USAC: A Universal Framework for Random Sample Consensus

Rahul Raguram, Ondřej Chum, Marc Pollefeys, Jiří Matas and Jan-Michael Frahm

Abstract—A computational problem that arises frequently in computer vision is that of estimating the parameters of a model from data that has been contaminated by noise and outliers. More generally, any practical system that seeks to estimate quantities from noisy data measurements must have at its core, some means of dealing with data contamination. The Random Sample Consensus (RANSAC) algorithm is one of the most popular tools for robust estimation. Recent years have seen an explosion of activity in this area, leading to the development of a number of techniques that improve upon the efficiency and robustness of the basic RANSAC algorithm. In this paper, we present a comprehensive overview of recent research in RANSAC-based robust estimation, by analyzing and comparing various approaches that have been explored over the years. We provide a common context for this analysis by introducing a new framework for robust estimation, which we call Universal RANSAC (USAC). USAC extends the simple hypothesize-and-verify structure of standard RANSAC to incorporate a number of important practical and computational considerations. In addition, we provide a general-purpose C++ software library that implements the USAC framework by leveraging state of the art algorithms for the various modules. This implementation thus addresses many of the limitations of standard RANSAC within a single unified package. We benchmark the performance of the algorithm on a large collection of estimation problems. The implementation we provide can be used by researchers either as a stand-alone tool for robust estimation, or as a benchmark for evaluating new techniques.

Index Terms—RANSAC, robust estimation.

1 Introduction

A computational task that arises in a number of application scenarios is the estimation of model parameters from data that may be contaminated with measurement noise and, more significantly, may contain points that do not conform to the model being estimated. These points, called outliers, have a dramatic effect on the estimation process – a non-robust technique, such as least squares regression, can produce arbitrarily bad model estimates in the presence of a single outlier. Consequently, the field of *robust estimation* has been well studied over the years, both in the statistics community [1], [2], [3], [4], as well as in computer vision [5], [6], [7]. A wide variety of algorithms have been proposed over the past four decades, varying in the degree of robustness that they provide to outliers, the assumptions they make about the data and their computational complexity, amongst other aspects. Of these many algorithms, perhaps the one that is used most widely, particularly in computer vision, is Random Sample Consensus, or RANSAC [7].

The RANSAC algorithm is a remarkably simple, yet powerful, technique. One compelling reason for its widespread adoption, in addition to its simplicity, is the ability of the algorithm to tolerate a tremendous level of contamination, providing reliable parameter estimates even when well over half the data consists of outliers. However, while robust, the basic RANSAC algorithm has its drawbacks, impacting its accuracy, efficiency and stability. Recent years have seen exciting advances in dealing with each of these problems. Indeed, these improvements in computational efficiency and robustness have helped drive forward the state of the art, particularly as the computer vision and robotics communities push towards more challenging problems on massive real-world datasets [8], [9], [10], [11], [12] and seek real-time performance [13], [14], [15], [16]. However, while a number of recent efforts have focused on addressing issues with RANSAC, relatively less attention has been paid to a unified review of these developments. Some recent efforts in this direction are those of [17], [18], which analyse and compare the performance of some recent RANSAC variants on a selection of geometric estimation problems. We seek to extend this idea further. Our goals in this work are two-fold:

- To present a comprehensive overview of recent research in RANSAC-based robust estimation, and to provide a common context within which to study these disparate techniques.To do so, we propose a generalization of the standard *hypothesize-and-verify* structure of standard RANSAC, which we term *Universal RANSAC*, to emphasize the fact that most of the important RANSAC variants can be viewed as special cases of this USAC framework.
- To provide a general-purpose software library that implements the USAC framework. This implementation draws from the strengths and collective wisdom of prior methods, thus addressing many of the limitations of the standard RANSAC algorithm within a single unified package. The implementation is modular and can easily be extended to include other algorithms or their components. We hope that

- *R. Raguram and J-M. Frahm are with the Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC, 27599. E-mail: rraguram@cs.unc.edu, jmf@cs.unc.edu*
- *O. Chum and J. Matas are with Czech Technical University, Faculty of Electrical Engineering, Department of Cybernetics, Karlovo namesti 13, 121 35 Prague, Czech Republic. E-mail: chum@cmp.felk.cvut.cz, matas@cmp.felk.cvut.cz*
- *M. Pollefeys is with the Department of Computer Science, ETH Zurich, CNB G105, Universitatstrasse 6, CH-8092 Zurich, Switzerland. E-mail: marc.pollefeys@inf.ethz.ch*

this will be of use to researchers as a stand-alone tool for robust estimation; as a starting point for developing new RANSAC variants; or as a benchmark for evaluating new techniques.

2 BACKGROUND

2.1 The Problem

Given a set of measurements $\mathcal{U}$ containing N data points, assume that an *a priori* unknown fraction ε of these points is consistent with some underlying model of interest. These points, or inliers, may be subject to small-scale random variations, or noise, which arise from imperfections in the measurement process. The remaining fraction of data points may be thought of as gross errors that do not conform to the model of interest, and are termed outliers. The goal of the robust estimation procedure is to compute the model parameters corresponding to the inlier population.

2.2 Random Sample Consensus

The RANSAC algorithm was originally proposed by Fischler and Bolles [7] as a general framework for model fitting in the presence of outliers. The goal in RANSAC is to efficiently explore the space of model parameters in order to maximize some objective function $\mathcal{C}$. RANSAC performs this maximization in a randomized, data-driven manner. Points in the parameter space are defined by repeatedly selecting random subsets of the data points, and generating model hypotheses from each subset. Each hypothesized model is then scored using the remaining data points and the hypothesis that obtains the best score is returned as the solution. The RANSAC algorithm is outlined in Algorithm 1. Below, we describe some of the important aspects of the algorithm.

2.2.1 Objective function

In the standard RANSAC formulation, the objective function $\mathcal{C}$ to be maximized is the size of the *support* for a given model. More specifically, given a model with parameters θ_k, the support is the number of data points from $\mathcal{U}$ that have residual errors smaller than some predefined threshold t. This set of points is also called the *consensus set* and RANSAC attempts to maximize the cardinality of this set. Thus, we have

$$\mathcal{C} = \sum_i \rho(e_i^2) \tag{1}$$

where the cost function $\rho(.)$ is defined as

$$\rho(e_i^2) = \begin{cases} 1 & \text{for } e_i^2 \le t^2\,, \\ 0 & \text{for } e_i^2 > t^2\,. \end{cases} \tag{2}$$

2.2.2 Subset size

In order to maximize the objective function $\mathcal{C}$, RANSAC operates in a hypothesize-and-verify loop, repeatedly sampling subsets of the data to hypothesize model parameters, and then verifying their support against all data points. Each subset is a *minimal sample* with size m, defined by the minimum number of points required to uniquely compute the model parameters. This marks a departure from traditional regression techniques, which typically use all the available data to estimate model parameters; in contrast, RANSAC uses as *little* data as possible. The goal in RANSAC is to draw uncontaminated samples; since the probability of drawing a contaminated sample increases exponentially with its size, the size is chosen to be as small as is feasible.

2.2.3 Stopping criterion

It can be shown [7], that in order to ensure with confidence η_0 that at least *one* outlier-free set of m points is sampled in RANSAC, we must draw at least k samples, where

$$k \ge \frac{log(1-\eta_0)}{log(1-\varepsilon^m)}, \tag{3}$$

where ε is the fraction of inlying data points in the data set, and η_0 is typically set to 0.95 or 0.99. In practice, since ε is often unknown, it is possible to use a worst-case estimate of ε to precompute the number of RANSAC trials. However, it is more efficient to use an *adaptive* stopping criterion where ε is initialized with a worst-case assumption, and is updated based on the size of the maximum consensus set found [19].

An additional useful way of thinking about the number of trials in RANSAC is the following: each sample drawn is either uncontaminated (a "success") or contaminated (a "failure"). The number of uncontaminated samples drawn thus follows a binomial distribution with probability of success $p = 1/\varepsilon^m$. For sufficiently small values of p, this can be approximated by a Poisson distribution. Thus, the probability of exactly n successes in k trials can be expressed as

$$p(n,\lambda) = \frac{\lambda^n e^{-\lambda}}{n!}, \tag{4}$$

where λ is the expected number of successes in k trials. In RANSAC, we want to ensure with some confidence that the probability of having no successes falls below some threshold. Thus, we have $p(0,\lambda) = e^{-\lambda} < 1-\eta_0$. For $\eta_0 = 0.95$, $\lambda \approx 3$, implying that on average, approximately 3 good samples are drawn before the 95% confidence in the solution is reached.

2.2.4 Threshold selection

An important input parameter required by RANSAC is the threshold t, which determines whether a data point supports a particular model or not. While it is often possible to make a reasonable empirical choice, the process for threshold selection can be formalized. For instance, assume that the data points are perturbed by Gaussian noise with zero mean and standard deviation σ. In this context, the point-model error d^2 can be expressed as a sum of n squared Gaussian variables, where n is the codimension of the model. The residuals thus follow a chi-square distribution with n degrees of freedom, and the inverse chi-square distribution can be used to determine a threshold t that captures a fraction α of the true inliers [20]:

$$t^2 = \chi_n^{-1}(\alpha)\sigma^2 \tag{5}$$

where χ is the cumulative chi-square distribution,, and α is the fraction of inliers, typically set to 0.95. Thus, when using the threshold computed with $\alpha = 0.95$, a true inlier will be incorrectly rejected only 5% of the time.

Algorithm 1 Standard RANSAC algorithm

```
Input: $\mathcal{U}$, $\eta_0$, $k_{max}$, $t$
Output: $\boldsymbol{\theta}^*$, $\mathcal{I}^*$
$k = 0$, $I_{max} = 0$
while $k < k_{max}$ do
    1. Hypothesis generation
    Randomly sample minimal subset of $m$ points
    Estimate model parameters $\boldsymbol{\theta}_k$
    2. Verification
    Calculate the support set $\mathcal{I}_k$
    if $|\mathcal{I}_k| > I_{max}$ then
        $\boldsymbol{\theta}^* = \boldsymbol{\theta}_k$, $\mathcal{I}^* = \mathcal{I}_k$
        Recompute $k_{max}$ from eqn. (3) using $\varepsilon = |\mathcal{I}^*|/N$
    end if
    $k = k + 1$
end while
```

2.3 Other robust estimators

While the focus of this paper is directed primarily towards RANSAC and its variants, it is worth noting that a rich variety of robust estimation algorithms have been proposed over the years. Though it is beyond the scope of this paper to exhaustively list all these techniques, we briefly discuss a selection of relevant techniques that have been applied to computer vision problems.

