# A Comparative Analysis of RANSAC Techniques Leading to Adaptive Real-Time Random Sample Consensus

Rahul Raguram[1], Jan-Michael Frahm[1], and Marc Pollefeys[1,2]

[1] Department of Computer Science, The University of North Carolina at Chapel Hill
{rraguram,jmf,marc}@cs.unc.edu
[2] Department of Computer Science, ETH Zürich
marc.pollefeys@inf.ethz.ch

**Abstract.** The Random Sample Consensus (RANSAC) algorithm is a popular tool for robust estimation problems in computer vision, primarily due to its ability to tolerate a tremendous fraction of outliers. There have been a number of recent efforts that aim to increase the efficiency of the standard RANSAC algorithm. Relatively fewer efforts, however, have been directed towards formulating RANSAC in a manner that is suitable for real-time implementation. The contributions of this work are two-fold: First, we provide a comparative analysis of the state-of-the-art RANSAC algorithms and categorize the various approaches. Second, we develop a powerful new framework for real-time robust estimation. The technique we develop is capable of efficiently adapting to the constraints presented by a fixed time budget, while at the same time providing accurate estimation over a wide range of inlier ratios. The method shows significant improvements in accuracy and speed over existing techniques.

## 1 Introduction

The RANSAC (Random Sample Consensus) algorithm [1] is a simple, yet powerful, technique that is commonly applied to the task of estimating the parameters of a model, using data that may be contaminated by outliers. RANSAC estimates a global relation that fits the data, while simultaneously classifying the data into inliers (points consistent with the relation) and outliers (points not consistent with the relation). Due to its ability to tolerate a large fraction of outliers, the algorithm is a popular choice for a variety of robust estimation problems.

RANSAC operates in a hypothesize-and-verify framework: a minimal subset of the input data points is randomly selected and model parameters are estimated from this subset. The model is then evaluated on the entire dataset and its *support* (the number of data points consistent with the model) is determined. This hypothesize-and-verify loop is repeated until the probability of finding a model with better support than the current best model falls below a predefined threshold (typically 1%-5%). RANSAC can often find the correct solution even for high levels of contamination; however, the number of samples required to do so increases exponentially, and the associated computational cost is substantial.

D. Forsyth, P. Torr, and A. Zisserman (Eds.): ECCV 2008, Part II, LNCS 5303, pp. 500–513, 2008.

There have been a number of recent efforts aimed at increasing the efficiency of the basic RANSAC algorithm. Some of these strategies [2,3,4] aim to optimize the process of model verification, while others [5,6,7] seek to modify the sampling process in order to preferentially generate more useful hypotheses. While these efforts have shown considerable promise, none of them are directly applicable in situations where real-time performance is essential. Relatively fewer efforts have been directed towards the goal of formulating RANSAC in a manner that is suitable for real-time implementations. In particular, Nister [8] describes the preemptive RANSAC framework, where a *fixed* number of hypotheses are evaluated in a *parallel, multi-stage* setting. In this case the goal is to find, within a fixed time budget, the best solution from a restricted set of hypotheses. While the preemptive RANSAC framework facilitates real-time implementation, there exist a few limitations in the scheme. One of the primary limitations of preemptive RANSAC is its inherent non-adaptiveness to the data. The selection of a fixed number of hypotheses implicitly implies that a good prior estimate of the inlier ratio is available; in practice, this is often not the case. For low contamination problems, preemptive RANSAC is often slower than standard RANSAC, since it evaluates many more hypotheses than necessary. On the other hand, when the inlier ratio is too low, preemptive RANSAC is unlikely to find a good solution, since it does not test enough hypotheses.

There has been recent work on dealing with issues arising from degenerate data configurations [9,10]. While we do not address degeneracy specifically in the paper, we note that both techniques are extensions of RANSAC and accordingly can be improved by the proposed scheme in a similar way.

The goal of the current paper is two-fold. First, we provide a comparative analysis of the important RANSAC algorithms that have been proposed over the past few years [2,3,4,5,7,8,11]. We classify the various approaches based on the aspect of RANSAC that they seek to optimize, and evaluate their performance on synthetic and real data. Following the evaluation, we propose a powerful new framework for real-time robust estimation. One of the main challenges in real-time robust estimation lies in being able to adapt to the strict limitations presented by the fixed time budget. The framework we develop is able to elegantly cope with variations in the contamination level of the data, while at the same time providing accurate estimation at real-time speeds.

## 2 Survey of RANSAC Techniques

### 2.1 RANSAC

As mentioned earlier, RANSAC operates in a hypothesize-and-verify framework. Given a set $\mathcal{U}$ containing $N$ tentative correspondences, RANSAC randomly samples subsets of size $m$ from the data, where $m$ is the minimal number of samples required to compute a solution, which is equivalent to the complexity of the geometric model. Once a model has been hypothesized from this minimal subset, it is verified against all data points in $\mathcal{U}$ and its support is determined.

