to run, with runtimes in the minutes for baseline or just above baseline levels of performance. All other adaptive methods performed better across all metrics and test cases.

The first element we consider is runtime, as its value will impact the possible uses of the algorithm in real-time applications. It also reveals difference in behavior between algorithms, like MUSE, Fast-AC-RanSaC, AC-RanSaC and MAGSAC++ which are mostly not impacted by the noise level, and their speed depends mainly on the number of points considered whereas other algorithms, like LRT, RanSaC and MAGSAC can present huge differences of runtime between low and high noise levels. Regardless of the situation RanSaC, LRT and MUSE are usually the fastest algorithms except when reaching hard test scenarios. In those cases, LRT and RanSaC can have runtimes increase up to five

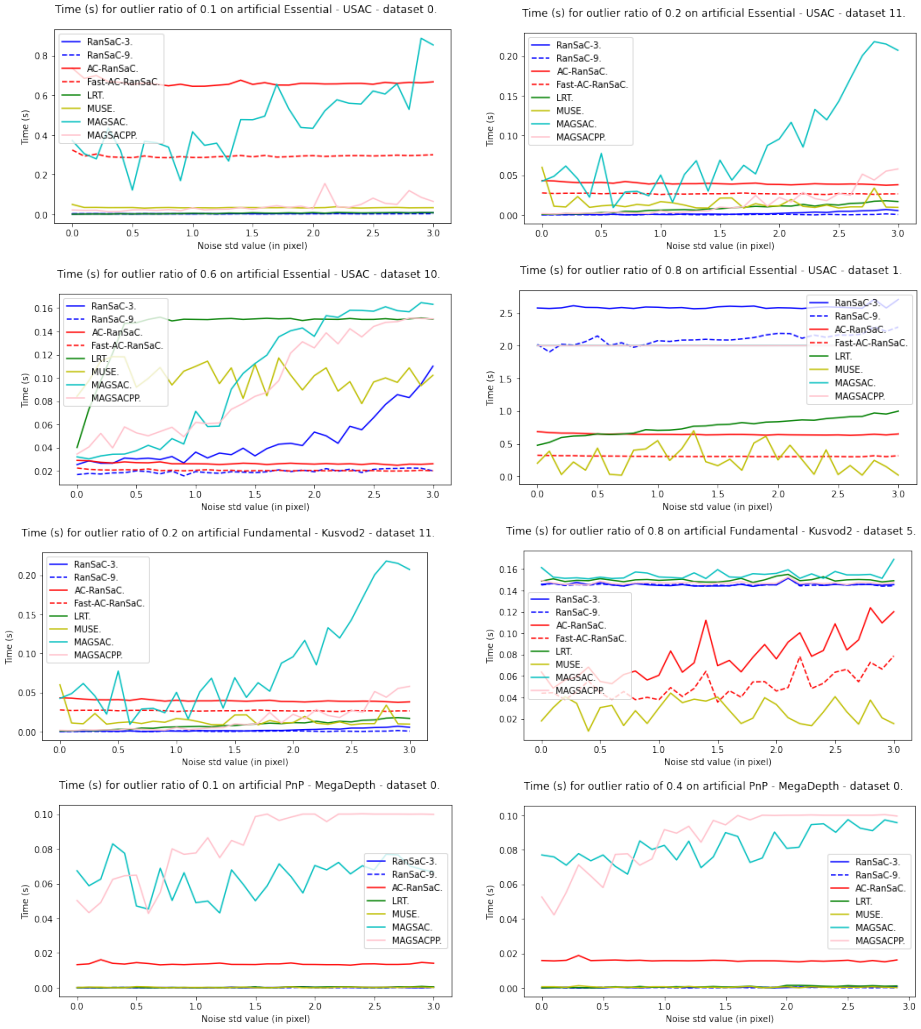

Fig. 5. Runtimes over different inlier noise levels and outlier ratios. The fitting problem, dataset name, and image pair number are in each graph title. `Ransac-`σ corresponds to RanSaC with threshold σ.