Since scoring models based on the support requires a threshold, one possible approach is to optimize a different cost function. One historically significant technique, contemporaneous with RANSAC, is Least Median of Squares (LMedS) [4]. LMedS scores models based on their median error residual, returning the model with lowest score. However, this breaks down when the data consists of more than 50% outliers, since the median residual is no longer informative. Minimum Probability of Randomness (MINPRAN) [21] searches for a combination of model parameters and inliers that are least likely to have occured by chance. While this approach does not require knowledge of the noise scale, it assumes that the dynamic range of the outlier data is known. Minimum Unbiased Scale Estimate (MUSE) [22] and Adaptive Least K^{th} order Squares (ALKS) [23] are related techniques that minimize the order statistics of the squared residuals. It has been noted [24] that their ability to handle large outlier ratios is limited. Projection Based M-Estimator (pbM) [25] reformulates the M-estimator optimization criterion in terms of a projection pursuit optimization problem, automatically deriving a suitable threshold from univariate kernel density estimates. However, this can be computationally expensive, particularly as the model complexity increases [26]. More generally, one practical limitation of all the above techniques is that while they do not require the noise scale to be specified, they do require the number of model hypotheses to be supplied by the user, which either requires knowledge of the inlier ratio, or a worst-case assumption – i.e., drawing enough samples to guarantee success for the *worst* inlier ratio that could occur in the data. In other words, these techniques still rely on a user supplied parameter that can be difficult to estimate.

The StaRSaC algorithm [27] modifies RANSAC to relax the requirement of a fixed threshold, by executing RANSAC multiple times with various thresholds, choosing the one that shows minimum "variance of parameters". This procedure can be slow, based on the number of thresholds tested and the number of RANSAC repetitions per threshold. The Residual Consensus (RECON) algorithm [28] adopts a different paradigm, testing pairs of models for consistency, using the intuition that the residual errors for "good" models are in some way "consistent" with each other. While this technique does not require the number of hypotheses to be determined *a priori*, it is still more computationally expensive than RANSAC due to the fact that it must test pairs of model hypotheses.

All the techniques discussed thus far broadly follow the same basic idea: randomly generate model hypotheses, and select the model that optimizes some function of the data point residuals. In recent years, there have been some approaches that try to invert this relationship – i.e., to look at the residuals for each data point, in order to classify or cluster the points into inliers and outliers. Some prominent approaches that leverage this idea are the Ensemble Method [26], J-Linkage [29] and Kernel Fitting [30], [31]. These aproaches use the observation that for a given point, the distribution of residuals with respect to a large set of randomly generated models can be used to reveal whether the point is an outlier or an inlier. While this proves to be effective, there is still the limitation that these methods must somehow generate a "sufficient" number of model hypotheses. It is worth noting, also, that some of the approaches outlined above are more general than RANSAC, in that they are designed for the case of multi-model robust estimation, which is not a focus of the present work.

Thus far, we have focused on algorithms that explore the parameter space in a randomized way. There exist *deterministic* alternatives; for instance, Joint Compatibility Branch and Bounch [32], Active Matching [33] and the Consensus Set Maximization algorithm of [34]. The former two approaches have been applied to the problem of verifying feature matches between images, by leveraging priors on the locations of image features. In applications where strong priors on the feature locations are available (such as in high frame-rate video tracking), this can provide good results, though efficiency concerns still remain. The Consensus Set Maximization [34] approach reformulates RANSAC as a mixed integer programming problem, and solves it using a modified branch and bound algorithm. This provides an optimal solution to the robust estimation problem, unlike the randomized results provided by RANSAC; however, this comes at a significantly higher computational cost.

In summary, it is evident from the discussion in this section that the field of robust estimation is a very active one, with a variety of challenges and open problems. In subsequent sections of this paper, we consider a category of popular algorithms – RANSAC-based robust estimators – and provide a unified framework for the analysis of these algorithms.

3 A Universal RANSAC framework

In Algorithm 1, we summarized the operation of RANSAC in the form in which it is often applied. However, note that there are some important limitations of the technique when used in this form. Most significantly:

1) Efficiency: From Section 2.2, note that the time complexity in RANSAC depends on the subset size m, the inlier ratio ε, and the number of data points N. Thus, there are scenarios where the runtime of RANSAC can be prohibitively high. In addition, note that there is an implicit assumption in equation (3), that a model generated from an all-inlier sample will be "perfect", resulting in all inliers being found. In practice, this is seldom the case, since the use of minimal samples implies that the model hypotheses themselves are "noisy". Consequently, the number of iterations required by RANSAC is typically more than that predicted by equation (3) [35].

2) Accuracy: As noted in Section 2.2.2, the reliance of RANSAC on minimal samples is due to efficiency considerations. However, model parameters that are computed from minimal subsets may be significantly far from their true values. It is thus worth stressing that the model parameters returned by standard RANSAC can often be sub-optimal and must be refined prior to being used for additional tasks.

3) Degeneracy. The objective function in RANSAC relies on the assumption that a model generated from a contaminated minimal sample will have low support, thus making it possible to distinguish between correct and incorrect models. This assumption may be violated when degenerate configurations are encountered. For instance, in the case of epipolar geometry estimation with the 7-point algorithm [20], when the minimal sample comprises of five coplanar points and two *arbitrary* points, the resulting model is consistent with all points on the plane, though the resulting epipolar geometry is incorrect.

A number of techniques have been proposed in recent years to deal with these, as well as other, limitations of RANSAC. To put these disparate techniques into context, we extend the simple hypothesize-and-verify structure of RANSAC and propose a *Universal RANSAC* framework, illustrated in Figure 1. This framework generalizes RANSAC by incorporating a number of important practical and computational considerations. In the remainder of this section, we describe the various components in Figure 1, and discuss the many variations and extensions proposed in the literature within this unified context. In particular, note that all the techniques discussed below can be viewed as special cases of the USAC framework.

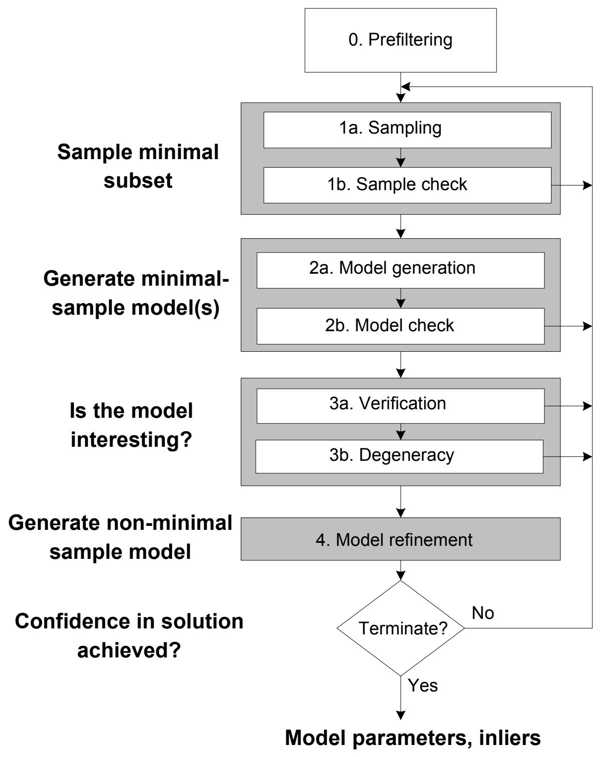


Fig. 1: The Universal RANSAC framework. The figure represents a generalized RANSAC framework that takes into account a number of important practical and computational considerations. USAC is a synthesis of various RANSAC techniques, and provides a unified view of recent advances in this area. In addition to providing a common context for the different RANSAC variants, the combination of these techniques into a single robust estimation framework allows them to interact with, and benefit from, each other. Thus, an implementation of the USAC framework as sketched above, with state of the art algorithms for each individual module, allows us to address the various limitations of RANSAC in a unified manner.

3.1 Prefiltering (Stage 0)

The input to RANSAC is a set $\mathcal{U}$, consisting of N data points. As seen in Section 2.2.3, the number of iterations required is a function of the contamination in this set. Given that the runtime of RANSAC is closely tied to the inlier ratio, a few efforts have been targeted towards improving the quality of the data points that form the input to the algorithm. Strictly speaking, this step is not a part of the core random sample consensus framework itself, since it involves "cleaning up" the input data before the algorithm commences. It is worth noting, however, that performing this step when applicable can significantly benefit RANSAC.

Considering the specific problem of estimating multi-view geometric relations, the input to RANSAC is typically a set of feature matches. Considerable effort has been directed towards imposing constraints on the output of a feature detector in order to provide a more reliable set of feature matches [36], [37], [38], [39]. In the RANSAC literature, one recent effort that investigates the computational benefit of a spatial consistency filter applied to an initial set of feature matches is SCRAMSAC [40]. This consistency filter results in a reduced set of matches that are more likely to be correct, in turn speeding up RANSAC by up to an order of magnitude.

3.2 Sample minimal subset (Stage 1)

Stage 1a: Sampling

In standard RANSAC, minimal subsets are generated by sampling uniformly at random from the set of data points. In the most general setting, this implies that no assumptions are being made about the data. In many practical situations, however, it may be possible to incorporate prior information and bias the sampling with a view towards preferentially generating models that are more likely to be correct. This can have a dramatic effect on the efficiency of RANSAC, particularly for low inlier ratios. This simple observation has been explored in a number of ways in the RANSAC literature. A review of some prominent work in this area follows.

3.2.1 NAPSAC

The N-Adjacent Points Sample Consensus (NAPSAC) algorithm [41] uses the observation that often, inliers in the data tend to be "closer" to one another than to outliers. Considering an n-dimensional space, assume that outliers are distributed uniformly within a bounded region, and that inliers are distributed on a d-dimensional manifold within the same region. Consider a hyper-sphere of radius r, centered at a point on the manifold. The number of inliers within the hypersphere is proportional to r^d, whereas the number of outliers is proportional to r^n. Since $n > d$, this implies that as the radius decreases, the probability of finding an outlier decreases faster than the probability of finding an inlier. In NAPSAC, an initial data point $\mathbf{x}_0$ is selected at random. Next, the algorithm finds a set of points $S_{\mathbf{x}_0}$ such that all points in the set lie within a hyper-sphere of radius r, centered around $\mathbf{x}_0$. If the number of points in $S_{\mathbf{x}_0}$ is less than the size of the minimal sample, then this set is discarded, and the sampling process repeats afresh. Otherwise, points are selected uniformly from $S_{\mathbf{x}_0}$ until a minimal sample of size m has been selected.

One of the notable advantages of NAPSAC arises in the case of high dimensional models, where the probability of drawing an an uncontaminated sample becomes very low even for relatively uncontaminated datasets. On the other hand, one of the disadvantages of a NAPSAC style strategy is that enforcing the spatial proximity requirement can make the algorithm prone to degeneracies. In fact, specific techniques have been proposed to enforce exactly the *opposite* requirement – that samples are prevented from incorporating data points that are too close to each other [42], [43].

3.2.2 PROSAC

The Progressive Sample Consensus (PROSAC) algorithm [44] uses a measure of the quality of the data points in order to preferentially generate hypotheses that are more likely to be valid. In many computer vision problems, this measure of quality is readily available. For instance, when finding feature matches between a pair of images, a similarity function is typically evaluated over a number of tentative matches, and subsequently thresholded to obtain a set of putative correspondences. In this scenario, similarity scores can be used as a weak measure of correctness of the correspondences.