This process is repeated until a termination criterion is met. The standard termination criterion for RANSAC is based on the minimum number of samples required to ensure, with some level of confidence $\eta_0$, that at least one of the selected minimal subsets is outlier-free. Given the true inlier ratio $\varepsilon$, the probability of selecting an uncontaminated sample of $m$ inliers is $\varepsilon^m$. The probability of selecting $k$ samples, each of which is contaminated with at least one outlier, is given by $(1-\varepsilon^m)^k$. Thus, the minimum number of samples that must be drawn in order to ensure that this probability falls below a $1-\eta_0$ threshold is

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

Since the true inlier ratio $\varepsilon$ is *a priori* unknown, a lower bound on this ratio can be found by using the sample which currently has the largest support. This estimate is then updated as the algorithm progresses. A number of recently proposed methods seek to optimize different aspects of the basic RANSAC algorithm. We discuss three strategies to optimize the model verification stage of RANSAC in Section 2.2. We also review in Section 2.3, methods that have been proposed to incorporate non-uniform sampling into the hypothesis generation stage. Finally, in Section 2.4, we discuss a very different RANSAC framework that operates in a real-time setting.

### 2.2 Optimizing Model Verification

While the number of samples drawn by RANSAC depends on the inlier ratio and the model complexity, an additional computational factor is the total number of correspondences $N$, since a hypothesis is verified against all data points. However, since most of the generated models are likely to be contaminated, these will be consistent only with a small fraction of the data. Thus, it is often possible to discard bad hypotheses early on in the verification process. Some of the early efforts to speed up RANSAC made use of precisely this observation.

#### 2.2.1 The $T_{d,d}$ Test

Matas and Chum [2] design a pre-evaluation stage which attempts to quickly filter out bad hypotheses. 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. A setting of $d=1$ is recommended as the optimal value. Due to the randomized hypothesis evaluation procedure, 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, in general, the overall running time is reduced due to the preverification procedure.

#### 2.2.2 Bail-Out Test

The idea of early termination of bad hypotheses was further extended by Capel in [3]. Given a randomly selected subset of $n$ points, the number of inliers $\kappa_n$ in this subset follows a hyper-geometric distribution: $\kappa_n \sim \text{HypG}(\kappa, n, \overline{\kappa}, N)$, where $\overline{\kappa}$ is the total number of inliers for the current hypothesis. Given the current

best hypothesis with inlier count $\overline{\kappa}_{best}$, consider a new hypothesis that has been partially evaluated against a subset of $n$ data points. The goal is to determine the probability that having observed $\kappa_n$ inliers so far, the total number of inliers for the new hypothesis, $\overline{\kappa}$, is greater than $\overline{\kappa}_{best}$. Since this is computationally expensive to calculate, a lower bound $\kappa^n_{min}$ is derived on the number of inliers observed after evaluating $n$ data points. If, at this point, the number of inliers is below the threshold, bail-out occurs and the hypothesis is discarded. For a given confidence bound $P_{conf}$, also note that the minimum bound $\kappa^n_{min}$ may be approximated as a binomial distribution (for small values of $n$) or as a normal distribution (for large values of $n$).

### 2.2.3 WaldSAC

Most recently, Chum and Matas described an optimal randomized model verification strategy [4,12] based on Wald's theory of sequential decision making. The evaluation step is cast as an optimization problem which aims to decide whether a model is good ($H_g$) or bad ($H_b$), while simultaneously minimizing the number of verifications performed per model. Wald's Sequential Probability Ratio Test (SPRT) is based on the likelihood ratio

$$\lambda_j = \prod_{r=1}^{j} \frac{p(x_r|H_b)}{p(x_r|H_g)} \tag{2}$$

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 $\varepsilon$. 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 $\delta$. If, after evaluating $j$ data points, the likelihood ratio becomes greater than some threshold $A$, the model is rejected. The decision threshold $A$ is the only parameter of the SPRT test, and it can be set to achieve optimal running time (for more details we refer to [5]). When the level of contamination is known *a priori*, the WaldSAC strategy is provably optimal. In practice, however, inlier ratios have to be estimated during the evaluation process and the threshold $A$ is adjusted to current estimates. An initial estimate of the parameter $\delta$ is obtained through geometric considerations, and this estimate is revised during the sampling process. The performance of the SPRT test is not significantly affected by the imperfect estimation of these parameters. A detailed comparative evaluation of the three methods described above is provided in Section 4. While we defer a discussion of the experimental results until later, we briefly note that in general, the performance of the bail-out and SPRT tests are comparable, both producing speed-ups ranging between 2-7 times compared to standard RANSAC.

## 2.3 Improving Hypothesis Generation

The RANSAC algorithm generates hypotheses by uniformly sampling the input data set. This implicitly implies that there is no *a priori* information available

about the accuracy of the data points. In practice, this may be an overly pessimistic approach, since *a priori* information is often available, and can be used to generate better hypotheses.

