Hindawi Publishing Corporation
ISRN Machine Vision
Volume 2013, Article ID 261956, 8 pages
http://dx.doi.org/10.1155/2013/261956

Research Article

Resection-Intersection Bundle Adjustment Revisited

Ruan Lakemond, Clinton Fookes, and Sridha Sridharan

Image and Video Research Laboratory, Queensland University of Technology, GPO Box 2434, 2 George Street, Brisbane, QLD 4001, Australia

Correspondence should be addressed to Ruan Lakemond; ruan.lakemond@gmail.com

Received 17 August 2013; Accepted 18 November 2013

Academic Editors: A. Gasteratos and M. Pardàs

Bundle adjustment is one of the essential components of the computer vision toolbox. This paper revisits the resection-intersection approach, which has previously been shown to have inferior convergence properties. Modifications are proposed that greatly improve the performance of this method, resulting in a fast and accurate approach. Firstly, a linear triangulation step is added to the intersection stage, yielding higher accuracy and improved convergence rate. Secondly, the effect of parameter updates is tracked in order to reduce wasteful computation; only variables coupled to significantly changing variables are updated. This leads to significant improvements in computation time, at the cost of a small, controllable increase in error. Loop closures are handled effectively without the need for additional network modelling. The proposed approach is shown experimentally to yield comparable accuracy to a full sparse bundle adjustment (20% error increase) while computation time scales much better with the number of variables. Experiments on a progressive reconstruction system show the proposed method to be more efficient by a factor of 65 to 177, and 4.5 times more accurate (increasing over time) than a localised sparse bundle adjustment approach.

1. Introduction

The nonlinear error minimisation step commonly referred to as bundle adjustment is a key step in many systems that recover projective geometry from image correspondences. While linear solutions are available in most cases, their accuracy is generally below requirements. Linear methods usually only serve as good initialisation for bundle adjustment [1]. Bundle adjustment remains a relatively computationally expensive part of the reconstruction process, despite a number of attempts to make it more efficient [2–4].

Resection-intersection is an implementation of bundle adjustment that interleaves the steps of refining camera parameters (resection) and 3D points (intersection). In this configuration each camera is treated as being independent of other cameras and each point is treated as being independent of others. While this has a number of advantages leading to reducing time spent on each iteration, the convergence rate is slow, primarily because interactions between variables are not accounted for, and because of redundancies between the two steps.

This paper proposes modifications to the classic resection-intersection algorithm that improve its accuracy and convergence rate and efficiently manages the propagation of parameter updates through a large, weakly overlapping network of cameras. The proposed algorithm is highly scalable, robust to high initial error, converges rapidly, and attains an error level comparable to a full bundle adjustment. It is a good choice for systems with many variables. Where high accuracy is paramount, the proposed approach can be inserted between attaining an initial solution and a full bundle adjustment to greatly reduce the amount of work that the full BA stage needs to perform. The update propagation control system makes the algorithm inherently well-suited to progressive reconstruction problems, where reconstructed data is added in batches (e.g., a Structure from Motion (SFM) system). Refinement effort is localised to the new data, while significant parameter updates are propagated to the existing data. Events such as loop closures are handled effectively, without the need for any additional network modelling or event detection.

2. Background

In general terms, bundle adjustment is the process of refining the parameters of a set of cameras and structure points to optimally fit a set of image observations. The problem has been studied in great depth [1].

Where the reconstruction involves one or more moving cameras, the number of observations and parameters to optimise overgrows rapidly as the video length increases. The resulting computational complexity can be a hindrance to achieving large scale or real time reconstruction. The primary approach to improving the efficiency of bundle adjustment is to take advantage of the inherent sparsity of the problem. Sparsity arises from the fact that not all world points are visible in all cameras and due to the independence of the model variables. Each point is independent of every other point and each camera is independent of every other camera, and the solution is only coupled through commonly visible points. An example implementation that uses sparse data structures and solvers to perform a Newton iteration is [2]. This implementation uses a matrix data structure that only stores nonzero values and it does not include zero elements in the matrix computations. Despite using sparse representations, the Jacobian and Hessian matrices require large amounts of memory, and operations on these matrices can be time consuming. Implementations for multiple processors and graphics processor type hardware have recently been made available [5] to apply more processing power to this demanding problem.