PROSAC can be viewed as a process that starts by deterministically testing the most promising hypotheses (generated from the most promising data points), gradually shifting to the sampling strategy of RANSAC as the confidence in the *a priori* sorting based on quality scores decreases. PROSAC is designed to draw the same samples as RANSAC, but in a more meaningful order. Consider a sequence of T_N samples of size m drawn by RANSAC from the set of all N correspondences. This sequence is denoted by $\{\mathcal{M}_i\}_{i=1}^{T_N}$. Let $\{\mathcal{M}_{(i)}\}_{i=1}^{T_N}$ denote a sequence of the same samples, but now sorted in descending order of sample quality. If we follow this ordering, then samples that are more likely to be good are drawn earlier. As the algorithm progresses, points with lower quality scores are gradually incorporated, and after T_N samples, all original RANSAC samples $\{\mathcal{M}_i\}_{i=1}^{T_N}$ are drawn.

In practice, the PROSAC approach has been shown to achieve significant computational savings, since good hypotheses are generated early on in the sampling process. However, while the ordering based on quality scores is less "local" than the NAPSAC strategy, it has been observed that invariably, PROSAC too needs safeguards against degeneracies [18]. In addition, when the quality scores are less helpful, for instance in the case of scenes with significant repetitive structure, the gains obtained with PROSAC are less significant [18], [45].

3.2.3 GroupSAC

While the NAPSAC strategy aimed to exploit the observation that inliers tend to be "closer" to each other than outliers, the recent GroupSAC algorithm [45] uses a more general version of this observation – that inliers are often more "similar" to each other. Assuming that we can separate data points into a number of groups that are similar according to some criterion, GroupSAC is a sampling strategy that aims to exploit this grouping.

In GroupSAC, the data points are partitioned into a set of groups $\{G_i\}$, $i = 1...K$, with the intuition that these groups tend to have either a high or a low fraction of inliers. Considering the feature matching problem for an image pair, groups may be obtained by optical flow based clustering. The assumption here is that the largest, more consistent clusters tend to have a higher fraction of inliers. As in PROSAC, sampling scheme is devised that starts by testing a small subset of the most promising groups, and gradually expands this to include all points.

While GroupSAC was shown to improve sampling efficiency, its applicability hinges on finding a grouping that makes sense in the context of the problem being considered. Furthermore, since the grouping stage is a part of the robust estimation module, the particular grouping strategy used (e.g., optical flow, image segmentation, etc.) should itself be very efficient, so as to not appreciably increase the overall runtime.

Stage 1b: Sample check

Once a minimal sample has been generated, the next step in standard RANSAC is to compute model parameters from this minimal sample. In some cases, however, it may be possible to insert a simple test to first check whether the sample is suitable for computing model parameters. For example, when computing a homography from a sample of four point correspondences, it is possible to incorporate chirality constraints to immediately rule out incompatible arrangements of the points in each image. More specifically, if the sign of the product of oriented area is not the same for the four points in each image, then the sample can be discarded. Note that this simple test requires very little overhead, particularly when compared to the expensive model generation and verfication stages.

3.3 Generate minimal sample model(s) (Stage 2)

Stage 2a: Model generation

In this step, model parameters are computed from the minimal subset. Note that in the minimal case, a single sample could result in the generation of multiple solutions, each of which needs to be evaluated by the algorithm. It is also worth noting that since the model parameters are generated from minimal samples, the influence of noise is the largest possible. Consequently, it is advisable that these minimal sample models should be used only after some form of refinement, either within the algorithm, or through a *post hoc* procedure.

Stage 2b: Model check

In standard RANSAC, once a model has been generated, its support is determined by verifying the model against all data points. It may be possible, however, to first introduce a preliminary test that checks the model based on application-specific constraints, and then performs the verification only if required. As a simple example, consider the case of fitting a circle to a set of points, where only circles with a certain range of radii are desired. By first testing the validity of the model parameters, it may be possible to completely skip the more expensive verification stage.

In [46], the computational benefit of a model check stage is demonstrated for the case of epipolar geometry estimation. By noting that for real cameras, only points in front of the camera are visible, it is possible to enforce *oriented* epipolar constraints via the model check stage. Each of the fundamental matrices is first tested to see whether it satisfies the oriented constraint. If this test is passed, the support of the fundamental matrix is computed as usual. If not, then the model may be discarded without further inspection. In [46], it was shown that, depending on the scene, anywhere between 10%-92% of the models could be rejected just based on this simple oriented constraint test.

3.4 Is the model interesting? (Stage 3)

In standard RANSAC, the goal of the model verification stage is to compute the support of the model by evaluating all data points, and to return the model that gathers the largest consensus set. We take a more general view, where the goal of this stage can be posed as providing an answer to the question: *is the model interesting*? Models generated from contaminated samples have arbitrary model parameters, and thus reveal no useful information about any data points that are not contained in the minimal sample itself. These models can be viewed as "non-interesting", and the goal of the verification process is to filter out these models. On the other hand, "interesting" models, or those that are likely to lead to the largest consensus set, are worth investigating further. In the Universal RANSAC formulation, the notion of a model being "interesting" has two components: (1) the model is likely to gather large support (2) the model is non-degenerate. While RANSAC seeks the answer to this question, the standard formulation is limited by the assumptions made (refer to the beginning of Section 3) as well as efficiency concerns. In the USAC framework, the objective is to answer this question while improving upon both the efficiency and the robustness of the standard algorithm.

Stage 3a: Model verification

The runtime of RANSAC can expressed as

$$t = k(t_M + \bar{m}_S t_V) \tag{6}$$

where k is the number of samples, m_S is the number of models per minimal sample, and t_M and t_V denote the time required to compute model parameters, and to verify the model, respectively. If for convenience, the time taken to verify a single point is chosen as the unit of time, then for RANSAC, $t_V = N$, where N is the number of data points. When N is large, the verification stage can thus consume a large fraction of the total runtime. Note, however, that the number of uncontaminated samples generated in RANSAC is typically small. In turn, this implies that almost almost all the hypothesized models are likely to be contaminated. If we assume that these contaminated models will be consistent with only a few data points, then it may be possible to discard "bad" models early on in the verification stage.

The techniques we discuss in this section propose variations of the same basic idea: conduct a statistical test on a small number of data points, and discard or accept the model based on the results of the test. The basic statistical test can be generalized as follows: given a model from the hypothesis generation stage, determine whether the model is "good" – i.e., it leads to the solution with maximal support, or it is "bad" – i.e., one of the data points in the sample is an outlier. The property "good" is a hidden state that is not directly observable, but is statistically linked to observable events. In the case of RANSAC, the observable events are the results of the individual point evaluations.

3.4.1 The $T_{d,d}$ Test

In [47], model verification is first performed using a subset of d randomly selected points (where $d \ll N$). The remaining $N - d$ points are evaluated only if the first d points are all inliers to the model. An expression for the number of verified correspondences per test, t_V, can be derived as a function of d, by considering two cases: the result of the test on a good vs. bad samples:

$$\begin{aligned} t_V(d) \;&=\; P_g((1-\alpha)N + \alpha\bar{t}_\alpha) \\ &\quad +(1-P_g)(\beta N + (1-\beta)\bar{t}_\beta) \end{aligned} \tag{7}$$

where P_g is the probability of drawing a good sample, α is the probability that a good sample does not pass the pre-verification test, and β is the probability that a bad sample passes the test. $\bar{t}_\alpha$ and $\bar{t}_\beta$ represent the average number of points tested in the two cases. Note that the efficiency of the test hinges on the relation $\beta \ll (1 - \alpha)$, i.e., that a bad sample should ideally be consistent with far fewer points than a good sample. The value α may be approximated as $1 - \varepsilon^d$ and β as δ^d, where δ is the probability that a data point is consistent with an incorrect model. In [47], an optimal setting of $d = 1$ was derived by minimizing the average time spent in the pre-verification step. In other words, the $T_{1,1}$ test checks a model against a randomly drawn data point. If the model is consistent with the point, verification continues. If not, the model is discarded and a new sample is generated. Note that a valid hypothesis may be mistakenly rejected by the $T_{d,d}$ test. Thus, one of the consequences of this approach is that it requires many more hypotheses than the original RANSAC. However, providing the hypothesis generation step is not too expensive, the overall runtime is typically reduced due to the preverification procedure [17], [18], [48].

3.4.2 Bail-Out Test

The idea of early termination of bad hypotheses was further extended in [49]. Given a model to be scored, a randomly selected subset of n points is evaluated against the model. If the observed inlier ratio within this subset, ε_n, is significantly less than the best inlier ratio observed so far, ε^*, then it is unlikely that the current model will yield a larger consensus set than the current maximum and can be discarded. More formally, given that I_n inliers have been seen within a subset of size n, the probability that the total number of inliers $\bar{I}$ for the current hypothesis will be greater than the current best inlier count, I^* is given by

$$p_{conf} = p(\bar{I} > I^*) = \sum_{\bar{I}=I^*}^{N} p(\bar{I}|I_n, n, N) \tag{8}$$

When this probability drops below a certain threshold (e.g., 1%), the model can be discarded. Since equation (8) is difficult to directly compute, an alternative is presented. The distribution of the number of inliers I_n in a subset of n points follows a hyper-geometric distribution. The idea behind the bail-out test is to compare the number of inliers seen within a subset of n points against a lower bound I_n^{min}. If $I_n < I_n^{min}$, then bail-out occurs. In order to compute I_n^{min}, note that the hypergeometric lower bounds may be approximated either using a binomial distribution for small values of n, or as a normal distribution for large values of n.

It was shown in [17], [18], [49] that the bail-out test typically results in a factor of 2-7 improvement in performance compared to standard RANSAC. Note, however, that the strategy outlined in [49] does not account for the fact that a good hypothesis might be incorrectly rejected by the bail-out test. As noted in [17], computing this probability is intractable. Thus, while the test is effective in practice, it is not, in a general sense, optimal.

3.4.3 SPRT test

Most recently, an optimal randomized model verification strategy was described in [17], [50]. The test is based on Wald's theory of sequential testing [51]. The theory was first applied for quality control in industrial inspection, with the goal of deciding whether a batch of products was "good" or "bad", while making the smallest number of observations possible. The verification stage of RANSAC has a similar goal: to decide whether a model is "good" (H_g) or "bad" (H_b). Wald's SPRT test is a solution of a constrained optimization problem, where the user supplies acceptable probabilities for errors of the first type (rejecting a good model) and the second type (accepting a bad model) and the resulting optimal test is a trade-off between the time to decision, and the errors committed. The SPRT test is based on the likelihood ratio

$$\lambda_j = \prod_{r=1}^{j} \frac{p(x_r|H_b)}{p(x_r|H_g)} \tag{9}$$

where x_r is equal to 1 if the r-th data point is consistent with a given model, and 0 otherwise. $p(1|H_g)$ denotes the probability that a randomly chosen data point is consistent with a good model, and this can be approximated by the inlier ratio ε. Similarly, $p(1|H_b)$ is the probability that a randomly chosen data point is consistent with a bad model, and this can be modeled using a Bernoulli distribution with parameter δ. If, after evaluating j data points, the likelihood ratio (9) becomes greater than some threshold A, the model is rejected.

