RATSAC: An Adaptive Method for Accelerated Robust Estimation and its Application to Video Synchronisation

Daniel Pooley
School of Computer Science,
University of Adelaide
dwpooley@cs.adelaide.edu.au

Michael Brooks
School of Computer Science,
University of Adelaide
mjb@cs.adelaide.edu.au

Anton van den Hengel
School of Computer Science,
University of Adelaide
anton@cs.adelaide.edu.au

Abstract

A new method for robust estimation is introduced. The presented algorithm seeks to unify adjustable per-iteration speedup methods with an adaptive assumption regarding the number of inliers. This is achieved by assuming a prior distribution on the true number of inliers, and using Bayesian inference to adjust the speedup whenever a best-so-far model estimate is found. Convincing results are obtained for both synthetic and real cases of the robust synchronisation of video pairs generated by independently moving cameras.

1 Introduction

Many vision problems require the estimation of a model from data contaminated with a significant proportion of outliers. Two classical methods for dealing with such problems are the Least Median of Squares (LMedS) [8] and RANSAC [5]. Both of these methods repeatedly draw upon small random subsets of the data. For each subset, the model is estimated and assessed according to how well it fits the entire data set. LMedS ranks an estimate by the median of the error measurements describing how well the data fit the model, whereas RANSAC uses the number of data classified as inliers, according to some noise distribution and threshold.

The RANSAC approach has been enhanced in a variety of ways. Some methods, such as MSAC [11], MLESAC [12] and MAPSAC [10] use more sophisticated criteria to assess each model, improving the probability that the best is identified and returned upon termination. Other enhancements have focused on efficiency, seeking to decrease the execution time. These approaches range from the problem-specific [4] to the general [3], and are particularly useful since larger vision problems such as structure from motion require many robust estimations.

Of particular relevence to this paper is Randomized RANSAC [2], in which a model estimate is discarded if it fails the $T_{d,d}$ test. This test chooses d data at random, and determines if all are classified as inliers by the model estimate. The average time for each iteration is reduced, yet more iterations are required to compensate for the higher probability that a 'correct' estimate is wrongly discarded. Given an assumed number of inliers, this tradeoff can be analysed in advance to find the choice for d that minimises the total execution time. A lenient variation of this test is used in [6], with a robust estimation strategy used to achieve structure from motion in real time. All model estimates are generated *first*, and the worst are progressively discarded as more data are considered.

2 Accelerated Robust Estimation

The following notation is used. The unknown model parameters to be estimated are denoted by the vector $\boldsymbol{\theta}$. Both inliers and outliers are contained within the data set $M = \{x_i | i \in [1 \dots n_x]\}$. A robust estimation process seeks to both compute $\boldsymbol{\theta}$, and classify each x_i as either an inlier or outlier. On each iteration, a data subset of size c is randomly chosen to compute an estimate of $\boldsymbol{\theta}$.

The termination condition for existing robust methods is often derived from the probability of selecting a random subset without any outliers. For an assumed number of inliers μ, this probability is denoted here as $\mathcal{G}(\mu)$, and is approximated in [8] as $(\mu n_x^{-1})^c$. It is noted in [2] that this is incorrect since random subsets are chosen without a repetition of elements, and the correct expression is

$$\mathcal{G}(\mu) = \frac{(n_x - c)!\mu!}{(\mu - c)!n_x!} \quad . \tag{1}$$

Whichever expression is assumed, the probability of never choosing an uncontaminated subset in η iterations is

$$p_f = (1 - \mathcal{G}(\mu))^\eta \quad . \tag{2}$$

Choosing an acceptably low probability of failure, say $p_f = 0.001$, this expression can be rearranged to determine the required number of iterations η.

Each iteration consists of randomly selecting data elements, estimating $\boldsymbol{\theta}$, and evaluating its quality using the whole data set. Now consider the case where each iteration is accelerated by the use of some process or test. The $T_{d,d}$ test is one such example, as it accelerates the evaluation step. A problem-specific process which accelerates the estimation step for video synchronisation is given in the following section. In general, any such speedup process will be adjustable, permitting faster iterations but also increasing the probability that each will fail.