Many reconstruction problems are approached in a sequential manner. Data is added one camera at a time or one batch of cameras at a time. The nonlinear minimisation step is also performed progressively, since the accuracy of the solution at any time influences the subsequent reconstruction process. In these cases it is common that data that has already been refined does not need to be refined much more in subsequent steps. Using bundle adjustment directly in this case results in exponentially increasing processing time as the data grows. Most of the processing effort results in little improvement in the model, as previous refinement has dealt with most of the error.

A number of approaches have been proposed to attempt to localise the refinement effort to the data that requires refinement. Hierarchical bundle adjustment [6] (HBA) breaks the reconstruction up into small sections of cameras and recursively combines the data again in a hierarchical scheme. A typical scheme is to refine triplets of cameras and to recursively combine pairs of neighbouring groups using a 3D registration algorithm [3]. The registration step is required because a global datum (reference coordinate system) can not be defined for all groups. Local bundle adjustment [4] (LBA) only refines the most recent n frames while including the most recent $N > n$ frames in the cost function in order to constrain the datum. This approach removes the need for registering subsections of the structure but does result in progressive error accumulation and datum drift over long sequences.

The above localised methods make use of a camera-wise localisation and do not provide a good approach to point-wise localisation. A side effect is that in many cases only a subset of the observations of a given world point are included in the minimisation at a time. This increases error drift and accumulation and introduces inconsistencies in datum control. In situations where the camera path intersects or nearly intersects itself (a scenario commonly referred to as loop closure) this problem becomes most pronounced, since the error between spatially close cameras can become very large relative to the error between temporally adjacent cameras.

Relative bundle adjustment [7, 8] (RBA) models the camera network using relative pose transformations between cameras within a network graph. This is said to eliminate the need for a global datum. Loop closures are dealt with by adding additional edges and relative pose parameters to the graph where cameras meet, though it is not clear how camera pairs defining a loop closure are identified. Processing localisation is achieved by searching the graph, starting at the latest camera, for cameras where the reprojection error changes by more than a set threshold. All points visible to these cameras are included in the optimisation. Additional cameras that view any of these points are included in the optimisation, but their parameters are fixed. The drawbacks of this method are the need to maintain a camera connectivity graph and the need to detect loop closures in some manner. Retrieving camera's parameters requires computing a chain of cameras between the reference and target camera, due to the relative camera representation.

Resection-intersection (RI) treats each camera and point as being independent and refines each of these individually, instead of jointly. The RI technique has several characteristics that could be beneficial to reducing computation time as follows.

(i) The sparsity of the overall problem is exploited to the full; refining a single point or camera is a dense problem.

(ii) The datum is very well defined during each refinement step; all observed world points define the coordinate frame for the camera being refined and all the observing cameras define the coordinate frame for the point being refined.

(iii) Reregistration is not required, as in hierarchical bundle adjustment [3], for example, and the algorithm is much less susceptible to relative datum drift.

(iv) Processing is focussed on smaller amounts of data for longer, since the refinement process is applied to a relatively small set of parameters and observations at a time. Depending on the hardware architecture used, this can eliminate the transfer of data between larger, slower memory resources and the processor's fast caches for the duration of the inner refinement loop, thereby greatly reducing delays.

(v) Parallel implementation of the algorithm is made simple. All cameras can be refined in parallel and all points can be refined in parallel, as long as there is good separation between the camera step and point step.

```
(1.1) begin
(1.2)   while Significant total change recorded do
(1.3)     foreach Camera with significant change do
(1.4)       Reset camera's change to zero.
(1.5)       Apply LM refinement to camera parameters.
(1.6)       Distribute error change resulting from refinement to points visible in current camera.
(1.7)     end
(1.8)     foreach Point with significant change do
(1.9)       Reset point's change to zero.
(1.10)      Triangulate using linear method.
(1.11)      Apply LM refinement to point.
(1.12)      Distribute change due to refinement to observing cameras.
(1.13)    end
(1.14)  end
(1.15) end
```

Algorithm 1: Resection-intersection with linear triangulation and update tracking.

(vi) Processing can easily be localised by omitting to refine points or cameras that do not need refinement.

Unfortunately, algorithms based on resection-intersection converge slowly because they do not model the interactions between variables. The size of the Newton step is consistently underestimated and parameter updates are propagated slowly through the network. The effects of slow convergence outweigh the reduction in individual iteration times, leading to poor performance [1].

3. Improved Resection-Intersection

Algorithm 1 gives a brief overview of the proposed resection-intersection (RI) algorithm. Let a model element be defined as a set of parameters associated with either one camera or one point. The RI algorithm essentially refines one element at a time. Each iteration consists of refining cameras (resection stage) then refining points (intersection stage). The Levenberg Marquardt (LM) algorithm is used to perform refinement, since the refinement of individual elements is a dense problem. The proposed algorithm includes two modifications over existing algorithms: an additional linear triangulation step and a change tracking system. To help ensure stability, an update is only applied if it results in reduced error.