### 2.3.1 PROSAC

Typically, correspondences between two or more images are obtained by the use of a local matching algorithm. A similarity function is evaluated over a number of points, and subsequently thresholded to obtain a set of tentative correspondences. Based on the assumption that points with high similarity are more likely to be inliers than points with low similarity, it may be possible to generate better hypotheses by sampling from a reduced set of points with high similarity scores. This fact is exploited in [7], where the correspondences are ordered based on their similarity scores, and progressively larger subsets of tentative correspondences are used to generate hypotheses. The Progressive Sample Consensus (PROSAC) algorithm is designed to draw the same samples as RANSAC, only in a different 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 $\mathcal{U}_n$ denote a subset of $\mathcal{U}_N$ containing $n$ points with the highest quality. The sequence $\{\mathcal{M}_i\}_{i=1}^{T_N}$ contains, on average, $T_n$ samples containing points only from $\mathcal{U}_n$, where

$$T_n = T_N \frac{\binom{n}{m}}{\binom{N}{m}} \tag{3}$$

A recurrence relation for $T_{n+1}$ can be obtained as: $T_{n+1} = \frac{n+1}{n+1-m} T_n$. Since this may not necessarily be an integer, $T'_{n+1}$ is defined as $T'_{n+1} = T'_n + \lceil T_{n+1} - T_n \rceil$. The $t$-th sample in PROSAC is denoted by $\mathcal{M}_t$ and is generated as follows:

$$\mathcal{M}_t = \{\mathbf{u}_{g(t)}\} \bigcup \mathcal{M}'_t \tag{4}$$

where $g(t)$ is a growth function defined as $g(t) = \min\{n : T'_n \geq t\}$ and $\mathcal{M}'_t$ is a set containing $m-1$ points drawn at random from the set $\mathcal{U}_{g(t)-1}$. In practice, the PROSAC approach often achieves significant computational savings, since good hypotheses are generated early on in the sampling process. Two important points must, however, be noted. First, though PROSAC often succeeds in dramatically reducing the number of hypotheses required, this is data-dependent, and also hinges on the availability of a reasonable similarity function to rank correspondences. Secondly, we observe that in many cases, correspondences with high similarity scores often lie on the same spatial structure and are potentially in a degenerate configuration. Thus, when utilizing the PROSAC approach, it is advisable to ensure robustness to degenerate configurations [9,10].

### 2.3.2 Guided Sampling for MLESAC

A similar approach to PROSAC was proposed earlier by Tordoff and Murray in [6], wheer the Maximum Likelihood Estimation Sample Consensus (MLESAC) algorithm [11] was combined with non-uniform sampling of correspondences. The

**Algorithm 1.** Preemptive RANSAC

```
Generate all hypotheses (indexed by 1, ..., f(1))
for i = 1 to N do
   Score the hypotheses 1, ..., f(i) using data point i
   Reorder the set to retain the best f(i + 1) hypotheses
   if f(i + 1) = 1 then
      Break with the remaining hypothesis as the top one
   end if
end for
```

MLESAC algorithm is a generalization of RANSAC, which adopts the same sampling strategy but attempts to maximize the likelihood of the solution, as opposed to the number of inliers. While MLESAC assumes a uniform prior for the validity of a match, the *guided-sampling* approach of [6] uses the quality function of the feature matching algorithm to derive probabilities of match validity. These are then incorporated as priors in MLESAC. The technique was experimentally shown to reduce the number of iterations required by MLESAC by an order of magnitude.

#### 2.3.3 Lo-RANSAC

One of the assumptions inherent in the standard termination criterion of RANSAC (equation (1)) is that a model computed from an uncontaminated sample is consistent with all inliers. In practice, this is often not the case, particularly when the data points are noisy. Chum et al. [5] define a locally optimized RANSAC variant to deal with this issue. By observing that a good model tends to find a significant fraction of the inliers, an *inner RANSAC* strategy is devised where a constant number of hypotheses ($\sim 20$) are generated using only the set of inliers to the current best model. Since inner RANSAC operates on a set of inliers, it is not essential that hypotheses are generated from minimal subsets. In addition to providing a more robust fit, the inner RANSAC technique has the effect of improving the consensus score more rapidly than standard RANSAC, which causes the termination criterion (1) to be met earlier.

### 2.4 Preemptive RANSAC

As discussed earlier, though the recent variants of RANSAC show an appreciable decrease in runtime, they are still not directly applicable in real-time implementations. This is primarily due to the fact that in order to achieve an $\eta_0$ confidence in the solution, all these methods need to draw a significantly large number of samples, particularly for low inlier ratios. In the case of real-time systems the goal is to achieve the threshold when possible; however, when this is not the case, it is desirable to obtain the best solution given the time constraints. Preemptive RANSAC attempts to do precisely this: a fixed number of hypotheses are generated beforehand, and then compared against each other by scoring them in parallel. The preemption schemes described in Section 2.2 are

*depth-first* in nature, meaning that a particular hypothesis is completely evaluated before moving on to the next hypothesis. Preemptive RANSAC uses a *breadth-first* approach, where all the hypotheses generated are evaluated on a subset of the data points. Subsequently, the hypotheses are reordered based on the results of this scoring procedure, and only a fraction of the hypotheses ar e evaluated on the next subset of data. A non-increasing preemption function $f(i)$, $i = 1, ..., N$ defines the number of hypotheses retained before evaluating the $i^{th}$ data point. The preemption function used in [8] is

$$f(i) = \lfloor M 2^{(-\lfloor \frac{i}{B} \rfloor)} \rfloor \tag{5}$$

where $M$ is the number of hypotheses in the initial set and $B$ is the number of data points that a hypothesis is evaluated against, before the preemption and reordering step takes place. This process continues until only one hypothesis remains, or all data points have been used. Recommended values for these parameters are $M = 500$ hypotheses and $B = 100$ data points per subset.