To permit a general analysis, it is assumed that the speedup process is adjusted by varying a parameter ψ, which can be considered analogous to d in the $T_{d,d}$ test. If a small ψ value is chosen, each iteration will be faster, but more will be required to ensure the robust algorithm will fail with a probability no larger than p_f. The objective is to choose a value for ψ such that the robust algorithm can terminate in the smallest possible time, balancing the tradeoff between iteration speed and the probability that each iteration will fail.

Two functions are of relevence here. The first, $\mathcal{T}(\psi, \mu)$, estimates the time required for a single iteration. The second, $\mathcal{P}(\psi, \mu)$, represents the probability that the iteration will generate an acceptable estimate of $\boldsymbol{\theta}$, if the random subset contains no outliers. For a given ψ, the probability of η iterations failing is

$$p_f = (1 - \mathcal{G}(\mu)\mathcal{P}(\psi, \mu))^\eta \ . \tag{3}$$

Rearranging to solve for η, and multiplying by the expected iteration time yields a total estimated execution time of

$$\mathcal{T}_{\text{EX}}(\psi, \mu, p_f) = \mathcal{T}(\psi, \mu) \left\lceil \frac{log(p_f)}{log(1 - \mathcal{G}(\mu)\mathcal{P}(\psi, \mu))} \right\rceil \ . \tag{4}$$

The $\lceil . \rceil$ (ceiling) operator is used since a partial iteration is an impossibility. This operator and differing notation aside, an approximation to (4) is used in [2] to choose the optimal d for the $T_{d,d}$ test. If the assumed μ remains constant, choosing ψ which minimises (4) is expected to result in the best speedup.

Instead of a constant assumed number of inliers, many implementations of RANSAC style algorithms use an adaptive approach. The number of inliers μ is initially set to c, since for most problems any random subset will fit the model it generates. When an estimate classifying more inliers is found, μ is updated, and the required number of iterations η is revised. A new method will now be described, which seeks to combine such an adaptive assumption with per-iteration speedups.

An update of μ may have a considerable effect on the ψ which minimises (4). It is therefore appropriate to reevaluate ψ whenever μ is updated. Defining the termination condition in terms of the total number of iterations is no longer appropriate. Since different iterations may have used different ψ values, the probability of failure is no longer given by (3). Instead, it is necessary to maintain a history of the values used for ψ, and the number of iterations that each was used for. Assuming n_ψ different choices for ψ have been used, with the t^{th} choice denoted ψ_t and used for k_t iterations, the probability that all previous iterations failed is given by

$$\mathcal{F}(\mu) = \prod_{t=1}^{n_\psi} (1 - \mathcal{G}(\mu)\mathcal{P}(\psi_t, \mu))^{k_t} \ . \tag{5}$$

The algorithm terminates when $\mathcal{F}(\mu)$ is smaller than p_f. Evaluating $\mathcal{F}$ is not necessary after every iteration. An accumulated probability of failure, denoted $\hat{p}_f$ can be maintained instead. After an iteration when a best-so-far estimate of $\boldsymbol{\theta}$ is found, $\mathcal{F}(\mu)$ is computed, and the result is stored in $\hat{p}_f$. Otherwise, μ remains unchanged, so it is sufficient to multiply $\hat{p}_f$ by the probability that the previous iteration failed.

Throughout much of the robust estimation μ will be an *under-estimate*. This can adversely affect the choice of value for ψ, particularly during the earlier iterations when μ is low. To resolve this, μ can be considered *only* a lower bound. Since the true number of inliers is unknown before termination, it is assumed to be an instance of the random variable N, following a prior probability distribution $P(N = k)$. The choice of distribution is problem dependent. For example, in the case of fundamental matrix estimation, with point matches determined by correlation measures, it could be assumed that the probability of each match being an inlier is constant. Consequently, $P(N = k)$ will be a binomial distribution. As the lower bound is given by μ, it is also convenient to define an upper bound on N, denoted $n_{\max}$. For most problems this will be n_x, but the following section requires a different form.