seconds for a run while MUSE will keep a very low runtime. MAGSAC and MAGSAC++ have intermediate runtimes, lower for easy settings and higher for complex settings but with a small range of variations, especially for MAGSAC++ which is usually a bit faster than MAGSAC and with almost no runtime variation regarding the inlier noise. For two-view tasks they usually fail to terminate in a reasonable amount of time, and so are interrupted at 2 seconds for a run. AC-RanSaC usually is the slowest algorithm on easy settings, but, as its runtime is not impacted by inlier noise but mostly by the number of matches in

the dataset, it can be competitive on complex algorithms when other algorithms show huge increase of their runtime. Fast-AC-RanSaC is usually twice as fast as Ac-RanSaC and is also not impacted by inlier noise. In easy settings its runtime is higher than most algorithms but it can be one of the fastest for the most complex settings.

When observing behaviors of algorithms, none of the three two-view geometry tasks presented a different challenge for the algorithms. A specific image pair or data generation parameter setup might prove more or less difficult but there was no major difference across homography, fundamental and essential matrix estimation. On the other hand, some algorithms presented huge change of performance when faced with the PnP problem.

Figures 6 and 7 illustrate the typical behaviors with low and high outlier ratios on a variety of estimation problems and datasets.

The two classic RanSaC show the expected behavior with good performance for easy settings that quickly degrade as the inlier noise and outlier ratio increase. On the one hand, the one with the lowest threshold has higher precision than the one with higher threshold with huge drops of performance for high noise or outlier ratio. On the other hand, concerning recall, the high threshold one performs better than the other and, moreover, its recall remains high while for a 3 pixels threshold, it drops significantly.

The MUSE algorithm offers quite poor performance, sometimes even below baseline in easy settings. However, its precision can remain high even in the most complex settings with more than 95% of good selections. This is due to selections of low thresholds by the algorithm, which negatively impacts its recall compared to other algorithms.

AC-RanSaC shows good performance across the benchmark, almost never showing significant drops in precision and small drops in recall for the most complex scenarios lowering the F1-Score.

For LRT, its performance is on par, on just below, the best performing algorithms for easy to medium settings. However, as the benchmark parameter increase, it gets closer to RanSaC performances with significant drops in F1-Score.

Fast-AC-RanSaC performs slightly worst than AC-RanSaC for the easiest settings, with good recall but lesser precision. Its performance is usually quite similar to that of LRT either slightly better or slightly worse depending on image pairs. This is due to a usually better recall but not always better precision from Fast-AC-RanSaC. It is also a bit more unstable with some runs where it fails to find a good model and stops on too high a threshold or a contaminated sample.

MAGSAC and its newest version MAGSAC++ have quite similar behavior. For all three methods to compute their precision and recall 5.1, they tend to perform better than most algorithms for all settings on two-view geometry tasks. They are the only algorithm that retain above 90% precision and above 80% recall even when presented with the highest inlier noise levels and outlier ratios. In those settings, MAGSAC++ obtains comparable or better results than MAGSAC but the difference between the two is small. However, for the PnP