## 3 Adaptive Real-Time Random Sample Consensus

In this section, we describe a novel framework for Adaptive Real-Time Random Sample Consensus (ARRSAC), which is capable of providing accurate real-time estimation over a wide range of inlier ratios. From the discussion on preemptive RANSAC (Section 2.4), it can be seen that the number of hypotheses $M$ in the initial candidate set is fixed *a priori*. In a real-time setting, this ensures that the runtime of the algorithm is bounded. However, fixing this number beforehand effectively places a restriction on the inlier ratio. When the true inlier ratio is high, preemptive RANSAC evaluates far too many samples. On the other hand, when the true inlier ratio is low, the likelihood that preemptive RANSAC finds a correct solution decreases drastically. The breadth-first scan advocated by preemptive RANSAC is indeed a natural formulation for real-time applications, since the primary constraint is the fixed time-budget. On the other hand, adapting the number of hypotheses to the contamination level of the data requires an estimate of the inlier ratio, which necessitates the adoption of a depth-first scan. The ARRSAC framework retains the benefits of both approaches (i.e, bounded run-time as well as adaptivity) by operating in a *partially depth-first* manner. Hypotheses for the initial candidate set are generated one-by-one, and are evaluated on the first data block. Bad hypotheses are discarded based on a depth-first preemptive strategy. Hypotheses that pass this verification procedure provide an estimate of the inlier ratio, which is used to determine the required number of hypotheses, keeping the upper limit fixed at $M$. Note that since the evaluation is performed on only a subset of the data (thus, partially depth-first), the estimated inlier ratio may be either above or below the true inlier ratio. In view of this fact, the ARRSAC algorithm allows the generation of additional hypotheses at a later stage in the evaluation process. While this discussion describes a method for adaptively determining the number of hypotheses, note that there is still an upper limit on the size of the hypothesis set. In such a scenario, it is then

**Algorithm 2.** The ARRSAC algorithm

**Initialization**
Set values for $M$ (max. number of candidate hypotheses) and $B$ (block size)
Set parameters for the SPRT test, calculate the initial value of $A$ (see [4])
**1. Generate initial hypothesis set** (Algorithm 3)
$k$ =total number of hypotheses generated in the initial stage
**2. Preemptive evaluation**
**for** $i = B + 1$ to $N$ **do**
  Set $p$ =number of hypotheses remaining, $n$ =min$(f(i), p/2)$
  Reorder and select hypotheses $h(1), ..., h(n)$
  **if** $n = 1$ **then**
    Break with the remaining hypothesis as the top one
  **end if**
  Score the hypotheses using data point $i$
  **if** $(i \bmod B) = 0$ **then**
    Calculate best inlier ratio $\hat{\varepsilon}$ and number of hypotheses $M'$ (equation (1))
    $M'$ =max$(M, M')$
    **if** $M' > k$ **then**
      Generate and evaluate $M' - k$ new hypotheses on $i$ data points
      $k = M'$
    **end if**
  **end if**
**end for**

important to choose a strong set of candidate hypotheses, since this facilitates good performance at low inlier ratios. ARRSAC accomplishes this by adopting a non-uniform sampling approach, generating hypotheses from the highest quality matches. Note, however, that even in the absence of information to guide non-uniform sampling (or when this does not help), ARRSAC still has a bounded runtime. This is in contrast with strategies such as PROSAC, which in the worse case operates like RANSAC (and hence has a potentially unbounded runtime). The ability to sample non-uniformly from the data is crucial only when the inlier ratios are very low. As an additional strategy, once a hypothesis passes the verification test, it is also possible to generate new hypotheses by sampling from data points that are consistent with this good hypothesis. This idea is similar to the *inner RANSAC* technique in [5]. Since the sampling now is selecting points from inlier data, it is not essential that hypotheses are generated from minimal subsets. An important advantage of the competitive evaluation framework is that ARRSAC does not spend excessive time in local optimization, since the hypotheses generated in the inner RANSAC are forced to compete among themselves. This prevents the algorithm from spending excessive time in the refinement, particularly when the inlier ratio is high.

Finally, while the partially depth-first evaluation strategy described above serves to adaptively determine the number of hypotheses, it also provides an additional computational advantage. In the original preemptive RANSAC algorithm, hypotheses are evaluated on all $B$ data points in a block before the reordering and preemption step takes place. As we have noted earlier, this