Whenever μ is updated, a new ψ is chosen that seeks to achieve the best speedup for the remaining iterations. Determining the expected time until termination is problematic, since both μ and ψ may be revised on later iterations. Instead, we model the remaining time until the termination condition is reached for the *true* number of inliers, given the distribution $P(N = k)$, and assuming ψ will remain unchanged for the remaining iterations. Assuming there are k inliers in the data set, the probability that all previous iterations have failed is given by $\mathcal{F}(k)$. To achieve an overall probability of failure p_f, the remaining iterations must only fail with probability

$$\mathcal{R}(k) = min(1, p_f \mathcal{F}(k)^{-1}) \ . \tag{6}$$

The $min(.)$ operator is necessary to ensure that $\mathcal{R}(k)$ does not exceed 1.

The expected remaining time can be computed by summing (4) for each number of inliers k in the constrained range $[\mu, n_{\max}]$, and scaling each summand by a probability weight $w(k)$. Each $w(k)$ represents the conditional probability that there are truly k inliers, given the observed lower bound μ and the history of ψ values used on previous iterations. If the observed μ is considered an instance of a random variable N', then the choice for ψ is given by

$$\psi^* = \underset{\psi}{arg\,min} \quad \sum_{k=\mu}^{n_{\max}} w(k)\, \mathcal{T}_{\mathrm{EX}}(\psi, k, \mathcal{R}(k)) \ , \qquad (7)$$
$$\text{where } w(k) = P(N = k | N' = \mu) \ .$$

Once the termination condition is reached for a hypothesised number of inliers k, $\mathcal{R}(k)$ will be 1. The corresponding summand will be 0 and not affect the choice of speedup. Choosing ψ in this manner may result in suboptimal speedups since the true number of inliers is modelled by a distribution, though this is an unavoidable consequence of uncertainty. Also, if the probability of failure is reached for the *true* number of inliers without ever finding an acceptable estimate, more iterations will follow since termination is based on the lower bound. This is true of adaptive termination in general.

The probability weights $w(k)$ can be determined using Bayes' theorem, and are given by

$$w(k) = \frac{P(N' = \mu | N = k) P(N = k)}{\sum_{j=\mu}^{n_{\max}} P(N' = \mu | N = j) P(N = j)} \ . \qquad (8)$$

The conditional probabilities $P(N' = \mu | N = k)$ describe the probability of observing a lower bound of μ given the history of ψ values and the number of past iterations, assuming k inliers. The derivation of these is problem-specific, and requires additional assumptions. If this is deemed intractable, the weights may be modified to include an additional assumption that all previous iterations have failed. If event S denotes success on any previous iteration, the probability weights for (7) are then given by

$$w(k) = P(N = k | N' = \mu \cap \neg S) \ . \qquad (9)$$

As above, these weights can be determined using Bayes' theorem, but require few assumptions. Since it is assumed all previous iterations have failed, $P(\neg S | N = k)$ is given by $\mathcal{F}(k)$. Additionally, $P(N' = \mu | \neg S \cap N = k)$ is assumed to be some constant value, independent of k, since the assumption of prior failure means the number of inliers had no effect on the observed μ. This leads to probability weights given by

$$w(k) = \frac{\mathcal{F}(k) P(N = k)}{\sum_{j=\mu}^{n_{\max}} \mathcal{F}(j) P(N = j)} \ . \qquad (10)$$

Note that in equations (8) and (10), the range of the denominator is restricted. This is due to the assumption that we cannot observe a μ which is higher than the true number of inliers.