The decision threshold A is the main parameter of the SPRT test and can be set to achieve optimal runtime, assuming that the two parameters ε and δ are known *a priori*. In practice, however, these parameters are typically unknown, and have to be estimated during the evaluation process, adjusting the value of the threshold A, based on current estimates. For the case of multi-view geometry problems, for instance, an initial estimate of the parameter δ can be obtained through geometric constraints. An initial value of ε can be estimated from the maximum number of RANSAC iterations that the user is willing to perform, and this may be updated by using the size of the largest support found so far.

In the multi-view geometry estimation experiments in [17], [18], it was shown that the randomized verification procedure based on the SPRT test results in a factor 2-9 runtime improvement over standard RANSAC. While the performance of the bail-out and SPRT tests are comparable for the case of higher inlier ratios (where the time to solution is low), the SPRT test was found to perform approximately 20% faster than the bail-out test for more challenging problems with lower inlier ratios.

3.4.4 Preemptive verification

The verification schemes discussed above are all *depth-first* in nature, meaning that a particular model is completely evaluated before moving on to the next model. Note that this approach is not directly applicable in the case where real-time performance is desired, since both the number of models, as well as the number of evaluations per model, may be arbitrary. In [52], a *breadth-first* formulation called preemptive RANSAC was introduced, where a fixed number of model hypotheses are generated beforehand and scored on a subset of the data points. Subsequently, the models are reordered based on these results, and a fraction of the models propagate to the next round of evaluation. This lends itself naturally to real-time applications, with a bound on the acceptable runtime. This approach was further improved upon in [18], where the Adaptive Real-Time Random Sample Consensus (ARRSAC) algorithm was introduced. ARRSAC retains the benefits of both breadth- and depth-first approaches (i.e., bounded run-time and adaptivity), and enables real-time operation over a wider range of inlier ratios.

It is important to note that the objectives of standard RANSAC and real-time RANSAC are significantly different. The goal in RANSAC is to find, with predefined confidence η_0, the "best" model according to some cost function. The goal in real-time RANSAC is to find the best model within a fixed time budget. Thus, there may exist cases where a good solution is simply out of the reach of these real-time methods.

Stage 3b: Degeneracy

In general, data is said to be degenerate if it does not provide sufficient constraints to compute a unique solution. Since RANSAC, in its standard form, does not have a safeguard against degeneracy, this can lead to incorrect solutions being returned by the algorithm.

A recent example that shows the utility of a test for degeneracy is that of DEGENSAC [53], where a detailed derivation is given for a nonlinear problem of epipolar geometry estimation. In this case, a notable example of degeneracy is that of points lying on a scene plane. In particular, when five of the seven image correspondences comprising the minimal sample for fundamental matrix estimation lie on the same plane in 3D, then the resulting model is consistent with all points that lie on the same plane. Consequently, in the case of images containing a dominant scene plane, an incorrect solution is often returned by the algorithm. To deal with this situation, a specific test was devised that aimed to identify samples where five or more correspondences in a minimal sample are related by a homography. The corresponding models are then subject to a model completion step, where the goal is to find, if possible, non-degenerate inliers that could be used to compute the correct model, given knowledge of the degeneracy.

A more general technique for dealing with degeneracy was proposed in [54], which describes a framework for RANSAC with (quasi-) degenerate data (QDEGSAC). The quasi-degenerate problem occurs when a majority of the data is degenerate, and thus does not provide enough constraints to compute the model uniquely, and only a few non-degenerate data points provide the remaining constraints. However, the probability of these few points being sampled is considerably low, and thus RANSAC will most often return the degenerate solution. QDEGSAC provides a solution to this problem for the case of estimating linear relations. Given a data matrix A, whose rows contain the constraints provided by the data points used to estimate the relation, the QDEGSAC framework can be viewed as a process that robustly measures the rank r_A of the data matrix, by using a sequence of calls to the RANSAC algorithm. The first execution estimates the most general model, which is appropriate for non-degenerate data. Following this, a series of RANSAC runs are performed successively on the inliers of the previous run, each run adding a linear constraint on the data. Finally, the most constraining model that successfully explains at least 50% of the original inliers to the first RANSAC is returned as the solution. Note that QDEGSAC is not a component of RANSAC; rather, it uses a number of RANSAC runs to effectively perform robust model selection. In the context of Figure 1, for instance, QDEGSAC can be thought of as a wrapper around the USAC framework.

3.5 Model refinement (Stage 4)

In general, data is said to be degenerate if it does not provide sufficient constraints to compute a unique solution. Since RANSAC, in its standard form, does not have a safeguard against degeneracy, this can lead to incorrect solutions being returned by the algorithm.

A recent example that shows the utility of a test for degeneracy is that of DEGENSAC [53], where a detailed derivation is given for a nonlinear problem of epipolar geometry estimation. In this case, a notable example of degeneracy is that of points lying on a scene plane. In particular, when five of the seven image correspondences comprising the minimal sample for fundamental matrix estimation lie on the same plane in 3D, then the resulting model is consistent with all points that lie on the same plane. Consequently, in the case of images containing a dominant scene plane, an incorrect solution is often returned by the algorithm. To deal with this situation, a specific test was devised that aimed to identify samples where five or more correspondences in a minimal sample are related by a homography. The corresponding models are then subject to a model completion step, where the goal is to find, if possible, non-degenerate inliers that could be used to compute the correct model, given knowledge of the degeneracy.

A more general technique for dealing with degeneracy was proposed in [54], which describes a framework for RANSAC with (quasi-) degenerate data (QDEGSAC). The quasi-degenerate problem occurs when a majority of the data is degenerate, and thus does not provide enough constraints to compute the model uniquely, and only a few non-degenerate data points provide the remaining constraints. However, the probability of these few points being sampled is considerably low, and thus RANSAC will most often return the degenerate solution. QDEGSAC provides a solution to this problem for the case of estimating linear relations. Given a data matrix A, whose rows contain the constraints provided by the data points used to estimate the relation, the QDEGSAC framework can be viewed as a process that robustly measures the rank r_A of the data matrix, by using a sequence of calls to the RANSAC algorithm. The first execution estimates the most general model, which is appropriate for non-degenerate data. Following this, a series of RANSAC runs are performed successively on the inliers of the previous run, each run adding a linear constraint on the data. Finally, the most constraining model that successfully explains at least 50% of the original inliers to the first RANSAC is returned as the solution. Note that QDEGSAC is not a component of RANSAC; rather, it uses a number of RANSAC runs to effectively perform robust model selection. In the context of Figure 1, for instance, QDEGSAC can be thought of as a wrapper around the USAC framework.

4 IMPLEMENTATION

In Section 3, we introduced a Universal RANSAC framework to provide a common context within which to analyse various variants of RANSAC. However, it becomes immediately clear that this framework can also provide the basis for a high performance robust estimator that addresses many of the limitations of RANSAC within a single unified system. Thus, one of the goals of our work is to provide a stand-alone C++ implementation that implements the Universal RANSAC framework presented in the previous section. More specifically, while Section 3 discussed a number of alternatives for each stage of the framework, we now describe a specific implementation

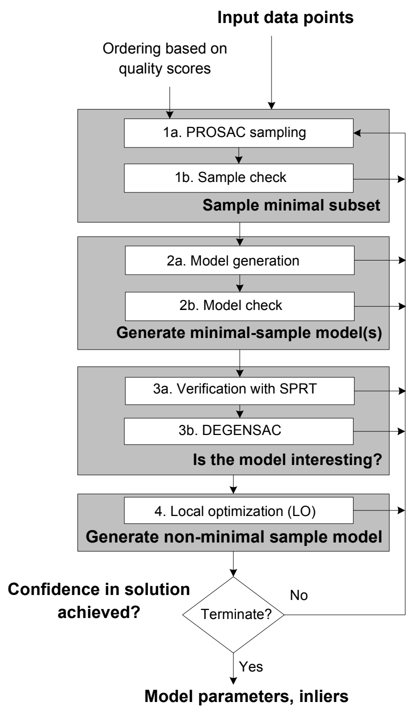

Input data points
Ordering based on quality scores
1a. PROSAC sampling
1b. Sample check
Sample minimal subset
2a. Model generation
2b. Model check
Generate minimal-sample model(s)
3a. Verification with SPRT
3b. DEGENSAC
Is the model interesting?
4. Local optimization (LO)
Generate non-minimal sample model
Confidence in solution achieved?
Terminate?
No
Yes
Model parameters, inliers

Fig. 2: USAC-1.0 implementation. We present a high-performance implementation of the general framework of Figure 1, by use of state of the art algorithms for the individual modules. The combination of these techniques results in a robust estimator that address many of the limitations of standard RANSAC within a unified software package. In addition, by optionally invoking the individual modules in isolation (e.g., just PROSAC sampling), the implementation can be modified to behave like the corresponding variant of RANSAC.

that uses an appropriate choice of algorithm for each module, keeping in mind concerns about robustness, performance and generality. These algorithmic choices are also illustrated in Figure 2. We call this implementation USAC-1.0 to emphasize that it is based on current state of the art techniques, and can be extended as new advances emerge. The implementation can be accessed at: http://cs.unc.edu/~rraguram/usac.

By making the implementation freely available, it is our hope that the system will be extended adding implementations of existing alternatives to the techniques implemented, or by developing new variants. In addition, once a new algorithm has been incorporated, its performance can easily be compared to other techniques using the presented framework and benchmarks. In summary, this implementation will be of use to other researchers as a standalone, high-performance tool for robust estimation; as a starting point for new RANSAC variants; or as a benchmark for the evaluation of new techniques.

4.1 Sample minimal subset

Given a (possibly prefiltered) set of data points, the goal of this step is to generate a sequence of minimal samples. From the discussion in Section 3.2, it is clear that the various sampling strategies proposed in the literature all come with trade-offs. In particular, all these sampling strategies invariably involve the preferential selection of samples that are in sensitive spatial configurations. Consequently, it is advisable to couple this the non-uniform sampling stage with modules that handle degeneracy detection and model refinement.

In our implementation, we choose the PROSAC (Section 3.2.2) sampling strategy for the sample generation stage. This choice was made to reach a compromise between performance, generality and susceptibility to degenerate configurations. PROSAC is more easily applicable in the general case than GroupSAC, and less susceptible to degenerate configurations than NAPSAC. PROSAC requires an additional input in the form of ordering information for the data points in terms of their quality scores, which is often available. In addition, while our implementation does not rely on it, once a minimal sample has been selected, problem-specific checks can be enforced to eliminate undesirable minimal samples, as outlined in Section 3.2 (Stage 1b).