**Algorithm 3.** ARRSAC: Generating the initial hypothesis set

$k = 1$, $M' = M$, $count_{in} = 0$, Inner RANSAC flag: $flag_{in} = 0$
Set max number of inner RANSAC iterations: $M_{in}$
**repeat**
  **if** $flag_{in} = 0$ **then**
    Generate hypothesis $h(k)$ by selecting the $k$-th PROSAC sample
  **else**
    Generate hypothesis $h(k)$ by sampling (perhaps non-minimal) subset from $\mathcal{U}_{in}$
    $count_{in} = count_{in} + 1$
    **if** $count_{in} = M_{in}$, reset inner RANSAC parameters ($count_{in}$, $flag_{in}$)
  **end if**
  Evaluate hypothesis $h(k)$ using Wald's SPRT
  **if** hypothesis $h(k)$ is rejected **then**
    Reestimate parameters of the SPRT (if required)
  **else if** hypothesis $h(k)$ is accepted and has the largest support so far **then**
    $flag_{in} = 1$, $count_{in} = 0$, set $\mathcal{U}_{in}$ = support of hypothesis $h(k)$
    Reestimate parameters of the SPRT
    Estimate inlier ratio $\hat{\varepsilon}$ and $M'$ (equation (1)). Cap $M'$ at a max of $M$
  **end if**
  $k = k + 1$
**until** $(k > M')$
Return $k$, set containing accepted hypotheses

constitutes an unnecessary overhead since most of the generated hypotheses are likely to be contaminated. Thus, in ARRSAC, bad hypotheses are discarded after partial evaluation and only the hypotheses that survive are selected for propagation to subsequent stages of evaluation. The preemption function (7) thus effectively defines an upper bound on the number of hypotheses required for the next stage of evaluation. The number of good hypotheses that need to be reordered and propagated in ARRSAC is typically much less than this bound.

To summarize the main contributions of the ARRSAC algorithm: the adoption of a partially depth-first strategy for hypothesis generation provides a great degree of control over the initial set. Adaptively determining the number of hypotheses ensures computational efficiency for higher inlier ratios. Furthermore, this strategy also allows more control over the quality of hypotheses that are included in the initial set, thereby increasing the likelihood of obtaining a good solution even for low inlier ratios. Finally, the use of optimized verification techniques is geared towards improving the overall efficiency of the algorithm. We note that unlike preemptive RANSAC, the proposed method makes no assumptions about the underlying data. The ability to efficiently adapt to the data and provide accurate real-time robust estimation makes ARRSAC easily deployable in real-time vision systems.

### 3.1 Algorithm Description

**Generating the Initial Hypothesis Set.** The ARRSAC algorithm adopts a PROSAC-based non-uniform sampling strategy to generate hypotheses. For

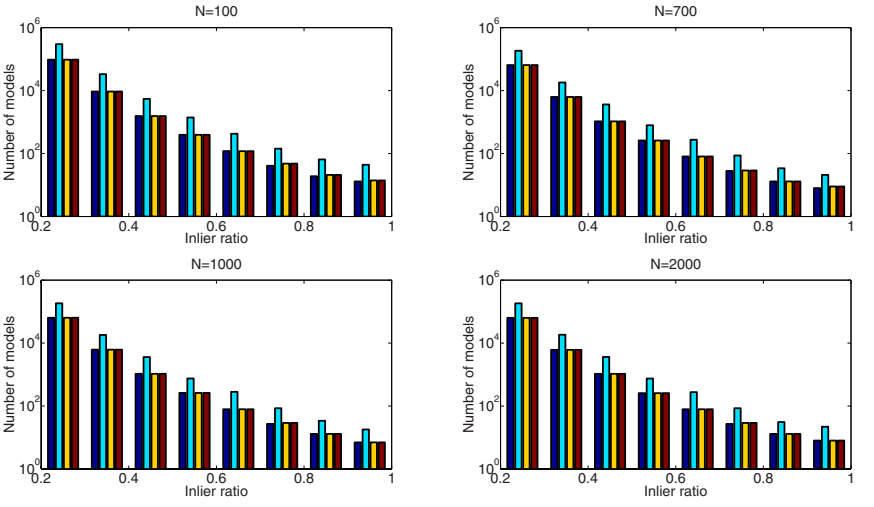



**Fig. 1.** Results of synthetic data evaluation. Number of models evaluated vs. inlier ratio for four different values of $N$. From left-right in each group: RANSAC, $T_{d,d}$, bail-out, WaldSAC. Note that the y-axis uses a log scale.

clarity, we also restrict our attention to a preemptive approach based on SPRT, though in principle any depth-first preemption scheme may be used. For the SPRT, the threshold parameter $A$ depends on the inlier ratio $\varepsilon$ and the Bernoulli parameter $\delta$. These are initialized conservatively ($\varepsilon = 0.1$, $\delta = 0.05$) and updated as the processing continues. During the SPRT, a hypothesis is either rejected or accepted. In the first case, we continue to the next hypothesis, updating $\delta$ if required. In the second case, when a hypothesis is accepted and has the best support so far, a fixed number of new hypotheses, $M_{in}$, are generated by sampling non-minimal subsets from the inliers to this hypothesis. Furthermore, an accepted hypothesis provides an estimate $\hat{\varepsilon}$ of the inlier ratio; this is used to determine the new number of hypotheses required, $M'$. Note that we cap the maximum number of hypotheses in the initial set at $M$, so $M' \leq M$. It is also important to note that the estimate $\hat{\varepsilon}$ may be either greater than or less than the actual inlier ratio. However, the flexibility of the ARRSAC algorithm allows us to always generate additional hypotheses at a later stage, if required to do so.