This section has described a new process which combines an adaptive estimate of the number of inliers with an adaptive choice of speedup that seeks the optimal tradeoff between the speed and reliability of the iterations. This approach is termed RATSAC (Random Adaptive Tradeoff Sample Consensus), and is summarised below.

```
Algorithmic Summary of RATSAC:
  Choose small overall probability of failure $p_f$
  Number of inliers $\mu \leftarrow c$
  Accumulated probability of failure $\hat{p}_f \leftarrow 1$
  Compute $\psi^*$ as given by (7)
  while $\hat{p}_f > p_f$ do
    $\hat{p}_f \leftarrow \hat{p}_f(1 - \mathcal{G}(\mu)\mathcal{P}(\psi^*, \mu))$
    Randomly select $c$ samples from $\{\boldsymbol{x}_i\}$
    Compute $\boldsymbol{\theta}$ and evaluate, according to $\psi^*$
    if $\boldsymbol{\theta}$ is the best so far then
      Set $\mu$ to the number of classified inliers
      $\hat{p}_f \leftarrow \mathcal{F}(\mu)$ as given by (5)
      Compute $\psi^*$ as given by (7)
    end if
  end while
  Return the best $\boldsymbol{\theta}$ and associated classifications
```

3 Video Synchronisation

Synchronisation is concerned with determining the temporal relationship between two or more videos. For videos recorded at constant frame rates, this relationship is linear. If frame indices in two videos are denoted f and f', then

$$f' = a + bf \qquad (11)$$

denotes the frame in the second video that was recorded at the same time as f. The parameters (a, b) define the line of synchrony, where b is a ratio of the cameras' frame rates, and a represents an offset in time. For known frame rates, synchronisation only requires the estimation of a. This parameter is typically found by imposing spatial constraints known to hold in the presence of synchrony. For example, projections of a moving scene point should satisfy the epipolar constraint in synchronous frames.

Feature based robust synchronisation algorithms tend to operate by iterating over a range for a, performing a robust estimation for each value. In [9], all moving feature points from approximately synchronous frames (according to a hypothesised a) are considered matches. In [1], matches consist of entire 2D trajectories. This reduces the number of matches, and therefore the number of iterations required. In

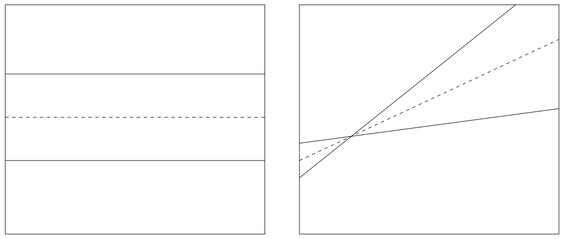

Figure 1. Epipolar line pairs (solid) and the result of a linear interpolation (dashed)

both these algorithms, the spatial relationship between cameras is assumed to remain constant over time.

A histogram method for the synchronisation of independently moving cameras is presented in [7]. This method assumes that corresponding stationary scene points have been observed in both videos, and can be used to define the epipolar geometry between any pair of frames. Specifically, $\mathsf{F}_{i,j}$ is the fundamental matrix relating frame i from the first video, and frame j from the second. A moving scene point projects to trajectories h and h' in the two videos, such that $\boldsymbol{p}_{h,i}$ is the image location in frame i of the first video, and $\boldsymbol{p}'_{h',j}$ is the location in frame j of the second video.

A hypothesised value of the offset a can be assessed by measuring the epipolar error associated with a pair of image points in synchronous frames. This can be problematic however, since the line of synchrony may not intersect integer frame pairs. The solution proposed in [7] consists of interpolating epipolar lines to enable the measurement of epipolar errors for non-integer frame indices. Consider frame pair (i, j) lying on the line of synchrony defined by (a, b), where i is an integer, but j is not. Clearly $\boldsymbol{p}'_{h',j}$ and $\mathsf{F}_{i,j}$ are undefined, preventing the measurement of epipolar errors. However, the two eipolar lines in frame i of the first video, generated by the 'closest' to synchronous frames from the second video are

$$\boldsymbol{l}_1 = \mathsf{F}_{i,\lfloor j \rfloor} p'_{h',\lfloor j \rfloor} \quad \boldsymbol{l}_2 = \mathsf{F}_{i,\lfloor j \rfloor+1} p'_{h',\lfloor j \rfloor+1} \ , \qquad (12)$$

where $\lfloor . \rfloor$ denotes the floor operator. Once computed, each of these line vectors is normalised by a function $\mathcal{N}$. This function normalises a line vector such that the sum of the squares of its first to elements is 1. One of the line vectors is also negated if their difference in orientation is greater than π. Once normalised, these line vectors can be linearly interpolated, so as to approximate the location of an epipolar line for the non-integer frame index j, given by

$$\hat{\boldsymbol{l}} = (1 + \lfloor j \rfloor - j)\mathcal{N}(\boldsymbol{l}_1) + (j - \lfloor j \rfloor)\mathcal{N}(\boldsymbol{l}_2) \ . \qquad (13)$$