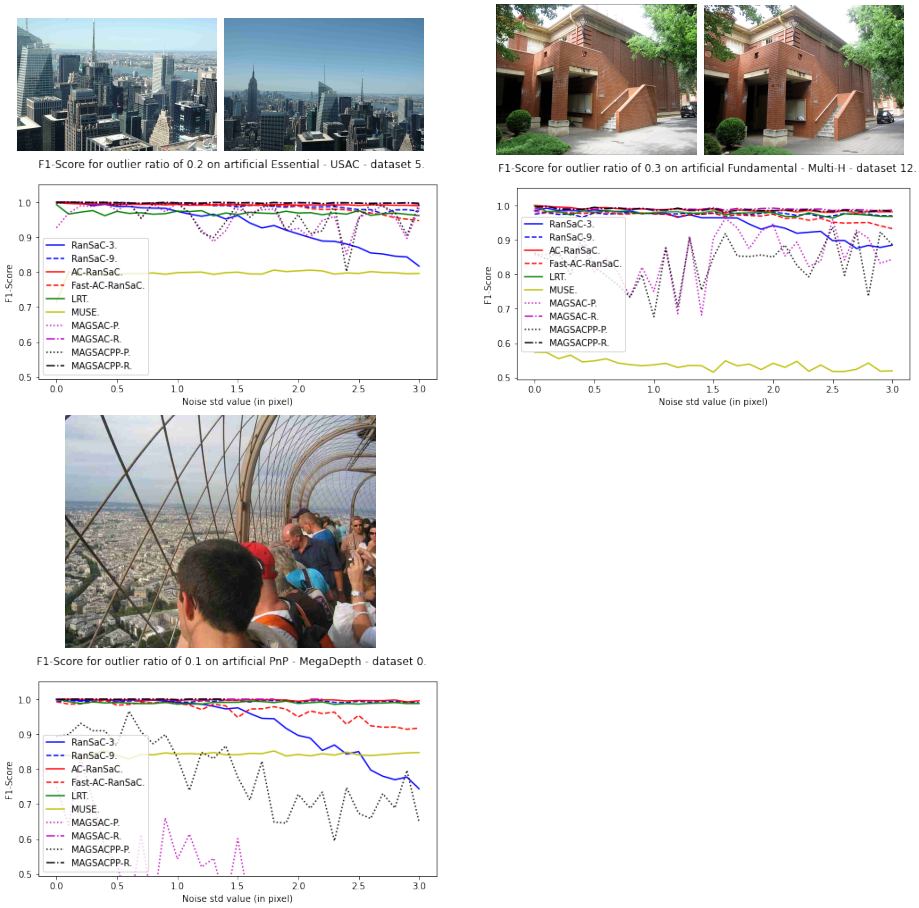

Fig. 6. Typical F1-score evolution over inlier noise for low outlier ratios. Estimation problem, dataset name and image pair number can be found in each graph's title. `Magsac-P`, `Magsac-R` and `Magsac-W` correspond to the metrics presented in Section 5.1.

task, MAGSAC fails to produce satisfactory results. MAGSAC++ still performs slightly better but keeps a lot of failure cases. The precision of both algorithms remains somewhat stable but their recall can drop to extremely low values and fail to find any good model, even when lifting the runtime limitations.

5.4 Analysis and comparison

As we observed similar behavior across different datasets, different images, our data generation methodology seems to be able to reveal intrinsic capacities of the tested algorithms. Indeed, while specific values might change, the ordering of algorithms, the drops in performance, the runtime evolution, are consistent across all test cases and depend on the data generation parameter and not the