3.1. Repeated Linear Triangulation. Resection and intersection are two subproblems with significantly different properties. Cameras are more complex entities than points. The linear or quasilinear methods used to initialize cameras [9, 10] can be unstable and imprecise, that is why bundle adjustment is so important. The linear multiview triangulation algorithm used to find points is, in contrast, simple and highly stable (given sufficient camera baseline) [3]. The poor accuracy of initial camera estimates is the most significant source of error in estimating point locations.

A linear triangulation step is added into the intersection stage, so that each point is retriangulated before applying LM refinement. This step takes advantage of the stable, inexpensive triangulation method to apply the camera updates to the point, before continuing on to nonlinear refinement. The effect is that a large update is applied to the point with a small amount of computation effort.

3.2. Change Tracking. The second major addition to the RI algorithm is an update tracking system. A scalar change accumulator is associated with each model element, where a model element is defined as either a single camera or a single point. For every parameter update, the resulting change in reprojection error is recorded against each model element affected by the updated parameters. For example, when a camera is refined, the change in error is computed as $e_\Delta = 1 - (e_1/e_0)$, where e is the reprojection error averaged over all observations made by that camera. e_0 is computed before refinement and e_1 is the error after refinement. The change, e_Δ, is then added to all the world points visible in the current camera, while the change value of the camera that was refined is set to zero. The total change affecting each point is accumulated from all observing cameras and normalised according to

$$e_p = n^{-1} \sum_{i=1}^{m} e_{\Delta i}, \tag{1}$$

where n is the number of cameras viewing point p and m is the number of refinement operations applied to these cameras (more than one refinement pass may be applied before e_p is reset to zero). An equivalent approach is applied while refining points to distribute the change in error to observing cameras.

A model element is only refined if its normalised change value is above a threshold. The result is that each model element is only refined if it is directly associated with significant updates made elsewhere in the model. The change value is computed such that the local change value is of comparable scale as the global change in reprojection error. The change threshold can therefore be set equal to the overall convergence threshold. Modifications made outside the refinement algorithm system must also be recorded in the change tracking system. All new elements and elements with new observations are given a high change value so that they

are refined at least once. Deletions (such as deleting outlier observations) result in a unit change which is added to any associated world points and observing cameras.

The change tracking system reduces computational effort by only propagating parameter updates related to significant change. It is also a very effective method for localising computational effort in progressive reconstruction systems, where data is added to the solution in sections. All new data is refined at least once and older data is only refined if needed. This method handles loop closures intrinsically, without the need to model the network topology.

4. Experimental Validation

Experiments were carried out using real and simulated data to assess the performance of the modified resection-intersection algorithm. This paper considers the case where a large volume of structure and observation data must be processed and where accuracy is of high priority. Both simulated and real data are processed progressively; cameras and points are added in chronologically ordered batches and refinement is applied at key frames.

The sparse bundle adjustment (SBA) implementation from [2] was used as reference implementation. A localised version of SBA (Local SBA) was produced by providing the SBA algorithm with a subsection of the data, similar to what is done in [4]. A sequence of cameras spanning four key frames is refined, with additional cameras spanning a further eight key frames included but held fixed. The fixed sequence is included in the cost function computation, but the camera parameters are not altered. This provides a method for constraining the datum for the subsection that is refined. The lengths of these sections were manually tuned to balance accuracy against processing times. Shorter sequences result in lower computational cost but increased error drift.

Three versions of the Resection-intersection algorithm were tested as follows:

(1) RI with LM: the original RI algorithm using LM refinement for both cameras and points,

(2) RI with triangulation: uses LM for cameras but linear triangulation for points,

(3) RI with triangulation + LM: uses LM for cameras; points are refined by first using linear triangulation, then LM, as outlined in Algorithm 1.

All RI-based algorithms use the change tracking system detailed in Section 3.2.

All experiments were implemented in C/C++ and executed in a single thread on an 8-core desktop computer with 16 Gb of local memory.

4.1. Simulation Experiments. Simulated data was produced by randomly generating a cloud of 3D points and a set of cameras with fixed calibration, outside the convex hull of the points. Image observations were produced by projecting the points using the cameras. Random noise with a Gaussian distribution and mean amplitude of 5% of the point cloud width was added independently to all points, camera pose parameters and observations. The point data was generated to simulate points that are becoming visible and eventually occluded as the sequence continues. The sequence begins with an initial set of points. After each batch of cameras has been added and projections have been taken, some new points are added and some old points are removed before producing the next set of cameras and projections.