4.2 Generate minimal sample model(s)

The process of model generation and checking are both dependent on the problem under consideration. While the complexity of model generation lies outside the scope of USAC, it should be noted that incorporating the model checking stage can significantly reduce the number of models that propagate to subsequent stages of the algorithm.

4.3 Is the model interesting?

The goal of this step is to efficiently verify the model, and also to check for degeneracy. We perform this step using the SPRT test outlined in Section 3.4.3, which represents the optimal strategy for randomized model verification. This test requires two additional input parameters, namely ε and δ. As noted in Section 3.4.3, these parameters can be initialized conservatively, and are automatically updated as the algorithm progresses.

We choose to employ a specific degeneracy detection module that calls a user provided function to check for model degeneracy once it has been verified to be the current best model. As seen in the case of DEGENSAC (Section 3.4.4), this is with a view towards increasing robustness while maintaining computational efficiency. Note, however, that integrating the USAC algorithm with QDEGSAC is simple: the sequence of calls to RANSAC that occur within QDEGSAC need to be simply replaced by calls to USAC.

4.4 Generate non-minimal sample model

For refining the current best model, we use the Lo-RANSAC approach (Section **??**), due to its generality. We have found that the inner-RANSAC approach, coupled with an iterative reweighted least squares procedure gives good results in practice, while not adding appreciably to the computational cost. In our implementation, we make an additional optimization in the interests of efficiency: before performing the local optimization step, a check is performed to determine the extent to which the current inlier set overlaps with the best inlier set found so far. If this overlap is substantial (e.g., $\geq 95\%$), then it is unlikely that the local optimization will significantly improve upon the best result, and this step can be skipped. The effect of this enhancement is to prevent excess time from being spent in the computation of a locally refined solution.

4.5 Confidence in solution achieved?

As discussed in Sec. 2.2.3, the standard RANSAC algorithm terminates when the probability of finding a set of inliers that is larger than the best support found so far falls under a predefined threshold η_0. For RANSAC, this probability can be computed as

$$\eta_R = (1 - 1/\varepsilon^m)^k, \tag{10}$$

where k is the number of samples drawn so far. This gives rise to the standard stopping criterion in equation (3). Two important modifications to this stopping criterion are discussed below.

Non-uniform sampling:
The use of non-uniform sampling results in good samples being drawn earlier on in the process. However, if the standard stopping criterion from equation (10) is used with ε being computed over the whole dataset, then the total number of samples drawn will be the same as in RANSAC. We thus consider two factors to determine when the algorithm should stop: non-randomness and maximality. The idea is to look for a *stopping length* n^* that satisfies both constraints.

The probability of having i random inliers *by chance* within the subset $\mathcal{U}_n$ is given by a binomial distribution

$$p_n(i) = \delta^{i-m}(1-\delta)^{n-i}\binom{n-m}{i-m}, \tag{11}$$

where δ is the probability of a random point being consistent with a contaminated model. For each n, the minimum size of "non-random" support is calculated as

$$I_n^{min} = min\{j : \sum_{i=j}^{n} p_n(i) < \psi\}, \tag{12}$$

where the inequality represents a simple p-value test to determine, for each n, the smallest support such that the probability of randomness falls below ψ (set to 0.05 for a 5% significance test). The stopping length n^* must have $I_{n^*} \geq I_n^{min}$

The maximality constraint is again given by equation (10), and is used to ensure, with some confidence, that no better solution exists within the set $\mathcal{U}_n$. Thus $\eta = (1 - 1/\varepsilon_n^m)^{k_n}$, where ε_n is given by I_n/n. The stopping length n^* is chosen to minimize the number of samples k_{n^*}, subject to the non-randomness constraint.

Randomized verification:
When a randomized verification strategy is adopted, there is a chance that good models may be erroneously rejected by the verification step. Thus, the stopping criterion must now account for this by drawing additional samples to achieve the same confidence. Since we use the SPRT test, the probability of finding a better model than the current best one is given by

$$\eta = (1 - (1-\alpha)/\varepsilon^m)^k, \tag{13}$$

where α is the probability that a "good" model is rejected by the SPRT. In practice, α is not constant, being reestimated based on current estimates of the test parameters, ε and δ. Given an SPRT threshold A_i computed using the current estimates of ε_i and δ_i, the probability of rejecting a good model, with inlier fraction ε, is $\alpha_i = A_i^{-h_i}$, where h_i is computed from the relation

$$\varepsilon\left(\frac{\delta_i}{\varepsilon_i}\right)^{h_i} + (1-\varepsilon)\left(\frac{1-\delta_i}{1-\varepsilon_i}\right)^{h_i} = 1. \tag{14}$$

This implies that whenever the value of the SPRT threshold A_i is updated, this results in a new α_i. Given the l^{th} SPRT test, the probability η from equation (13) is now given by the product over all individual tests, as

$$\eta = \prod_{i=0}^{l}(1 - (1 - A_i^{-h_i})/\varepsilon^m)^{k_i}. \tag{15}$$

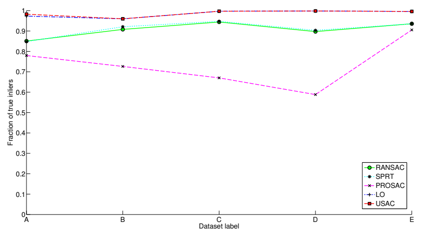
5 Evaluation

In this section, we evaluate the performance of the USAC-1.0 implementation against current state of the art techniques, on a variety of estimation problems. Note that while robust estimators are perhaps most frequently applied in geometric vision problems, USAC is a general robust estimator, meaning that it can be used in virtually any scenario where models are being estimated from contaminated data measurements.

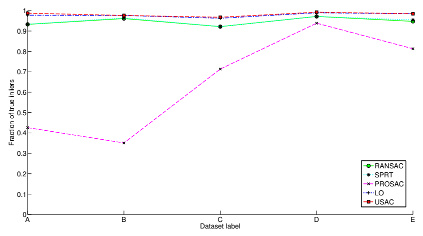
One interesting feature of the software library we provide is that it has a modular structure, and can be easily configured to behave in a number of different ways. In particular, as noted in Section 3, many of the important RANSAC variants can be interpreted as special cases of this implementation. In its most reduced form, the implementation behaves exactly like the standard RANSAC algorithm outlined in Algorithm 1. By "switching on" the individual modules of Figure 2, it becomes possible to independently evaluate the effect that an specific module (e.g., PROSAC based non-uniform sampling) has on the estimation results. The choice of modules and parameters can be easily specified using a common configuration file. In other words, the implementation in Figure 2 not only represents USAC-1.0, but, it also in effect provides an implementation of a number of RANSAC variants, all within a single, easily configurable software package. For the experimental comparisons in this Section, we thus configured our implementation to represent state of the art robust estimators. In particular, we compare the performance of USAC-1.0 against the following:

1) RANSAC: the standard RANSAC algorithm, as laid out in Alg. 1. This denotes the baseline for comparison.
2) SPRT: RANSAC with optimal randomized verification, which represents the state of the art for efficient model verification.
3) PROSAC: RANSAC with non-uniform sampling based on ordering data points by quality scores, leading to efficient hypothesis generation.
4) LO: RANSAC with local optimization, resulting in more accurate solutions.

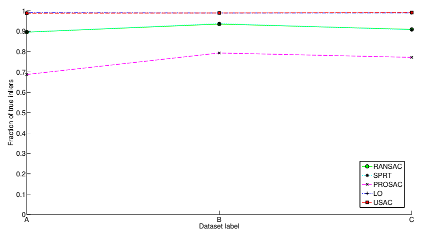
In all experiments, since the ground truth is unknown, the true inlier ratios (along with the ground truth inliers) were estimated by performing 10^7 evaluations of random models, followed by a refinement step. Following this, the inliers were manually inspected to ensure validity of the data points and



(a) Homography

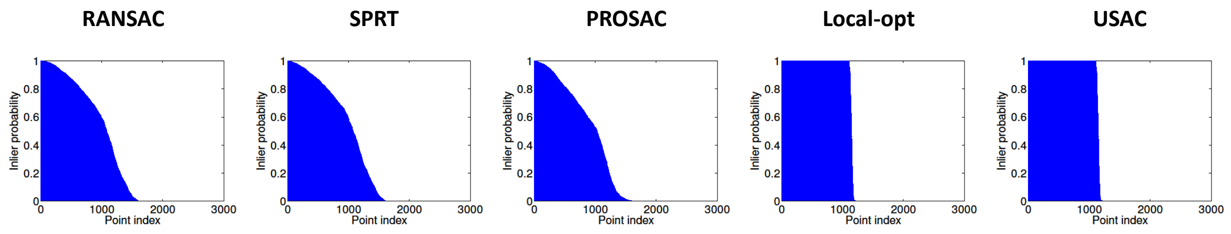


(b) Fundamental matrix

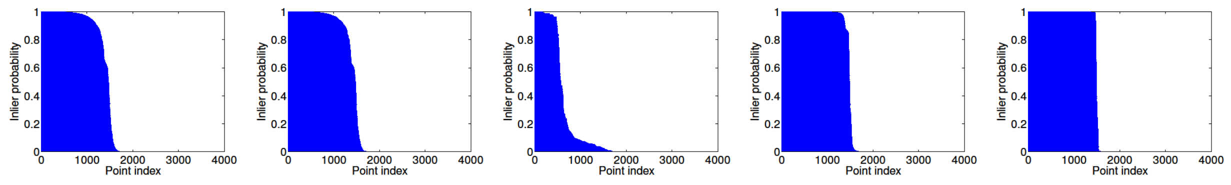


(c) Essential matrix

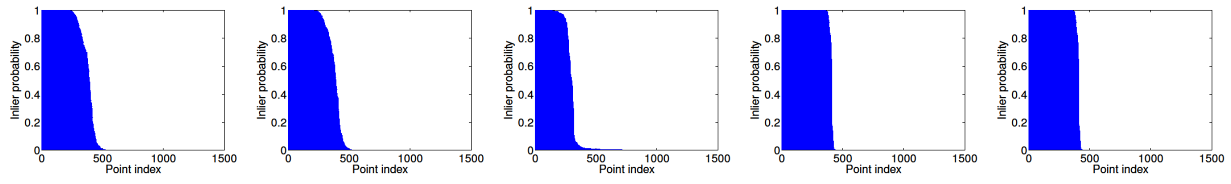
Fig. 3: Fraction of true inliers returned by each algorithm for three estimation problems. The labels on the X-axis correspond to the datasets in Tables 1, 2, and 3. The plots represent average values over 500 runs, for each algorithm and dataset. Note that for all estimation problems, USAC-1.0 consistently returns close to 100% of the true inliers. The USAC-1.0 and LO curves are virtually identical, as are the RANSAC and SPRT curves. PROSAC, while fast, often returns solutions that have fewer inliers.