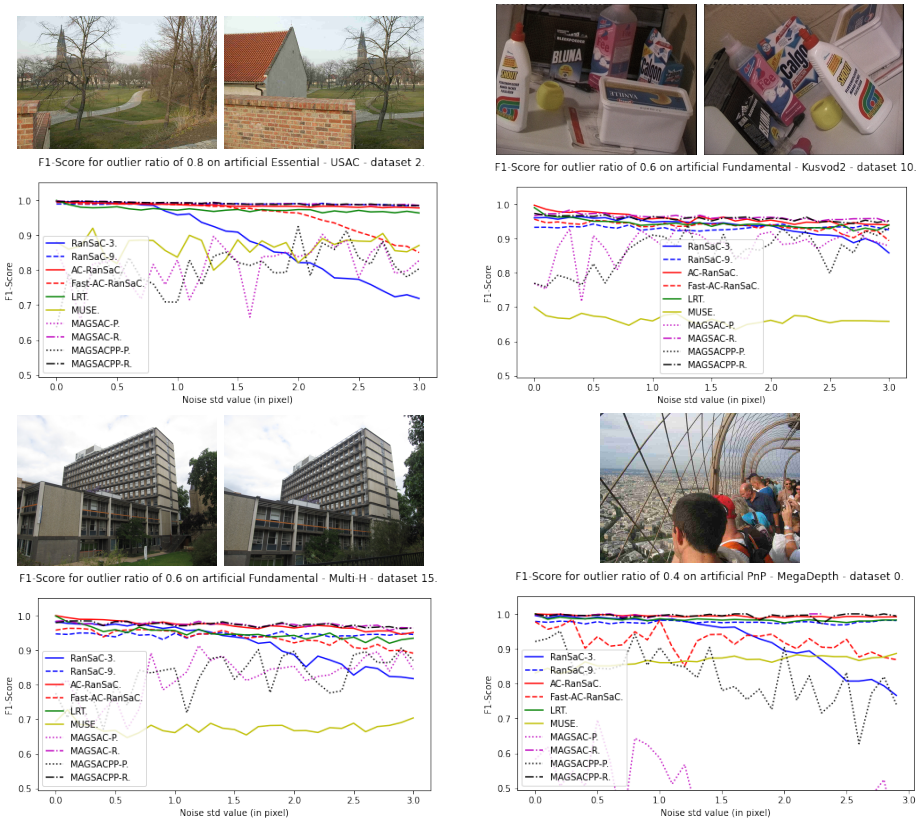

Fig. 7. Typical F1-score evolution over inlier noise for high outlier ratios. Estimation problem, dataset name and image pair number can be found in each graph's title. `Magsac-P`, `Magsac-R` and `Magsac-W` correspond to the metrics presented in Section 5.1.

input task or images. The only difference being MAGSAC and MAGSAC++ for the PnP problem, as discussed below.

StaRSaC and MUSE offer poor performance compared to other algorithms. The StaRSaC algorithm is just too slow to justify the small increase in performance compared to RanSaC, even when RanSaC's threshold is poorly chosen. MUSE performs better, with very high speed and high stability but, when compared to newer methods, it's classification performance is poorer than almost all newest methods.

Then, algorithms can be separated in two categories, fast and slow algorithms. This separation makes sense both in term of purpose, as algorithms that run in less than a tenth of a second can be used for real-time applications, and in term of performance as most slow algorithms perform better than faster ones. The runtime of an algorithm is mainly dependent on the computation of residuals and thus the number of iterations needed to reach a satisfactory model. Indeed,

for most algorithms, the only two steps at each iteration are the computation of a model and the estimation of the quality of this model, through its residuals. The first step is usually very fast, almost instantaneous, while the second represents the bulk of the runtime. For some algorithms, some post-processing to compute the model quality might slow down the process. For example, for AC-RanSaC the sorting step required to compute the NFA at each possible threshold, and this easily takes as much time as the residual evaluation. It is also impacted by the reserved number of iterations, as it will always at least do 10% of the maximum number of allowed iterations. For MAGSAC and MAGSAC++ the addition of the σ-consensus step impacts runtime as well. However, if better model are chosen more often, it might speed up the process. For example, LRT is a fast algorithm in easy settings, as its early bailout strategy helps it skip useless computation of residuals. But for hard settings, it can be slowed down by missing too many good models. The new Fast-AC-RanSaC algorithm offers a good compromise for a fast algorithm as it is usually fast enough to perform real-time operations and its runtime is very stable across settings.

For fast algorithms LRT performs slightly worse than Fast-AC-RanSaC in a few cases, and is almost always faster for easy cases. As it is very sensitive to the complexity of the task, it might be better to use only when the setting is known and prefer the slowest but more stable Fast-AC-RanSaC when needing a fast algorithm.

For slow algorithms, AC-RanSaC is one of the slowest but most stable and consistent solution. Baring the most complex image pairs and generation parameters it always offers good precision and recall. MAGSAC is almost always slower or less effective or both than MAGSAC++, which is to be expected as the second is an improvement on the first one. For two-view geometry, MAGSAC++ performs almost always better than AC-RanSaC, producing good results even when other algorithms fail. It is also usually faster so it is a very stable and powerful solution. On the other hand, for the PnP problem, neither MAGSAC nor MAGSAC++ produce satisfactory results. The σ-consensus step seems to not adapt well to the EPNP estimator.