Two sets of data were produced: a small set to accommodate evaluation of the time consuming SBA algorithm and a large set to test the faster algorithms on more data. The small set consists of 400 initial points, with 50 points added and removed after every set of 10 cameras is added. A total of 400 cameras are generated. The large set consists of 10000 initial points, with 1000 points added and removed after every set of 10 cameras is added. A total of 1000 cameras are generated.

Two experiments were performed: a convergence experiment and a progressive data addition experiment.

4.1.1. Convergence. The entire small data set is used for this experiment. Each refinement algorithm is applied once to the data and the results of each iteration of the algorithm are recorded to study its convergence path. Convergence thresholds were set very low (10^{-12}) in order to have the algorithms run for many iterations, up to a maximum of 100 iterations. The results are shown in Figure 1, where the solution reprojection error is plotted against the time at which an iteration is completed.

The original RI algorithm completes each iteration in relatively short time but converges very slowly, as previously observed [1]. The proposed method using both linear triangulation and LM to refine points is much more effective, while individual iterations take only a fraction more time. Compared to SBA, the proposed method yields a final error only 0.2% higher in much shorter time. Using only triangulation to refine points results in the shortest iteration time, while the end result is almost exactly equal to the method that also uses LM refinement. This method converges in only 6 iterations. The rapid convergence can be attributed to the fact that the linear triangulation method is optimal under Gaussian noise. Under real image conditions, its performance may be significantly different.

4.1.2. Progressive Reconstruction. This experiment treats the data as a progressive reconstruction that requires refinement after each key frame. Data is added to the solution in batches of 10 cameras at a time, as described above. A refinement algorithm is applied after each batch of data is added and its performance measured. Results are presented by plotting reprojection error and refinement processing time against the camera index at which refinement was performed. The reprojection error at index j is computed as the error for cameras 0 to j.

Results for the small data set are presented in Figure 2, and Figure 3 shows the results for the large data set.

All methods initially yield a low error due to overfitting to the initially small amount of data. The time taken to complete each full SBA pass increases rapidly as data is added. The SBA method achieves a model error only 6% lower than

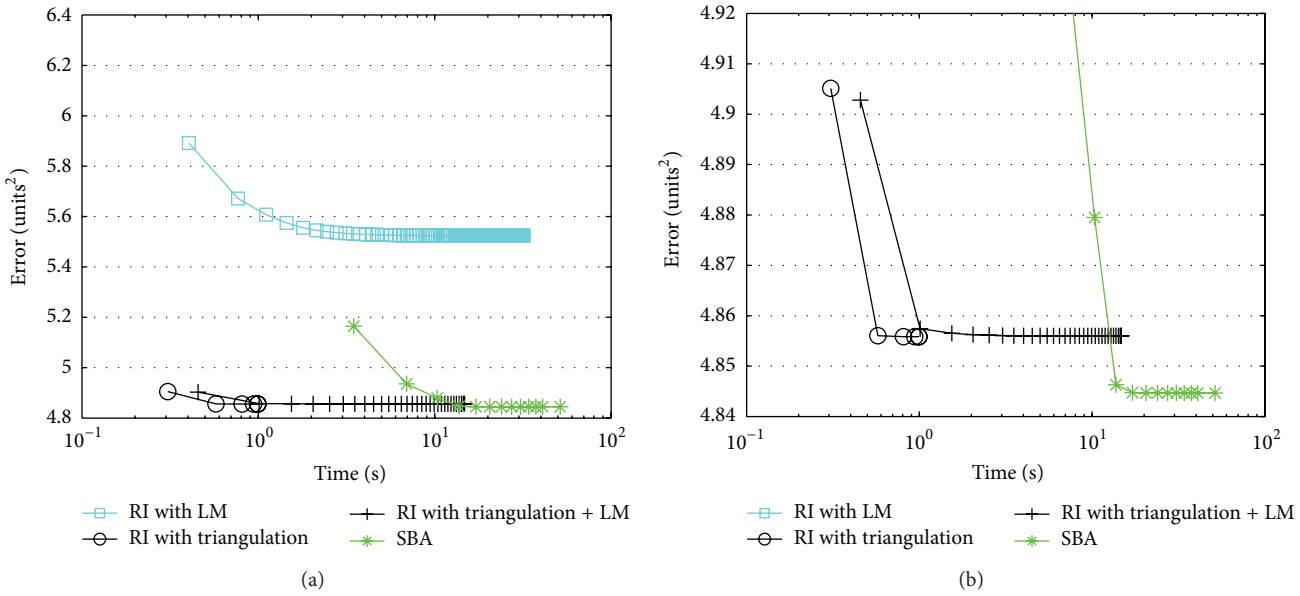


FIGURE 1: Progress of iterative refinement processes plot against process time. Each marker indicates the end of an iteration of each algorithm. Both plots show the same data at different error scales.