**Preemptive evaluation.** The evaluation procedure in ARRSAC is based on the partially depth-first approach outlined earlier. While preemptive RANSAC uses the fixed preemption function (7), ARRSAC is more flexible in this regard. Thus, at any given point, there are typically far less than $f(i)$ hypotheses in the set, due to the fact that contaminated hypotheses are discarded. In practice, this turns out to be a very significant computational saving (see Section 4). After every evaluation of $B$ data points, the inlier ratio $\hat{\varepsilon}$ is re-estimated, and the value of $M'$ is updated. If, at any point, the inlier ratio is found to be an overestimate, additional hypotheses are evaluated as shown in Algorithm 2.

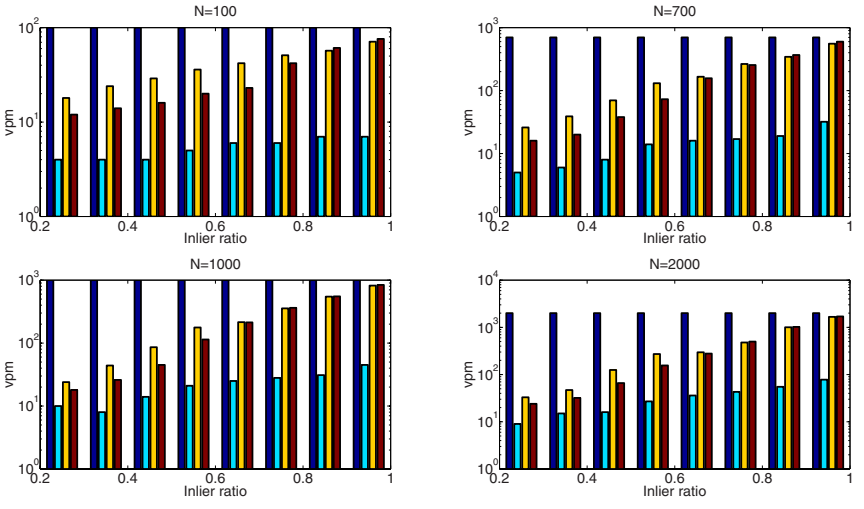



**Fig. 2.** Results of synthetic data evaluation. Number of verifications per model (vpm) vs. inlier ratio for four different values of $N$. From left-right in each group: RANSAC, $T_{d,d}$, bail-out, WaldSAC. Note that the y-axis uses a log scale.

## 4 Experimental Evaluation

In this section, we provide a detailed comparative evaluation of the approaches described in this paper. First, we compare the performance of the various depth-first preemptive approaches discussed in Section 2.2. For this experiment, we estimate the epipolar geometry for synthetic data over a wide range of parameter variations. Gaussian noise ($\mu = 0, \sigma = 0.2$) was added to the pixel locations. The results of the synthetic evaluation are shown in Figures 1 and 2. Figure 1 shows the variation in the number of models evaluated as a function of the inlier ratio $\varepsilon$, for four different values of $N$ (total number of correspondences). Figure 2 shows the number of verifications per model, again as a function of $\varepsilon$ and $N$. The results are averaged over 200 runs for each tested case. Figure 2 shows that the $T_{d,d}$ test succeeds in greatly reducing the number of verifications per model. On the other hand, it requires the generation of a much larger number of hypotheses. The performance of the bail-out and SPRT tests are comparable over the range of parameter variations, with the SPRT producing slightly greater speed-ups on average. However, for high inlier ratios, when the total number of models generated is small, the bail-out test often performs marginally better than SPRT, since the parameters in the SPRT are initialized conservatively.

We next evaluate the performance of preemptive RANSAC on synthetic data, over a range of parameter variations. The goal in this case is not to examine the number of verifications or the execution time (which are constant). Rather, we seek to examine the ability of preemptive RANSAC to cope with varying inlier ratios. Figure 3(a) shows the fraction of inliers found, as a function of $\varepsilon$ and $N$. The main trend to be observed from the graph is that for lower inlier ratios

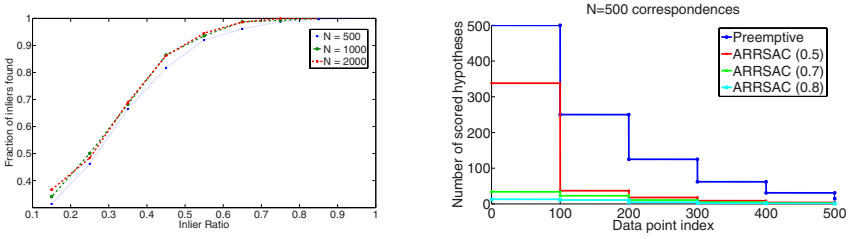
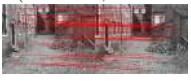
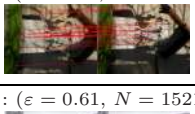



**Fig. 3.** (a) Variation in fraction of inliers found by preemptive RANSAC as a function of inlier ratio. (b) Number of hypotheses scored at each stage of preemption in preemptive RANSAC and ARRSAC, for four values of $\varepsilon$.