Using this line, an epipolar error in frame i can now be measured, given by

$$d(\boldsymbol{p}_{h,i}, \hat{\boldsymbol{l}}) \ , \qquad (14)$$

where $d(.)$ denotes the euclidean distance between a point and a line. The histogram method in [7] relies on the fact that rearranging (14) permits the search for frame pairs that exhibit an epipolar error of exactly 0. Such pairs, termed synchrony pairs, exist since the space of interpolated lines between $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$ is a bounded *region* in the image, as shown in Fig. 1. To find synchrony pairs, consider choosing an integer frame index i from the first video, and consecutive integer frame indices $(k, k+1)$ from the second. Image points in the $(k, k+1)$ frame pair are used to generate the line vectors $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$ in frame i of the first video, by assuming $k = \lfloor j \rfloor$. If a j exists between k and $k+1$, such that the interpolated line $\hat{\boldsymbol{l}}$ *exactly* passes through $\boldsymbol{p}_{h,i}$, then the frame index pair (i, j) is considered a synchrony pair.

Each synchrony pair found defines an estimate of the frame offset a, and a histogram can be built from these values, with a tentative estimate of synchronisation then being provided by the tallest histogram peak.

A search for synchrony pairs need not be exhaustive. An accelerated search can be conducted by randomly selecting just a small number of frames from first video. For each chosen frame i, a search can be conducted across all consecutive frame pairs $(k, k+1)$ from the second video. A reciprocal search can then be performed by randomly choosing a small number of frames from the second video. This forms the basis for an accelerated robust estimation, where the speedup parameter ψ controls how many frames are randomly chosen in each video.

A difficult case is considered here, where m and m' trajectories of moving scene points are tracked in the first and second videos respectively. It is assumed that no correlation information is available, so the set of matches consists of *all* pairs of trajectories. Some matches will preclude others, since two trajectories with overlapping frame ranges in one video cannot both be matched with the same trajectory in the other video. A single iteration of the robust estimation consists of choosing a trajectory pair at random, performing an accelerated search for synchrony pairs, refining the initial estimate given by the histogram peak, and evaluating the resulting synchronisation estimate against all matches.

Before the RATSAC approach to choosing speedup values can be employed, it is necessary to model both the time and probability functions $\mathcal{T}(\psi, \mu)$, and $\mathcal{P}(\psi, \mu)$, as described in section 2.

Trajectories h and h' in the first and second videos have lengths measured in frames denoted v_h and $v'_{h'}$. If ψ controls the proportion of frames chosen for the synchrony pair search, then this search takes time

$$\mathcal{T}_{\text{SEARCH}}(\psi) = \psi(2E[v_h v'_{h'}] - E[v_h] - E[v'_{h'}]) \ . \qquad (15)$$

The expected trajectory length expressions $E[.]$ can be computed in advance by examining each match in turn. Note that this expression is quadratic, and reflects the expense of a synchrony pair search. The iteration time $\mathcal{T}(\psi, \mu)$ is given as

$$\mathcal{T}(\psi, \mu) = \mathcal{T}_{\mathrm{SEARCH}}(\psi) + \mathcal{T}_{\mathrm{REM}}(\psi, \mu) \ , \qquad (16)$$