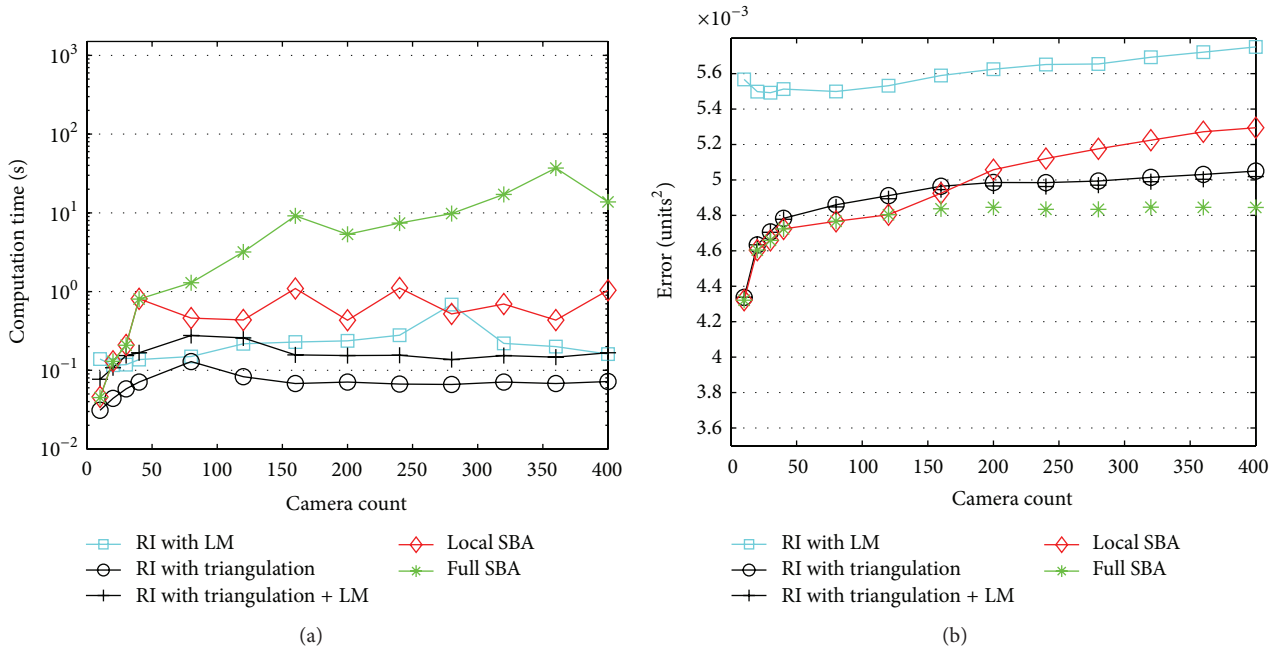


FIGURE 2: Small simulated data set results. Data is added to the reconstruction progressively and nonlinear refinement is applied at key frames. The refinement time (a) and resulting error (b) are plotted against key frame indices.

the other methods in the small data experiment. Both LSBA and the proposed methods show approximately constant time performance as the data set grows. LSBA initially produces the same error as SBA. Once the processing window comes into effect, its processing time stops increasing and its error starts increasing. For the large data test, the proposed RI method with both triangulation and LM refinement requires on average 6.3 times less computation time than LSBA. The method using only triangulation is 13.7 times faster than LSBA but shows greater error drift as the sequence length grows. The original RI algorithm again shows significantly higher error and even becomes unstable during the large data test.