($< 0.5$), preemptive RANSAC does not evaluate a sufficient number of hypotheses, and thus misses a significant number of inliers. This presents a drawback for the robust operation of a real-time system. Figure 3(b) compares the number of hypotheses retained at each stage of preemption for preemptive RANSAC and ARRSAC, for synthetic data at four values of $\varepsilon$. Note that for this experiment, it was assumed that a PROSAC-based ordering of the hypotheses was unavailable. It can be seen that even without PROSAC, the partially depth-first evaluation strategy used in ARRSAC causes an appreciable decrease in the number of hypotheses evaluated at each stage. Thus, the effective preemption function in ARRSAC always remains below that of equation (7), providing considerable computational savings. This demonstrates one of the advantages associated with being able to adapt to the contamination level of the data.

Finally, we provide a comprehensive comparative evaluation of the various RANSAC methods on real image data, and demonstrate the efficiency of the ARRSAC approach, in terms of accuracy and computational efficiency. A sample of the test images chosen for this experiment are shown in Table 1. These comprise of image pairs from well-known sequences (A-C), as well as images taken from various other sources (D-F, G, H-I, J). The images were selected to cover a wide range of inlier ratios and number of correspondences. Corresponding points in the image pairs were detected using Harris corner matching. The results for the evaluation are shown in Table 1 which lists, for each method and test sequence, the number of inliers found ($I$), the number of models evaluated ($k$), the number of verifications per model (vpm) and the relative speed-up obtained with respect to standard RANSAC (spd-up). A few important trends are noticeable from the table. Firstly, for images where the total number of models evaluated is low (A, B, C), it can be observed that preemptive RANSAC evaluates more hypotheses than standard RANSAC and is thus slower. On the other hand, ARRSAC is able to efficiently estimate the inlier ratio and accordingly limit the number of hypotheses, producing a considerable speed-up. Secondly, for images with low inlier ratios (D, E, F), preemptive RANSAC is often unable to find the correct solution since it evaluates too few hypotheses. In contrast, the adaptive hypothesis generation in ARRSAC enables accurate estimation of

**Table 1.** Evaluation results for ten selected real image pairs. The images show variation over a range of inlier ratios and number of correspondences. It can be observed from the above results that the ARRSAC approach produces significant computational speed-ups, while simultaneously providing accurate robust estimation in real-time. In practice, the ARRSAC technique stays well within the time budget, with estimation speeds ranging between 55-350 Hz. It can be seen from the table that the number of hypotheses evaluated by ARRSAC is always less than preemptive RANSAC. In addition, the correct epipolar geometry is always recovered.

| | | RANSAC | $T_{d,d}$ | Bail-out | Wald | PROSAC | Pre-emptive | ARRSAC |
|---|---|---|---|---|---|---|---|---|
| **A** : $\varepsilon = 0.83$, $N = 1322$  | $I$ | 884 | 884 | 885 | 889 | 885 | 933 | 1099 |
| | $k$ | 112 | 156 | 118 | 148 | 5 | 500 | 6 |
| | vpm | 1322 | 531 | 361 | 594 | 1322 | 396 | 434 |
| | spd-up | 1.0 | 1.9 | 2.6 | 2.2 | 12.9 | 0.4 | 37.3 |
| **B** : ($\varepsilon = 0.7$, $N = 795$)  | $I$ | 514 | 512 | 514 | 517 | 514 | 527 | 559 |
| | $k$ | 242 | 329 | 242 | 283 | 43 | 500 | 47 |
| | vpm | 795 | 149 | 65 | 132 | 795 | 374 | 218 |
| | spd-up | 1.0 | 2.4 | 4.1 | 3.9 | 7.7 | 0.3 | 11.1 |
| **C** : ($\varepsilon = 0.65$, $N = 2162$)  | $I$ | 1344 | 1342 | 1344 | 1345 | 1344 | 1372 | 1413 |
| | $k$ | 208 | 947 | 208 | 211 | 17 | 500 | 17 |
| | vpm | 2162 | 59 | 67 | 52 | 2162 | 400 | 430 |
| | spd-up | 1.0 | 1.9 | 5.5 | 5.8 | 15.2 | 0.9 | 29.1 |
| **D** : ($\varepsilon = 0.33$, $N = 1500$)  | $I$ | 483 | 483 | 484 | 484 | 484 | 350 | 491 |
| | $k$ | 15506 | 49090 | 15509 | 15514 | 201 | 500 | 241 |
| | vpm | 1500 | 22 | 37 | 28 | 1500 | 398 | 59 |
| | spd-up | 1.0 | 1.9 | 5.7 | 6.2 | 104.2 | 90.6 | 305.7 |
| **E** : ($\varepsilon = 0.44$, $N = 420$)  | $I$ | 174 | 174 | 173 | 177 | 177 | 134 | 183 |
| | $k$ | 2714 | 4889 | 2719 | 2716 | 439 | 500 | 499 |
| | vpm | 420 | 10 | 24 | 13 | 420 | 305 | 70 |
| | spd-up | 1.0 | 2.0 | 4.4 | 5.7 | 5.3 | 11.5 | 20.2 |
| **F** : ($\varepsilon = 0.43$, $N = 1325$)  | $I$ | 557 | 557 | 557 | 557 | 557 | 488 | 582 |
| | $k$ | 4353 | 11753 | 4366 | 4359 | 288 | 500 | 329 |
| | vpm | 1325 | 5 | 31 | 24 | 1325 | 396 | 182 |
| | spd-up | 1.0 | 2.1 | 4.7 | 5.3 | 6.9 | 12.3 | 15.1 |
| **G** : ($\varepsilon = 0.56$, $N = 2875$)  | $I$ | 1537 | 1524 | 1539 | 1537 | 1537 | 1539 | 1616 |
| | $k$ | 432 | 628 | 437 | 432 | 30 | 500 | 33 |
| | vpm | 2875 | 20 | 24 | 31 | 2875 | 400 | 456 |
| | spd-up | 1.0 | 2.2 | 3.1 | 4.2 | 16.6 | 1.3 | 47.2 |
| **H** : ($\varepsilon = 0.67$, $N = 1986$)  | $I$ | 1249 | 1247 | 1251 | 1252 | 1250 | 1237 | 1330 |
| | $k$ | 335 | 1153 | 351 | 347 | 17 | 500 | 18 |
| | vpm | 1986 | 29 | 41 | 32 | 1986 | 400 | 467 |
| | spd-up | 1.0 | 2.4 | 4.2 | 4.9 | 22.2 | 1.1 | 60.9 |
| **I** : ($\varepsilon = 0.76$, $N = 1508$)  | $I$ | 945 | 934 | 948 | 951 | 949 | 960 | 1149 |
| | $k$ | 498 | 1644 | 505 | 502 | 58 | 500 | 65 |
| | vpm | 1508 | 43 | 31 | 33 | 1508 | 398 | 277 |
| | spd-up | 1.0 | 1.9 | 2.7 | 2.5 | 12.3 | 1.8 | 45.1 |
| **J** : ($\varepsilon = 0.61$, $N = 1521$)  | $I$ | 872 | 872 | 873 | 873 | 872 | 870 | 927 |
| | $k$ | 362 | 1167 | 381 | 372 | 40 | 500 | 44 |
| | vpm | 1521 | 39 | 28 | 25 | 1521 | 398 | 338 |
| | spd-up | 1.0 | 2.2 | 6.1 | 6.3 | 9.8 | 1.9 | 26.1 |