(a) Homography: **A**



(b) Fundamental matrix: **A**



(c) Essential matrix: **A**

Fig. 4: Visualization of the stability of the estimation results for each of the five algorithms evaluated. The results are shown for (a) homography estimation (dataset **A** from Table 1) (b) fundamental matrix estimation (dataset **A** from Table 2) (c) essential matrix estimation (dataset **A** from Table 3). In each graph, for every feature correspondence in the dataset (x-axis), we determine the fraction of runs, out of 500, where that correspondence is classified as an inlier (y-axis). The points on the x-axis are sorted in decreasing order of these probabilities. In the ideal case, each true inlier would have a score of 100%, and each true outlier would have a score of 0%. It can be seen that RANSAC and SPRT show comparable trends, while PROSAC is often significantly worse, missing true inliers on many of the runs. The results for LO and USAC-1.0 are very stable, and correspond closely to the ideal situation.

to verify that no outliers were erroneously included. In all experiments, the results we report are average values obtained over 500 executions of each algorithm.

5.1 Homography

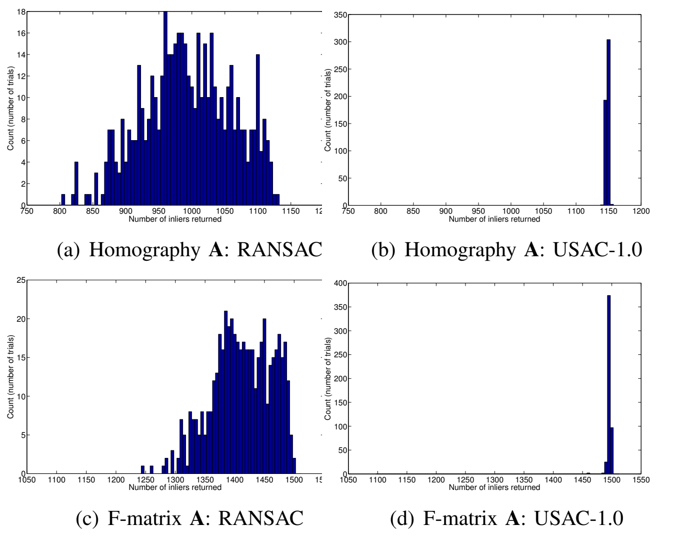
We evaluate the performance of USAC-1.0 for the problem of 2D homography estimation. The results are tabulated in Table 1. The inlier ratios in the data range from 10%-46%, making this a challenging dataset for robust estimation algorithms. The table lists, for each algorithm, the number of inliers found, the number of hypotheses and models, the number of verifications per model, and the total runtime (in milliseconds). For USAC-1.0, the table also lists the number of hypotheses/models that are rejected by the sample/model check steps from Figure 2 (steps 1b and 2b). To determine the accuracy of the solution, we compute the Sampson error [20] and report the root mean square (RMS) value over 500 executions of each algorithm.

It can be seen from the results that USAC-1.0 consistently delivers solutions that capture the most inliers, and which are the most accurate in terms of mean error. This is due to the local optimization step incorporated in the algorithm. In addition, the non-uniform sampling, optimized model verification and sample/model check modules all result in significant computational savings. In particular, USAC-1.0 generates orders of magnitude fewer samples than standard techniques, and evaluates fewer points per model. The overall runtime is thus on the order of a few milliseconds, even for very low inlier ratios, representing effective speedups ranging between 5x-520x compared to RANSAC. It is worth noting that the combination of techniques in USAC results in consistently good performance, even when individual techniques are relatively ineffective – for e.g., in example **E** in Table 1, where PROSAC-based sampling results in a significant runtime, while USAC-1.0 remains very efficient due to the other optimizations present.

USAC-1.0 in general has much lower runtimes compared to the other techniques tested, save for PROSAC, which can sometimes be slightly faster than USAC-1.0. However, note that the solutions delivered by PROSAC can be unstable, due to the fact that the points selected are often poorly distributed in the image (e.g., the highest ranking points are typically spatially very close to each other). In USAC-1.0, this effect is mitigated by the local optimization step, which incurs a small additional computational cost but provides much more accurate and stable results. This effect is illustrated in Figure 3(a), which shows the fraction of true inliers that are returned by each algorithm. Note from the graphs that USAC-1.0 typically returns a significant fraction of the inliers, while the corresponding values for PROSAC are often much lower. This is due to the fact that the local optimization step uses non-minimal samples to improve the accuracy of the model parameter estimates, thus resulting in a larger consensus set. By restimating model parameters using the set of all inliers, the local optimization is able to "break out" of the spatially localized fits provided by PROSAC-based sampling.

Another visualization of the stability of the results may be obtained by running each robust estimator multiple times on the same dataset, and calculating the fraction of runs in which a particular correspondence is classified as being an inlier. For homography estimation, this visualization is presented in Figure 4(a), for dataset **A** from Table 1. In each graph, for each input feature correspondence (x-axis), we plot the fraction of runs in which that correspondence is classified as an inlier (y-axis). This provides an empirical estimate of the probability that a true inlier is correctly classified by each robust estimation algorithm. The points on the x-axis are sorted in decreasing order of these probabilities. In the ideal case, each true inlier would have a score of 100%, and each true outlier would have a score of 0% . It can be seen that RANSAC, SPRT and PROSAC show comparable trends; each of these algorithms misses some inliers on each run, thus further underscoring the randomized nature of these methods. On the other hand, the results for LO and USAC-1.0 are very similar, and correspond closely to the ideal situation. This is an indication that these methods successfully classify a majority of the inliers on every run of the algorithm and are thus very stable.

Finally, note that the standard deviation of the number of inliers returned by USAC-1.0 is significantly lower than that of RANSAC. Figures 5(a) and 5(b) compare the histograms of retrieved inlier counts over 500 runs of RANSAC and USAC-1.0, for image pair **A** from Table 1. It can be seen that the number of inliers returned by RANSAC can vary quite widely over different runs, while the histogram for USAC-1.0 is much more "peaked", implying that a much more stable solution is returned in the vicinity of the global maximum. This again, is due to the application of local optimization, which by using non-minimal samples to compensate for inlier noise, ensures that the model parameters are more stable. This also indicates that while computationally very expensive techniques such as [34] can be used to maximize the consensus set, USAC-1.0 achieves approximately similar results, at a small fraction of the computational cost. USAC-1.0 is thus able to deliver more accurate and stable solutions than current techniques, while doing so extremely efficiently.



(a) Homography **A**: RANSAC (b) Homography **A**: USAC-1.0

(c) F-matrix **A**: RANSAC (d) F-matrix **A**: USAC-1.0

Fig. 5: Histogram of inlier counts for RANSAC vs. USAC-1.0, computed over 500 trials of each algorithm. Each graph shows the number of inliers returned on the x-axis and how many times this number was reached (in 500 runs of the algorithm) on the y-axis. The graphs show results for homography estimation (a-b) for dataset **A** from Table 1 and fundamental matrix estimation (c-d) for dataset **A** from Table 2. Note that the histogram of inlier counts is more spread out for RANSAC, indicating that different inlier maxima are reached in different runs, owing to the randomized nature of the algorithm. USAC-1.0, on the other hand, has a much more peaked distribution, implying that the correct maxima is being reached on most executions.

5.2 Fundamental matrix

We evaluate the performance of USAC-1.0 for the problem of fundamental matrix estimation using the 7-point algorithm. These results are tabulated in Table 2. The inlier ratios in the data range from 22%-92%, and examples were chosen to cover a range of challenging cases, including narrow and wide baselines, scale change and dominant scene planes (a degenerate configuration for the fundamental matrix). The table lists the same statistics as for homography estimation.

As before, it can be seen from the results that USAC-1.0 consistently delivers stable and accurate solutions with low overall runtimes, with up to 4x-7000x speedups compared

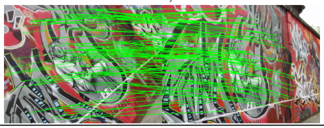
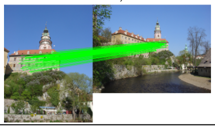
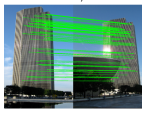
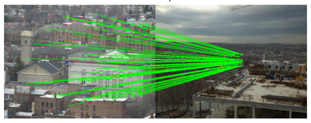
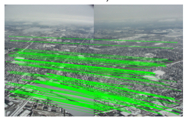
		RANSAC	SPRT	PROSAC	Local-opt	USAC-1.0
A : $\varepsilon = 0.46, N = 2540$ 	I	994±68	991±72	911±160	1137±27	1148±2
	error	1.71±0.21	1.62±0.41	2.14±0.93	1.17±0.12	1.04±0.00
	k	220	241	11	115	7/0
	models	220	241	11	115	7/0
	vpm	2540.0	80.3	2540.0	2540.0	426.7
	time	40.62	9.16	2.46	34.72	7.71
B : $\varepsilon = 0.15, N = 514$ 	I	70±4	71±4	56±11	74±0	74±3
	error	1.88±0.68	1.87±0.52	4.63±3.44	1.13±0.07	1.19±0.33
	k	16766	18800	11	11548	9/1
	models	16766	18800	11	11548	8/0
	vpm	514.0	25.4	514.0	514.0	110.4
	time	940.73	470.07	0.66	652.88	6.21
C : $\varepsilon = 0.23, N = 1317$ 	I	286±17	287±17	203±30	302±1	302±6
	error	1.63±0.44	1.63±0.54	8.49±9.12	0.89±0.06	0.89±0.13
	k	2433	2534	4	1717	4/0
	models	2433	2534	4	1717	4/0
	vpm	1317.0	18.4	1317.0	1317.0	374.1
	time	254.62	57.50	0.50	196.90	4.23
D : $\varepsilon = 0.34, N = 495$ 	I	151±11	152±10	99±24	168±0	168±0
	error	2.22±0.45	2.23±0.51	6.81±3.12	1.44±0.00	1.43±0.00
	k	663	669	9	354	8/2
	models	663	669	9	354	5/0
	vpm	495.0	15.1	495.0	495.0	124.0
	time	36.00	16.05	0.53	27.71	6.13
E : $\varepsilon = 0.10, N = 994$ 	I	93±6	93±5	90±7	99±0	99±0
	error	3.43±1.42	3.13±1.01	11.43±4.10	2.52±0.36	2.59±0.27
	k	75950	84329	12059	49533	7266/6511
	models	75950	84329	12059	49533	755/0
	vpm	994.0	30.8	994.0	994.0	38.0
	time	6532.22	2134.44	1034.98	4266.51	25.74

TABLE 1: Results for homography estimation. The table lists, for each algorithm, the number of inliers found (I), the error of the solution with respect to the "true" result (error), the number of samples (k) and models (models), the number of verifications per model (vpm), and the total runtime (time in milliseconds). For USAC-1.0, the table additionally lists, the number of samples and models that are rejected by the sample/model check tests. The results are averaged over 500 runs of each algorithm.