4.2. Real Data Experiment. The real data experiment was produced using an SFM system designed for high accuracy tracking. For each image in the sequence, the camera pose

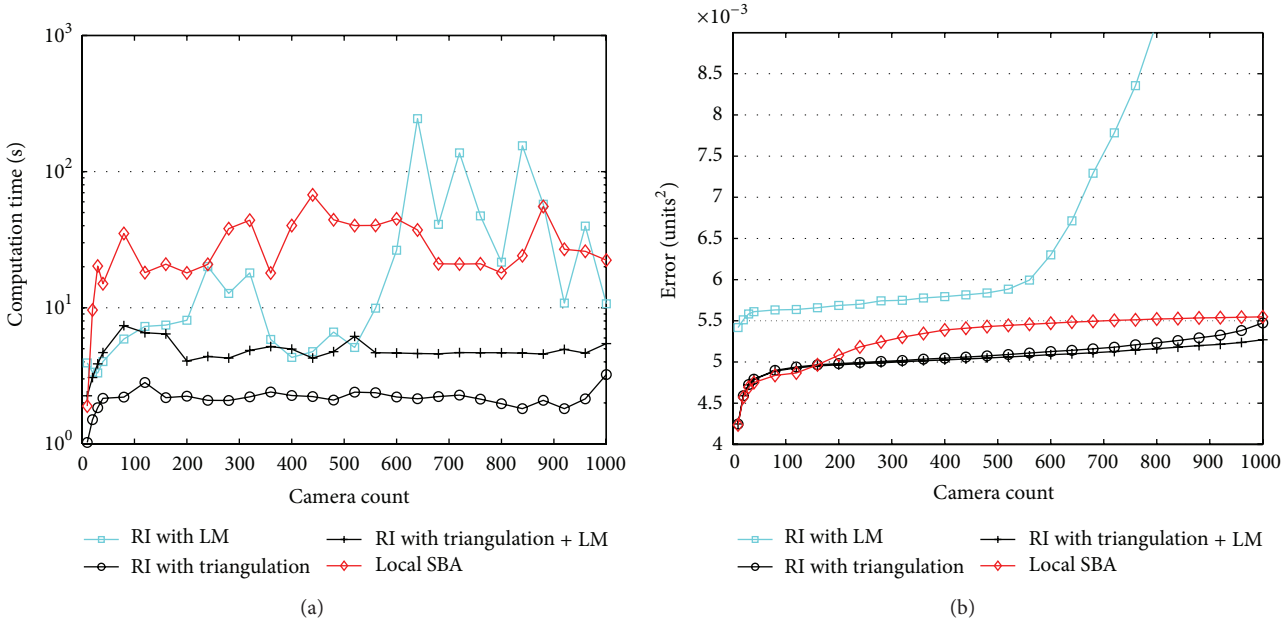


Figure 3: Large simulated data set results. Data is added to the reconstruction progressively and nonlinear refinement is applied at key frames. The refinement time (a) and resulting error (b) are plotted against key frame indices.

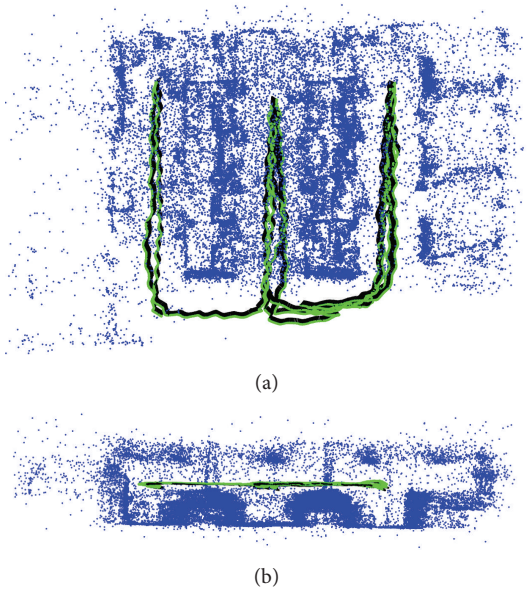
is recovered using the ePnP algorithm [9], followed by LM minimisation applied to the new camera pose parameters only. Outlier rejection is used at this stage to remove gross tracking errors. Key frames are selected at frames where the baseline between key frames surpasses a threshold. At each key frame, new world points are triangulated and the refinement process is applied, with all cameras in-between key frames included in the minimisation.

The data used in this experiment consists of video captured in an office environment. A camera was carried by hand down the three isles of the office multiple times with different orientations. The camera path includes several loops and overlapping sections. Figure 4 shows example images from the data set and Figure 5 shows two views of the final reconstruction. The model contains 7×10^4 points viewed from 6813 cameras, to produce 1.7×10^7 observations.