where $\mathcal{T}_{\mathrm{REM}}$ represents the time for the portion of the iteration which is not directly accelerated by the choice of ψ. The portion of the estimation step not affected by ψ consists of resetting the histogram to 0, incrementing the cells, locating the peak, and refining the offset a by minimising the average of squared interpolated epipolar errors for the chosen trajectory pair. To speed up the refinement, epipolar errors are only measured for a coarse uniform sampling of frames from each video, and the minimisation, performed by Levenberg-Marquardt, terminates after no more than 15 iterations. The resulting line of synchrony defines a range of frames in each trajectory for which epipolar errors are measurable. The estimate is discarded if these ranges are too small, or if the the average of these errors fails a threshold test. For iterations passing these tests, the synchrony estimate is assessed by evaluating the average squared error for every match. Initially, this is done using a coarse sampling in the same fashion as used for the minimisation, and matches are classified as inliers or outliers accordingly. If the costs and classifications imply the estimate is the best found so far, a full evaluation follows. An expression for $\mathcal{T}_{\mathrm{REM}}$ (ommited here) can therefore be derived with assumptions regarding how many synchrony pairs are found, how many iterations pass the range and measurability tests, and how many measurable residuals are available. Additionally, to determine the average time for evaluation, it is assumed that the cost used to rank an estimate is the same for both a full and coarse sampling.

The function $\mathcal{P}(\psi, \mu)$ is far simpler, and has been derived empirically using the observation that at ψ of 0.1 yields a high probability of success for simple point and camera motions. Accordingly,

$$\mathcal{P}(\psi, \mu) = \psi^{0.1} \ . \qquad (17)$$

Note that a ψ of 0 implies guaranteed failure, and a ψ of 1 implies guaranteed success for a correct match.

The robust cost, either full or coarse, is obtained by measuring the average squared epipolar residuals associated with each match according to the synchrony estimate. Matches can then be classified as inliers or outliers by a simple threshold test. Classification must occur in ascending order of the associated costs, to ensure the best matches are kept and that no conflicts occur. If the sum of squared residuals for a given match is denoted $s_{h,h'}$, and the number of measurable residuals contributing to the sum is $w_{h,h'}$, and the matches are partitioned into inlier and outlier sets denoted M_{I} and M_{O}, the robust cost is

$$SIZE(M_{\mathrm{I}}) \frac{\sum_{(h,h') \in M_{\mathrm{I}}} s_{h,h'}}{\sum_{(h,h') \in M_{\mathrm{I}}} w_{h,h'}} + SIZE(M_{\mathrm{O}})t \ , \qquad (18)$$

where t is the threshold. Note this cost guarantees that, in a fashion similar to MSAC, two estimates with the same number of associated inliers are ranked according to the average cost associated with those inliers. After the robust estimation, this cost can be reduced using Levenberg-Marquardt to refine the synchronisation estimate, and possibly reclassify the matches.

All that remains is to define the discrete probability distributions to be used in selecting ψ. The distribution for the number of inliers $P(N = k)$ is assumed to be uniform in the range of 1 to $n_{\max}$, since *all* trajectory pairings are given as matches. The upper bound $n_{\max}$ can not be given by n_x, since matches may conflict. Determining the maximum sized set of non-conflicting matches is an NP-complete problem. Instead, an initial value of $min(m, m')$ is used, and updated later if it is found to be too low. Such a choice is appropriate for long 'unbroken' trajectories. Estimating the conditional probabilities for the probability weights is also complex, due to the fact that matches may conflict, so the expression for the assumption of prior failure (10) is used.

4 Results

Using RATSAC for the synchronisation of moving cameras has been tested with both synthetic and real examples. In the synthetic case, three temporal configurations are considered, with each test being repeated 1000 times for random stationary and linearly moving scene points. All moving points are visible in every frame, with video lengths denoted n and n'. There are 10 moving point trajectories for each video, but only 5 are in common. This admits a worst case scenario of finding no correct matches, and 10 non-conflicting incorrect matches from a possible set of 95. Point locations are perturbed by Gaussian noise, such that the average squared distance to the true location is 1 pixel. The required probability of failure p_f was chosen to be 0.001.

For each test, RATSAC provides an initial estimate of the frame offset and inlier/outlier classification. These are then refined by a minimisation of the robust cost function. Table 1 shows the results of the classifications. Note that every test correctly classified the 5 correct matches as inliers. Furthermore, no test wrongly classified more than 1 incorrect match. In a sense, none of these 3000 tests actually failed. The maximum error in frame offset for all tests was 0.55, synchronising the videos to within almost one half of a frame. The median errors in frame offset for each setup were 0.022, 0.035, and 0.032, indicating a highly accurate level of synchronisation. The higher incidence of incorrect matches in the second setup is most likely due to the smaller overlap in time. An incorrect match is more likely to have low errors across a short range of frames than a longer one. The quality of these results also validates the choice for function $\mathcal{P}$, by demonstrating an acceptably low probability of failure.