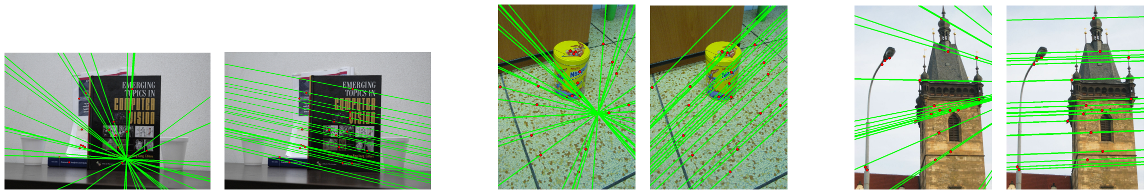


Fig. 6: Example degenerate (left) and non-degenerate (right) epipolar geometries for three scenes containing dominant scene planes. Figures (a), (c) and (e) are results produced by RANSAC, while (b), (d) and (f) are computed with USAC-1.0. The figures show some selected epipolar lines corresponding to inlier correspondences. Note that for the degenerate cases, the epipolar geometry is consistent with points on the dominant plane, but not with the (fewer) off-plane inliers.

to RANSAC. The case of fundamental matrix estimation is particularly challenging since care must be taken to avoid degenerate solutions. PROSAC, in particular, can be prone to degeneracies, since even when there is no dominant plane in the scene, the top ranked points often lie on the same surface. In urban scenes, this will often lead to a degenerate solution (or one that is poorly conditioned). Note, for instance, examples **A** and **B** in Table 2, where PROSAC returns solutions very quickly, but with high errors. The DEGENSAC module in USAC-1.0 (step 3b in Figure 2) is able to effectively detect and handle these cases. When the scene contains a dominant plane (example **E**), all techniques that do not account for degeneracy return incorrect solutions. This is further illustrated in Figure 6, which depicts sample epipolar geometries for degenerate and non-degenerate solutions provided by RANSAC and USAC-1.0, respectively. Note that for degenerate solutions, the epipolar lines are consistent with points on the dominant plane, but not with off-plane inliers. Figure 3(b) shows the fraction of true inliers that are returned by each algorithm. Note that USAC-1.0, due to the local optimization, typically returns all true inliers.

As in the case of homography estimation, Figure 4(b) provides a visualization of the stability of the results. Note that while the trends for RANSAC and SPRT are similar, PROSAC is significantly worse, since it is particularly prone to degeneracy. Thus, inliers are often misclassified on a significant fraction of the runs, as reflected in the corresponding plot. The results for LO and USAC-1.0 are again comparable, with USAC-1.0 producing slightly better results. This small improvement is again explained by the fact that USAC-1.0 is able to detect and recover from degenerate data configurations, whereas LO is sometimes trapped when the minimal sample contains points that lie on a scene plane. Finally, the histograms of inlier counts (Figures 5(c) and 5(d)) indicate that, as for the homography, USAC-1.0 returns very stable solutions due to the local optimization.

5.3 Essential matrix

We evaluate the performance of USAC-1.0 for the problem of essential matrix estimation using the 5-point algorithm [55]. These results are tabulated in Table 3, with inlier ratios in the data ranging from 26%-65%. We tabulate the same statistics as for the fundamental matrix, with the error again being given by the Sampson distance. The dominant plane degeneracy is not an issue for the 5-point method. However, the time to compute a minimal solution is more than that for the 7-point. In this context, note that even for very low inlier ratios (such as in example **F** in Table 3), USAC-1.0 is still able to deliver correct solutions with low runtimes – in fact, well within real-time, which is useful for real-time 3D reconstruction systems. Figure 4(c) again shows the relative stability of USAC-1.0, in regard to the probability of recovering true inliers, as compared to the baseline methods.

6 Conclusion

In this paper, we have presented a comprehensive overview of the state of the art in RANSAC-based robust estimation. To provide a unified context for this analysis, we propose a Universal RANSAC framework, which is a synthesis of the various optimizations and improvements that have been proposed to RANSAC over the years. This framework provides a common context within which to analyse current state of the art robust estimation algorithms, and to investigate the interplay between these varied algorithmic components.

As a second contribution of this work, we have developed a stand-alone C++ library that implements the Universal RANSAC framework using state of the art algorithms for each stage (USAC-1.0). The implementation is modular, and the effect of individual optimizations can be studied in isolation, as well as in combination with each other. We have evaluated its effectiveness on a challenging collection of estimation problems, thereby demonstrating the advantages of unifying the various RANSAC techniques . This provides state of the art performance, and can be used as a benchmark against which to compare newly developed robust estimation algoritms. The library can be used as a stand-alone tool for use in specific applications and, furthermore, provides an easily extendible base for developing new algorithms. By making the datasets and implementation freely available, it is our hope that this leads to a standardized evaluation framework for robust estimation algorithms, in addition to providing a framework for others to use and to build upon.

Acknowledgments

The authors are grateful to the sponsors of this research. R. Raguram and J-M. Frahm were supported by NSF grant IIS-0916829 and Department of Energy Award DE-FG52-08NA28778. O. Chum and J. Matas were supported by GACR P103/12/2310, and EC project FP7-ICT-270138 DARWIN. M. Pollefeys gratefully acknowledges the support of the 4DVideo ERC Starting Grant Nr. 210806.

References

[1] J. W. Tukey, *Exploratory Data Analysis*. Addison-Wesley, 1977.

[2] P. J. Huber, *Robust Statistics*. John Wiley & Sons, 1981.

[3] A. F. Siegel, “Robust regression using repeated medians,” *Biometrika*, vol. 69, no. 1, pp. 242–244, 1982.

[4] P. J. Rousseeuw, “Least median of squares regression,” *Journal of the American Statistical Association*, vol. 79, no. 388, pp. 871–880, 1984.

[5] P. Hough, “Method and means for recognizing complex patterns,” U.S. Patent 3.069.654, December 1962.

[6] J. Illingworth and J. Kittler, “A survey of the hough transform,” *Computer Vision, Graphics, and Image Processing*, vol. 44, no. 1, pp. 87–116, 1988.

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

[8] K. Tanaka and E. Kondo, “Incremental ransac for online relocation in large dynamic environments,” in *IEEE International Conference on Robotics and Automation*, May 2006, pp. 68 –75.

[9] N. Snavely, S. M. Seitz, and R. Szeliski, “Modeling the world from Internet photo collections,” *International Journal of Computer Vision*, vol. 80, no. 2, pp. 189–210, November 2008.

[10] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski, “Building Rome in a day,” in *International Conference on Computer Vision (ICCV)*, 2009.

[11] A. Torii, M. Havlena, and T. Pajdla, “From google street view to 3d city models,” in *International Conference on Computer Vision Workshops (ICCV Workshops)*, 2009, pp. 2188 –2195.

[12] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys, “Building rome on a cloudless day,” in *European Conference on Computer Vision (ECCV)*, vol. 6314, 2010, pp. 368–381.

[13] D. Nister, O. Naroditsky, and J. Bergen, “Visual odometry,” in *Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2004, pp. 652–659.

[14] E. Malis and E. Marchand, “Experiments with robust estimation techniques in real-time robot vision,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006.

[15] M. Pollefeys, D. Nistér, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. J. Kim, P. Merrell, C. Salmi, S. N. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, and H. Towles, “Detailed real-time urban 3d reconstruction from video,” *International Journal of Computer Vision*, vol. 78, no. 2-3, pp. 143–167, 2008.

[16] B. Clipp, J. Lim, J.-M. Frahm, and M. Pollefeys, “Parallel, real-time visual slam,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2010.

[17] O. Chum and J. Matas, “Optimal randomized RANSAC,” *IEEE Trans. PAMI*, vol. 30, no. 8, pp. 1472–1482, 2008.

[18] R. Raguram, J.-M. Frahm, and M. Pollefeys, “A comparative analysis of RANSAC techniques leading to adaptive real-time random sample consensus,” in *European Conference on Computer Vision (ECCV)*, 2008, pp. 500–513.

[19] P. H. S. Torr, A. Zisserman, and S. J. Maybank, “Robust detection of degenerate configurations while estimating the fundamental matrix,” *Computer Vision and Image Understanding*, vol. 71, no. 3, pp. 312–333, 1998.

[20] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[21] C. V. Stewart, “MINPRAN: A new robust estimator for computer vision,” *IEEE Trans. PAMI*, vol. 17, no. 10, pp. 925–938, 1995.

[22] J. Miller and C. V. Stewart, “MUSE: Robust surface fitting using unbiased scale estimates,” in *Computer Vision and Pattern Recognition (CVPR)*, 1996, pp. 300–306.

[23] K.-M. Lee, P. Meer, and R.-H. Park, “Robust adaptive segmentation of range images,” *IEEE Trans. PAMI*, vol. 20, pp. 200–205, 1998.

[24] H. Wang and D. Suter, “Robust adaptive-scale parametric model estimation for computer vision,” *IEEE Trans. PAMI*, vol. 26, no. 11, pp. 1459–1474, Nov 2004.

[25] H. Chen and P. Meer, “Robust regression with projection based m-estimators,” in *International Conference on Computer Vision (ICCV)*, 2003, pp. 878–.

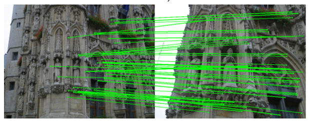
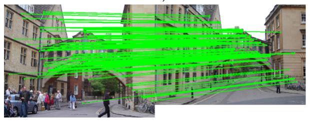
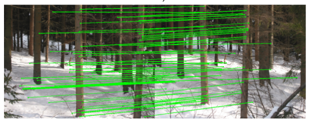
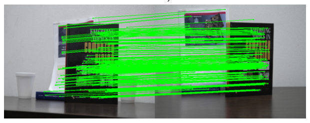
		RANSAC	SPRT	PROSAC	Local-opt	USAC-1.0
A : $\varepsilon = 0.48, N = 3154$	I	1412±50	1413±48	645±120	1480±33	1495±8
	error	1.28±0.58	1.26±0.55	21.31±11.14	0.85±0.50	0.63±0.24
	k	1420	1455	4	957	2/0
	models	3499	3587	10	2359	4/0
	vpm	3154.0	20.1	3154.0	3154.0	940.9
	time	255.90	18.19	2.01	191.16	16.84
B : $\varepsilon = 0.57, N = 575$	I	315±12	316±11	115±13	328±1	328±3
	error	0.71±0.18	0.69±0.17	60.99±13.05	0.45±0.04	0.45±0.06
	k	385	371	7	240	3/0
	models	941	906	17	588	8/1
	vpm	575.0	11.5	575.0	575.0	423.8
	time	14.98	5.35	1.35	11.22	5.12
C : $\varepsilon = 0.38, N = 1088$	I	381±13	381±12	295±16	398±11	406±4
	error	1.79±1.27	1.81±1.22	30.49±2.68	1.07±0.83	0.58±0.28
	k	7935	8403	4	5762	2/0
	models	19578	20735	9	14221	3/0
	vpm	1088.0	22.6	1088.0	1088.0	472.2
	time	546.53	108.77	5.41	408.26	21.32
D : $\varepsilon = 0.22, N = 1516$	I	324±10	324±9	313±19	333±3	334±3
	error	0.93±0.47	0.94±0.46	1.30±0.63	0.54±0.27	0.49±0.22
	k	267465	280961	6	198554	4/0
	models	661141	694471	16	490798	12/4
	vpm	1516.0	17.3	1516.0	1516.0	268.6
	time	23892.92	3434.69	1.77	18008.08	9.17
E : $\varepsilon = 0.92, N = 786$	I	685±37	690±36	588±0	712±31	722±0
	error	2.77±6.43	2.24±5.78	23.65±0.00	1.85±5.57	0.29±0.00
	k	14	15	1	11	1/0
	models	34	36	3	25	3/0
	vpm	786.0	299.8	786.0	786.0	675.7
	time	0.85	0.59	0.12	9.31	21.01

TABLE 2: Results for fundamental matrix estimation. The table lists the same summary statistics as in Table 1.