Figures 6 and 7 show the results for the real data experiment. The full SBA algorithm increases in processing time nearly exponentially. The SBA algorithm was terminated when its run time became excessive, since this is no longer reasonable for practical use. Local SBA requires approximately constant processing time (variations are mostly due to local data conditions) but suffers from large accumulating error over the course of the reconstruction. Large error spikes are also observed, but the algorithm tends to recover from these to maintain a reasonable error profile. The error produced by Local SBA can be reduced by extending the length of the local processing region, at the expense of increased processing time.

Resection-intersection using only LM to refine points is the fastest on average but yields poor accuracy. The accuracy can be improved by tightening the convergence threshold,

Figure 4: Example frames from the real video data set.



(a)

(b)

Figure 5: Real video data set reconstruction. Small blue (dark) dots indicate world points; larger green (light) dots indicate camera centres, with black lines indicating camera viewing direction. The large number of cameras results in the camera path appearing as a line. (a) top view and (b) front view.

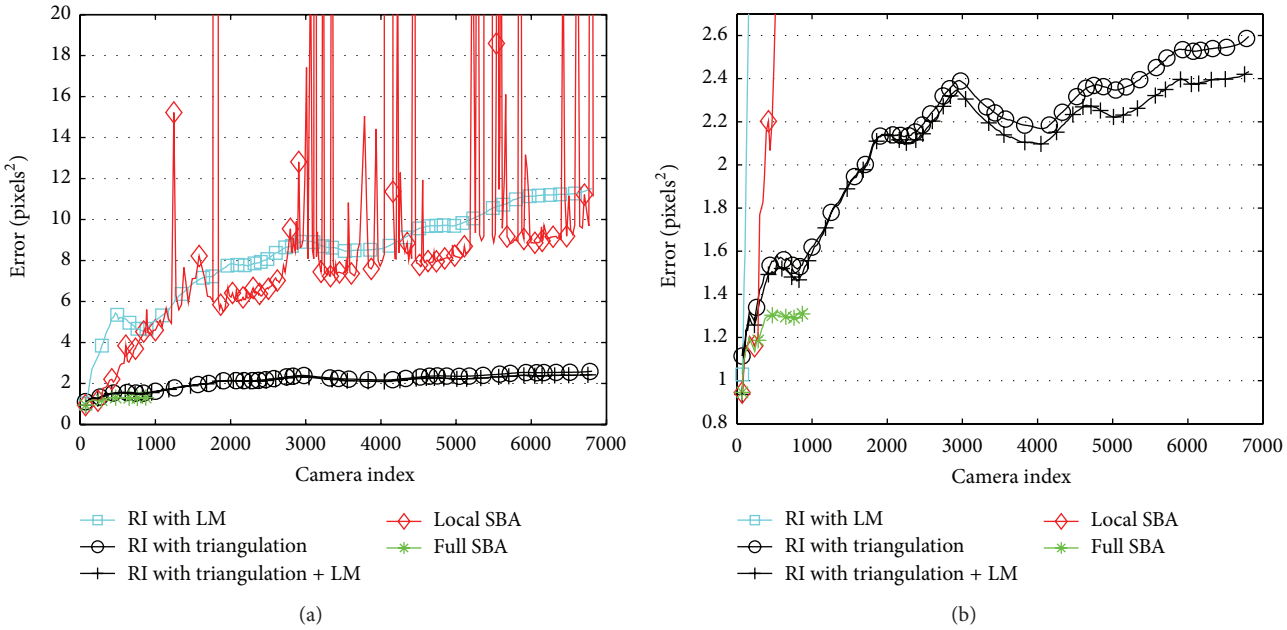


Figure 6: Real video SFM reconstruction results. Data is added to the reconstruction progressively and nonlinear refinement is applied at key frames. The error after refinement is plotted against key frame indices at two different error scales.

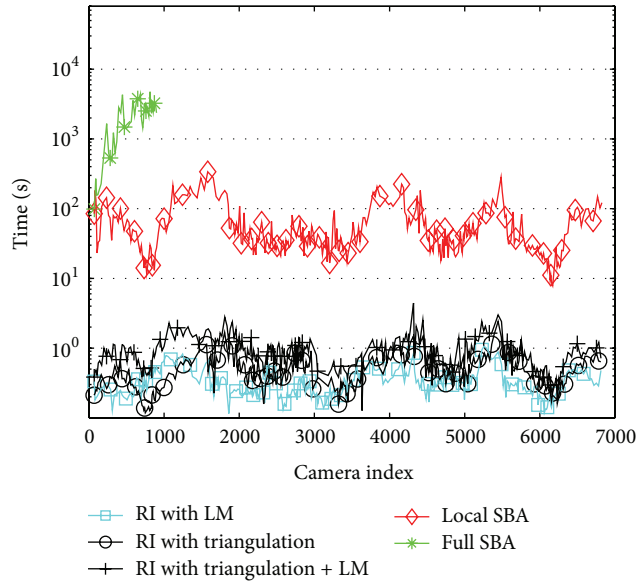


Figure 7: Real video SFM reconstruction results. Data is added to the reconstruction progressively and nonlinear refinement is applied at key frames. The refinement time is plotted against key frame indices.

but only a small improvement can be achieved at the cost of processing time increasing by orders of magnitude.