The initial ψ values chosen for each setup were 0.01, 0.01, and 0.02, indicating that fast iterations are given preference over slower more reliable ones. The resulting execution times of the robust estimation (without the subsequent minimisation) were compared with cases where ψ was unconditionally set to 1, which represents exhaustive synchrony pair searches, with no speedup employed. By comparison, The percentage of time saved by using an adaptive ψ for each of the three setups is 85%, 88%, and 68%. This reduction demonstrates the clear benefit of using random sampling to find a synchronisation estimate with histogram methods.

RATSAC was also tested 1000 times on a video pair recorded by independently moving handheld cameras. The footage captured shows three purple balls, bouncing repeatedly. The ball locations were determined using colour and gradient measures, and combined to form 2D trajectories with simple distance constraints. Due to occasional tracking failure and occlusions, there are 22 trajectories in the first video, and 19 in the second, yielding a total of 418 possible matches. The initial ψ chosen was 0.042, again indicating a preference for many fast iterations.

The synchronisation achieved by RATSAC and a minimisation of the robust cost function is satisfactory for this video pair. When compared to a manual estimate, the median error in frame offset across the 1000 tests was 0.12. This is well within the range of visual error for a manual estimate, and suggests sub-frame synchronisation has been achieved. Match classification was also accurate, with 19 correct matches being identified in every test. According to the manual synchronisation estimate, this is the true number of inliers. Additionally, no tests wrongly classified an incorrect match as an inlier.

Over 1000 tests, the execution time was compared to that of tests where ψ was unconditionally set to 1. The percentage of time saved by using an adaptive approach was 18%. This is a significantly lower saving than that achieved in the synthetic tests, and is most likely due to the shorter trajectory sizes. Shorter trajectories yield faster synchrony pair searches, so evaluation will occupy a greater proportion of the iteration time.

An illustration of the synchronisation achieved by RATSAC is shown in Fig. 2, depicting cropped regions of frames from the two videos, displayed with their relative position determined by synchrony. Reconstructions of the 3D ball trajectories are shown from new views in Fig. 3. These reconstructions are obtained by assuming that, for each match, the path of the ball follows a Bezier curve model parameterised by time. Each curve can then be estimated from image correspondances by minimising reprojection error.

5 Conclusions

Section 2 describes a new robust estimation strategy, incorporating an adaptive termination condition with a probability based decision process to select ideal speedup parameters. The underlying premise is general enough to be applied to a variety of problems, and can be used in conjuncton with various robust cost measures. Additionally, a histogram based synchronisation method for moving cameras has been extended to the robust case, and demonstrates a high rate of success for identifying matches, for both real and synthetic test cases, even without the prior use of correlation measures.

References

[1] Y. Caspi, D. Simakov, and M. Irani. Feature-based sequence-to-sequence matching. In *ECCV Workshop on Vision and Modelling of Dynamic Scenes (VAMODS)*, 2002.

[2] O. Chum and J. Matas. Randomized RANSAC with $T_{d,d}$ test. In *Proc. British Machine Vision Conference*, pages 448–457, 2002.

[3] O. Chum and J. Matas. Matching with PROSAC - progressive sample consensus. In *Proc. Conference on Computer Vision and Pattern Recognition*, volume 1, pages 220–226, 2005.

[4] O. Chum, T. Werner, and J. Matas. Epipolar geometry estimation via RANSAC benefits from the oriented epipolar constraint. In *Proc. International Conference on Pattern Recognition*, volume 1, pages 112–115, 2004.

[5] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24:381–395, 1981.

[6] D. Nistér. Preemptive RANSAC for live structure and motion estimation. In *Proc. International Conference on Computer Vision*, pages 199–206, 2003.