As a test conclusion, a user who needs speed should prefer Fast-AC-RanSaC, a user who needs precision AC-RanSaC and for robustness in two-view geometry tasks MAGSAC++.

6 Conclusion

In the field of robust estimation, many algorithms were proposed but few reliable and complete benchmarks were existing in the field of Structure from Motion and Multi-View Stereo, due to the complexity to get ground-truth. In this paper, we solve this problem with a semi-artificial generation method that offers a good referential to compare algorithms to each other and to study the behavior of a specific algorithms depending on the test case. Thanks to this new method we were able to improve algorithms like Fast-AC-RanSaC and validate the value of the improvement compared to other state-of-the-art algorithms.

RanSaC is known to struggle with high outlier ratios, especially when its inlier/outlier threshold is poorly chosen. Adaptive methods offer a solution to this problem, with no method proving superior overall but offering compromises between robustness, accuracy and speed. Fast-AC-RanSaC proposes a fast and robust algorithm that will provide good performance for its runtime. LRT is the most efficient in easy settings but will likely fail when faced with challenging cases. MAGSAC++ and AC-RanSaC offer good precision and recall across test cases, with a cost in runtime. For two-view geometry tasks, the first is the most performing, usually faster and more robust while for the PnP problem, the second performs largely better. The ability to reveal the specifics of each RanSaC algorithm shows that the data-generation method will be able to produce clearer benchmarks and conclusions when faced with new improvements.

Thanks to such observation, we plan on implementing the best performing algorithms in a reconstruction pipeline to see the impact of removing the user set threshold to compute the appropriate value at each step and hopefully improve the whole reconstruction.

References

1. Barath, D., Matas, J., Noskova, J.: MAGSAC: marginalizing sample consensus. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10197–10205 (2019)
2. Barath, D., Noskova, J., Ivashechkin, M., Matas, J.: MAGSAC++, a fast, reliable and accurate robust estimator. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1304–1312 (2020)
3. Choi, J., Medioni, G.: StaRSaC: Stable random sample consensus for parameter estimation. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 675–682 (June 2009). https://doi.org/10.1109/CVPR.2009.5206678
4. Choi, S., Kim, T., Yu, W.: Performance evaluation of RANSAC family. In: Proceedings of the British Machine Vision Conference (BMVC) (2009)
5. Cohen, A., Zach, C.: The likelihood-ratio test and efficient robust estimation. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 2282–2290 (Dec 2015). https://doi.org/10.1109/ICCV.2015.263
6. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
7. Dehais, J., Anthimopoulos, M., Shevchik, S., Mougiakakou, S.: Two-view 3D reconstruction for food volume estimation. IEEE Transactions on Multimedia **19**(5), 1090–1099 (2017). https://doi.org/10.1109/TMM.2016.2642792
8. Fan, L., Pylvänäinen, T.: Robust scale estimation from ensemble inlier sets for random sample consensus methods. In: Proceedings of the IEEE European Conference of Computer Vision (ECCV). pp. 182–195. Springer Berlin Heidelberg (2008)
9. Fischler, M., Bolles, R.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM **24**(6), 381–395 (1981)
10. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2012)

11. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. Cambridge University Press, 2nd edn. (2004), iSBN 978-0521540513
12. Isack, H., Boykov, Y.: Energy-based geometric multi-model fitting. International Journal of Computer Vision (IJCV) **97**(2), 123–147 (Apr 2012). https://doi.org/10.1007/s11263-011-0474-7, https://doi.org/10.1007/s11263-011-0474-7
13. Lepetit, V., Moreno-Noguer, F., Fua, P.: Epnp: An accurate o (n) solution to the pnp problem. International journal of computer vision **81**(2), 155–166 (2009)
14. Leroy, A.M., Rousseeuw, P.J.: Robust regression and outlier detection. Wiley series in probability and mathematical statistics (1987)
15. Li, Z., Snavely, N.: MegaDepth: Learning single-view depth prediction from internet photos. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2041–2050 (2018)
16. Li, Z., Snavely, N.: Megadepth: Learning single-view depth prediction from internet photos. In: Computer Vision and Pattern Recognition (CVPR) (2018)
17. Lowe, D.G.: Object recognition from local scale-invariant features. In: Proceedings of the seventh IEEE international conference on computer vision. vol. 2, pp. 1150–1157. Ieee (1999)
18. Magri, L., Fusiello, A.: T-linkage: A continuous relaxation of J-linkage for multi-model fitting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3954–3961 (June 2014). https://doi.org/10.1109/CVPR.2014.505
19. Magri, L., Fusiello, A.: Robust multiple model fitting with preference analysis and low-rank approximation. In: Proceedings of the British Machine Vision Conference (BMVC). pp. 20.1–20.12 (2015). https://doi.org/10.5244/C.29.20
20. Miller, J.V., Stewart, C.V.: MUSE: Robust surface fitting using unbiased scale estimates. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 300–306 (11 1996). https://doi.org/10.1109/CVPR.1996.517089
21. Moisan, L., Moulon, P., Monasse, P.: Automatic homographic registration of a pair of images, with a contrario elimination of outliers. Image Processing On Line (IPOL) **2**, 56–73 (2012)
22. Moisan, L., Moulon, P., Monasse, P.: Fundamental matrix of a stereo pair, with a contrario elimination of outliers. Image Processing On Line (IPOL) **6**, 89–113 (2016)
23. Moisan, L., Stival, B.: A probabilistic criterion to detect rigid point matches between two images and estimate the fundamental matrix. International Journal of Computer Vision (IJCV) **57**(3), 201–218 (2004)
24. Moulon, P.: AC-RanSaC implementation https://github.com/pmoulon/IPOL_AC_RANSAC (2012), [Online; accessed 22-January-2021]
25. Moulon, P., Monasse, P., Perrot, R., Marlet, R.: OpenMVG: Open multiple view geometry. In: International Workshop on Reproducible Research in Pattern Recognition. pp. 60–74. Springer (2016)
26. Moulon, P., Monasse, P., Perrot, R., Marlet, R.: OpenMVG: Open multiple view geometry. In: International Workshop on Reproducible Research in Pattern Recognition. pp. 60–74. Springer (2016)
27. Nistér, D.: An efficient solution to the five-point relative pose problem. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) **26**(6), 756–770 (2004)

28. Rabin, J., Delon, J., Gousseau, Y., Moisan, L.: MAC-RANSAC: a robust algorithm for the recognition of multiple objects. Proceedings of the Fifth International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT) pp. 51–58 (2010)
29. Raguram, R., Chum, O., Pollefeys, M., Matas, J., Frahm, J.M.: USAC: a universal framework for random sample consensus. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) **35**(8), 2022–2038 (2012)
30. Riu, C., Nozick, V., Monasse, P.: Automatic ransac by likelihood maximization. Image Processing On Line **12**, 27–49 (2022)
31. Riu, C., Nozick, V., Monasse, P., Dehais, J.: Classification performance of ransac algorithms with automatic threshold estimation. In: 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2022). vol. 5, pp. 723–733. Scitepress (2022)
32. Schonberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4104–4113 (2016)
33. Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., Fitzgibbon, A.: Scene coordinate regression forests for camera relocalization in rgb-d images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2930–2937 (2013)
34. Toldo, R., Fusiello, A.: Image-consistent patches from unstructured points with J-linkage. Image and Vision Computing **31**, 756–770 (2013)
35. Wang, H., Suter, D.: Robust adaptive-scale parametric model estimation for computer vision. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) **26**(11), 1459–1474 (Nov 2004). https://doi.org/10.1109/TPAMI.2004.109