The proposed linear triangulation addition to the RI algorithm results in a significant improvement in accuracy over the original RI algorithm, while maintaining a low computation time. RI with triangulation and LM yields an error 88% lower than Local SBA at the conclusion of the sequence and exhibits a slower overall increase in error, while reducing the processing time by a factor of 65 on average. Disabling the LM step after triangulation results in a slight increase in the error growth over time (6% higher error over the entire sequence) but reduces computation time further by 45%. The proposed methods yield a reconstruction error approximately 20% higher than full SBA.

5. Conclusions and Future Work

This paper proposes modifications to the resection-intersection method for bundle adjustment that makes the method much more effective and efficient. The proposed methods

are suitable for large scale reconstructions and for use in Structure from Motion systems.

Two modifications to the RI algorithm are proposed. Firstly, adding a linear triangulation step into the intersection stage greatly improves accuracy and convergence speed. This approach is so effective that the nonlinear refinement of world points can be omitted, resulting in further reduction in computation time of 45%, with a very small increase in error. Secondly, a change tracking system is proposed to localise the computational effort so that stable parameters are not subject to redundant processing. Change tracking is particularly effective for progressive reconstructions, where the addition of new data has a limited effect on existing data. Loop closures are handled intrinsically without the need for additional camera network modelling or closure detection.

Simulation experiments show that the proposed method converges much more quickly than the original RI algorithm and a full Newton method, while the final error is only a few percent higher than the full Newton method. Experiments on a real SFM system show that the proposed methods can maintain approximately constant processing time independent of image sequence length, while maintaining good accuracy. Compared to a localised sparse bundle adjustment (SBA) algorithm, the proposed RI with triangulation and nonlinear refinement of points requires 65 times less computation time, while maintaining 4 times lower error, slower error growth over time, and much more stable results. Omitting the nonlinear refinement step in the intersection phase results in processing time 177 times faster than Local SBA, while the error increases only slightly faster than the method using nonlinear refinement (6% increase over 7000 frames). These results show that the nonlinear refinement of world point parameters is of limited value, given accurate camera estimates.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This project was supported by the Australian Research Council Grant no. LP0990135.

References

[1] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment: a modern synthesis," in *Vision Algorithms: Theory and Practice*, vol. 1883 of *Lecture Notes in Compute Science*, pp. 298–372, Springer, 2000.

[2] M. I. A. Lourakis and A. A. Argyros, "SBA: a software package for generic sparse bundle adjustment," *ACM Transactions on Mathematical Software*, vol. 36, no. 1, article 2, 2009.

[3] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, New York, NY, USA, 2nd edition, 2003.

[4] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, "Generic and real-time structure from motion using local bundle adjustment," *Image and Vision Computing*, vol. 27, no. 8, pp. 1178–1193, 2009.

[5] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz, "Multicore bundle adjustment," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '11)*, pp. 3057–3064, June 2011.

[6] H.-Y. Shum, Q. Ke, and Z. Zhang, "Efficient bundle adjustment with virtual key frames: a hierarchical approach to multi-frame structure from motion," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '99)*, vol. 2, pp. 2538–2543, Fort Collins, Colo, USA, June 1999.

[7] G. Sibley, "Relative bundle adjustment," Technical Report 2307/09, University of Oxford, 2009.

[8] G. Sibley, C. Mei, I. Reid, and P. Newman, "Adaptive relative bundle," in *Proceedings of the Robotics: Science and Systems*, pp. 1–8, University of Washington , Seattle, Wash, USA, June 2009.

[9] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: an accurate o(n) solution to the pnp problem," *International Journal of Computer Vision*, vol. 81, no. 2, pp. 155–166, 2009.

[10] V. Pradeep and J. Lim, "Egomotion estimation using assorted features," *International Journal of Computer Vision*, vol. 98, no. 2, pp. 202–216, 2012.