[7] D. Pooley, M. Brooks, A. van den Hengel, and W. Chojnacki. A voting scheme for estimating the synchrony of moving-camera videos. In *Proc. International Conference on Image Processing*, volume 1, pages 413–416, 2003.

[8] P. Rousseeuw and A. Leroy. *Robust Regression and Outlier Detection*. John Wiley & Sons, Inc., 1987.

[9] G. P. Stein. Tracking from multiple view points: Self-calibration of space and time. In *DARPA Image Understanding Workshop*, pages 1037–1042, 1998.

[10] P. Torr. Bayesian model estimation and selection for epipolar geometry and generic manifold fitting. *International Journal of Computer Vision*, 50(1):35–61, 2002.

[11] P. Torr and A. Zisserman. Robust computation and parameterization of multiple view relations. In *Proc. International Conference on Computer Vision*, pages 727–732, 1998.

[12] P. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000.

Setup				Percentage of tests with classified matches		
n	n'	a	b	5 Correct	0 Incorrect	1 Incorrect
80	100	10.63	1.1875	100	99.5	0.5
80	100	42.3	1.1875	100	98.3	1.7
20	100	10.63	4.9375	100	99.7	0.3

Table 1. Correct and incorrect inliers classified by RATSAC followed by a minimisation of the robust cost function.

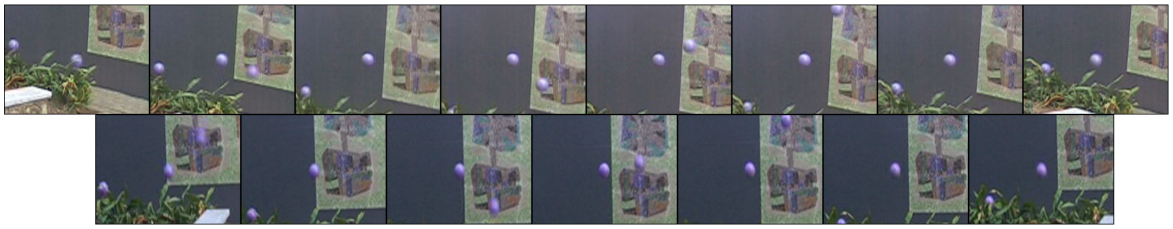

Figure 2. Regions of frames from video 1 (top) and video 2 (bottom), with horizontal position determined by synchrony.

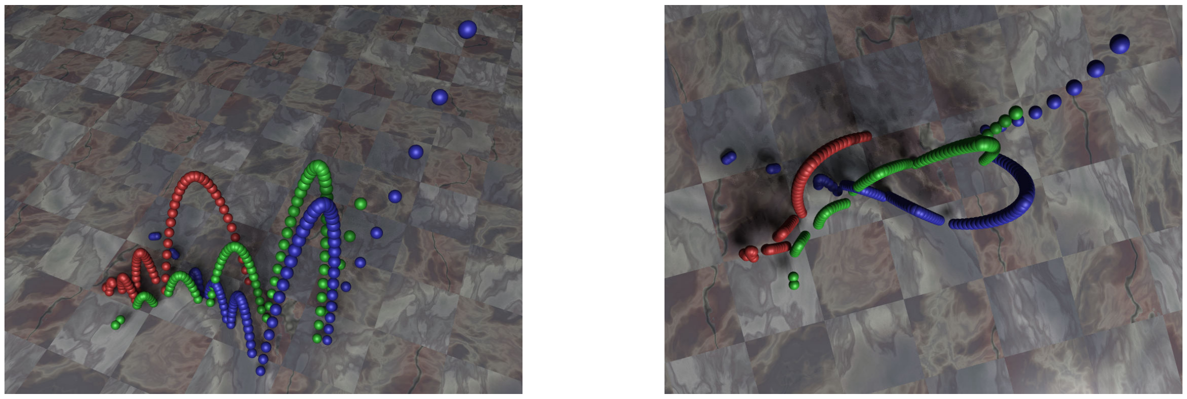

Figure 3. 3D Ball trajectories, projected to virtual (non-existent) camera locations.