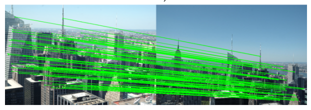
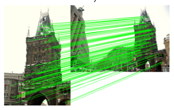
		RANSAC	SPRT	PROSAC	Local-opt	USAC-1.0
A : $\varepsilon = 0.35, N = 1207$	I	395±16	396±15	335±32	418±8	418±8
	error	2.21±1.48	2.47±1.74	9.27±9.55	0.99±0.59	0.98±0.54
	k	1321	1363	17	887	19/0
	models	5948	6139	73	3989	83/25
	vpm	1207.0	18.3	1207.0	1207.0	72.2
	time	522.77	193.18	6.71	348.49	10.11
B : $\varepsilon = 0.65, N = 1110$	I	646±18	646±17	496±14	715±3	713±4
	error	2.94±2.20	2.99±2.25	3.99±3.06	1.01±0.25	1.06±0.31
	k	73	77	12	48	12/0
	models	281	299	45	214	44/11
	vpm	1110.0	37.2	1110.0	1110.0	94.5
	time	24.41	10.60	4.10	17.02	7.08
C : $\varepsilon = 0.26, N = 2273$	I	537±23	536±22	456±28	585±5	586±5
	error	1.82±0.57	1.76±0.62	5.39±2.03	1.10±0.25	1.08±0.28
	k	6826	7830	41	4117	35/0
	models	29281	33580	198	17631	155/64
	vpm	2273.0	20.7	2273.0	2273.0	25.6
	time	4100.09	1101.12	27.41	3746.64	20.02

TABLE 3: Results for essential matrix estimation. The table lists the same statistics as Tables 1 and 2.

[26] W. Zhang and J. Kosecka, "A new inlier identification procedure for robust estimation problems," in *Robotics: Science and Systems*, 2006.

[27] J. Choi and G. Medioni, "StaRSaC: Stable random sample consensus for parameter estimation," *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[28] R. Raguram and J.-M. Frahm, "Recon: Scale-adaptive robust estimation via residual consensus," in *International Conference on Computer Vision (ICCV)*, November 2011.

[29] R. Toldo and A. Fusiello, "Robust multiple structures estimation with j-linkage," in *European Conference on Computer Vision (ECCV)*, 2008, pp. 537–547.

[30] T.-J. Chin, H. Wang, and D. Suter, "Robust fitting of multiple structures: The statistical learning approach," in *International Conference on Computer Vision (ICCV)*, 2009.

[31] T.-J. Chin, J. Yu, and D. Suter, "Accelerated hypothesis generation for multi-structure robust fitting," in *European Conference on Computer Vision (ECCV)*, 2010.

[32] J. Neira and J. Tardós, "Data association in stochastic mapping using the joint compatibility test," *IEEE Transactions on Robotics and Automation*, vol. Vol. 17, no. No. 6, pp. 890 – 897, December 2001.

[33] M. Chli and A. J. Davison, "Active matching," in *European Conference on Computer Vision (ECCV)*, 2008, pp. 72–85.

[34] H. Li, "Consensus set maximization with guaranteed global optimality for robust geometry estimation." in *International Conference on Computer Vision (ICCV)*, 2009, pp. 1074–1080.

[35] B. Tordoff and D. W. Murray, "Guided sampling and consensus for motion estimation," in *European Conference on Computer Vision (ECCV)*, 2002, pp. 82–98.

[36] C. Schmid and R. Mohr, "Local grayvalue invariants for image retrieval," *IEEE Trans. PAMI*, vol. 19, no. 5, pp. 530–535, May 1997.

[37] T. Tuytelaars and L. V. Gool, "Wide baseline stereo matching based on local, affinely invariant regions," in *British Machine Vision Conference (BMVC)*, 2000, pp. 412–425.

[38] I.-K. Jung and S. Lacroix, "A robust interest points matching algorithm," in *Computer Vision, Eighth IEEE International Conference on*, vol. 2, 2001, pp. 538–543.

[39] J. Cech, J. Matas, and M. Perd'och, "Efficient sequential correspondence selection by cosegmentation," in *Computer Vision and Pattern*

Recognition (CVPR), June 2008, pp. 1–8.

[40] T. Sattler, B. Leibe, and L. Kobbelt, "SCRAMSAC: Improving RANSAC's efficiency with a spatial consistency filter," in *International Conference on Computer Vision (ICCV)*, 2009.

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

[42] Z. Zhang, R. Deriche, O. Faugeras, and Q. Luong, "A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry," *Artificial intelligence*, vol. 78, no. 1-2, pp. 87–119, October 1995.

[43] Z. Zhang and T. Kanade, "Determining the epipolar geometry and its uncertainty: A review," *International Journal of Computer Vision*, vol. 27, pp. 161–195, 1998.

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

[45] K. Ni, H. Jin, and F. Dellaert, "GroupSAC: Efficient consensus in the presence of groupings," in *International Conference on Computer Vision (ICCV)*, October 2009.

[46] O. Chum, T. Werner, and J. Matas, "Epipolar geometry estimation via RANSAC benefits from the oriented epipolar constraint," in *International Conference on Pattern Recognition (ICPR)*, 2004, pp. 112–115.

[47] J. Matas and O. Chum, "Randomized RANSAC with $T_{d,d}$ test," *British Machine Vision Conference (BMVC)*, pp. 448–457, 2002.

[48] O. Chum, J. Matas, and J. Kittler, "Locally optimized RANSAC," in *DAGM-Symposium*, 2003, pp. 236–243.

[49] D. Capel, "An effective bail-out test for RANSAC consensus scoring," in *British Machine Vision Conference (BMVC)*, 2005.

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

[51] A. Wald, *Sequential analysis*. New York: Dover, 1947.

[52] D. Nistér, "Preemptive RANSAC for live structure and motion estimation," in *International Conference on Computer Vision (ICCV)*, 2003.

[53] O. Chum, T. Werner, and J. Matas, "Two-view geometry estimation unaffected by a dominant plane," in *Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 772–779.

[54] J.-M. Frahm and M. Pollefeys, "RANSAC for (quasi-)degenerate data (QDEGSAC)," in *Computer Vision and Pattern Recognition (CVPR)*, 2006, pp. 453–460.

[55] D. Nistér, "An efficient solution to the five-point relative pose problem," *IEEE Trans. PAMI*, vol. 26, pp. 756–777, June 2004.

PLACE PHOTO HERE

Rahul Raguram received the MS degree in Electrical and Computer Engineering from the University of Arizona, Tucson, in 2007,. He is currently a PhD student at the University of North Carolina at Chapel Hill. His research interests include robust estimation, large scale structure from motion from video and internet photo collections, 2D/3D scene analysis and segmentation, applications of computer vision in security/privacy, image and video compression.

PLACE PHOTO HERE

Ondřej Chum received the MSc degree in computer science from Charles University, Prague, in 2001 and the PhD degree from the Czech Technical University in Prague, in 2005. From 2005 to 2006, he was a research Fellow at the Centre for Machine Perception, Czech Technical University. From 2006 to 2007 he was a post-doc at the Visual Geometry Group, University of Oxford, UK. Recently, he is back at the Centre for Machine Perception as a senior researcher. His research interests include object recognition, large-scale image and particular-object retrieval, invariant feature detection, and RANSAC-type optimization. He is a reviewer for the IEEE Transactions on Pattern Analysis and Machine Intelligence and other journals. He is regularly a program committee member at major computer vision conferences (for example, ICCV, CVPR, and ICPR). He has coorganized the "25 years of RANSAC" Workshop in conjunction with CVPR 2006. He was the recipient of the Best Paper Prize at the British Machine Vision Conference in 2002. He is a member of the IEEE.

PLACE PHOTO HERE

Marc Pollefeys is a full professor in the Dept.of Computer Science of ETH Zurich since 2007. He also remains associated with the Dept. of Computer Science of UNC at Chapel Hill where he started as an assistant professor in 2002. He holds a Ph.D. (1999) from the KULeuven. His main area of research is geometric computer vision and his aim is to develop flexible approaches to capture visual representations of real world objects, scenes and events. Prof. Pollefeys received a Marr prize, NSF CAREER award, Packard Fellowship and ERC grant. He is the author of more than 190 peer-reviewed publications. He is the General Chair for ECCV2014 and was a Program Co-Chair for CVPR2009. He is on the Editorial Board of IJCV, was an associate editor for IEEE PAMI and is a Fellow of the IEEE.

PLACE PHOTO HERE

Jiří Matas is a full professor at the Center for Machine Perception, Czech Technical University in Prague. He holds a PhD degree from the University of Surrey, UK (1995). He has published more than 200 papers in refereed journals and conferences. His publications have approximately 5000 citations in the ISI Thomson-Reuters Science Citation Index and about 12500 in Google scholar. His h-index is 23 (ISI) and 38 (Google scholar) respectively. He received the best paper prize at the British Machine Vision Conferences in 2002 and 2005 and at the Asian Conference on Computer Vision in 2007. J. Matas has served in various roles at major international conferences (e.g. ICCV, CVPR, ICPR, NIPS, ECCV), co-chairing ECCV 2004 and CVPR 2007. He is on the editorial board of IJCV and was the Associate Editor-in-Chief of IEEE T. PAMI. His research interests include object recognition, image retrieval, tracking, sequential pattern recognition, invariant feature detection, and Hough Transform and RANSAC-type optimization.

PLACE PHOTO HERE

Jan-Michael Frahm received his Ph.D in computer vision in 2005 from the Christian-Albrechts University of Kiel, Germany. His dissertation, "Camera Self-Calibration with Known Camera Orientation" received the prize for that year's best Ph.D. dissertation in CAU's College of Engineering. His Diploma in Computer Science is from the University of Lbeck. He is currently Assistant Professor at University of North Carolina at Chapel Hill. His research interests include structure from motion, real-time multi-view stereo, camera-sensor systems for 3D scene reconstruction with fusion of multiple orthogonal sensors, robust estimation methods, high performance feature tracking and the development of data-parallel algorithms for commodity graphics hardware.