the epipolar geometry even for very low inlier ratios. Thirdly, even in the case where a PROSAC-based sampling strategy does not provide any significant advantage (E), ARRSAC is still able to achieve a performance speed-up, due to the efficient preemption strategy employed (as illustrated in Fig 3(b)). As the results in Table 1 indicate, the ARRSAC algorithm is able to elegantly adapt to the data and ensure accurate robust estimation over a wide range of inlier ratios, while at the same time providing considerable computational savings.

## 5 Conclusion

In this paper, we presented a discussion and comparative analysis of a number of important RANSAC techniques. In addition, a powerful new formulation for adaptive real-time random sample consensus (ARRSAC) was proposed. In contrast to existing techniques for real-time robust estimation, the ARRSAC approach makes no limiting assumptions about the underlying data, and is capable of efficiently adapting to the contamination level of the data. The framework we develop is suitable for use in real-time applications with a limited time budget, while at the same time providing accurate robust estimation over a wide range of inlier ratios. Experimental results on synthetic and real data demonstrate very significant improvements in accuracy and speed over existing methods.

## References

1. Fisher, M., Bolles, R.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Comm. of the ACM 24(6), 381–395 (1981)
2. Matas, J., Chum, O.: Randomized RANSAC with $T_{d,d}$ test. Image and Vision Computing 22(10), 837–842 (2004)
3. Capel, D.: An effective bail-out test for RANSAC consensus scoring. In: Proc. BMVC, pp. 629–638 (2005)
4. Matas, J., Chum, O.: Randomized ransac with sequential probability ratio test. In: Proc. ICCV, pp. 1727–1732 (2005)
5. Chum, O., Matas, J., Kittler, J.: Locally optimized RANSAC. In: Michaelis, B., Krell, G. (eds.) DAGM 2003. LNCS, vol. 2781, pp. 236–243. Springer, Heidelberg (2003)
6. Tordoff, B., Murray, D.W.: Guided sampling and consensus for motion estimation. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2350, pp. 82–98. Springer, Heidelberg (2002)
7. Chum, O., Matas, J.: Matching with PROSAC - progressive sample consensus. In: Proc. CVPR, vol. 1, pp. 220–226 (2005)
8. Nister, D.: Preemptive RANSAC for live structure and motion estimation. In: Proc. ICCV, vol. 1, pp. 199–206 (2003)
9. Chum, O., Werner, T., Matas, J.: Two-view geometry estimation unaffected by a dominant plane. In: Proc. CVPR, pp. I: 772–779 (2005)
10. Frahm, J.M., Pollefeys, M.: Ransac for (quasi-)degenerate data (QDEGSAC). In: Proc. CVPR, pp. 453–460 (2006)
11. Torr, P., Zisserman, A.: MLESAC: A new robust estimator with application to estimating image geometry. CVIU, 138–156 (2000)
12. Chum, O., Matas, J.: Optimal randomized RANSAC. Pattern Analysis and Machine Intelligence (to appear)