COMPARISON OF BUNDLE ADJUSTMENT FORMULATIONS

Zach Moore, Research Assistant
Daniel Wright, Research Assistant
Dale E. Schinstock, Associate Professor
Chris Lewis, Associate Professor
Kansas State University
3002 Rathbone Hall
Manhattan, KS 66506
zmoore@ksu.edu
dgw2222@ksu.edu
dales@ksu.edu
clewis@ksu.edu

ABSTRACT

Bundle adjustment (BA) is an optimization process refining the estimates of extrinsic camera parameters (position and orientation, or pose) and the three dimensional (3D) positions of features using overlapping images from multiple views. The optimization in BA may be formulated many ways. We compare four alternatives, including the standard formulation, from the perspectives of the region of convergence for errors in the initial estimates of pose and the complexity of the formulation. We consider cost functions formed by the sum of squared elements and minimized using a Levenberg-Marquardt (LM) algorithm. The customary elements of the cost function for BA are found using corresponding features in the images. In the standard formulation (SBA) the cost elements are the difference in the measured image coordinates and the image coordinates found by projecting the current estimate of the 3D position of the features into the camera's focal plane, and the camera poses and the feature positions are varied as explicit parameters in the minimization. In another formulation, implicit BA (IBA), the cost function is the same but the current feature position estimates are implicit parameters not directly varied. They are functions of the current camera pose estimates. This reduces the number of parameters used in the minimization search and therefore reduces the size of the Jacobian matrix, but complicates the calculation of the Jacobian elements. The third formulation, reduced BA (RBA), is the same as the second but uses a simplified approximation of the Jacobian elements. The fourth and last formulation is Alternate Cost BA (ABA). As the name suggests, this formulation is the only one to use a completely different cost function. Again it only uses camera poses as search parameters, but uses a cost function formulated in 3D space rather than image space. The conclusions demonstrate that IBA is inferior to SBA in region of convergence despite the reduced number of parameters and that RBA and ABA are similar to SBA in convergence.

Key Words: Bundle Adjustment, Stereo Image Processing, Photogrammetry

MOTIVATION

The Autonomous Vehicles Lab (AVS) at Kansas State University has developed a capability to collect very high-resolution imagery from small, hand launched unmanned aerial vehicles. Since the planes fly very close to the ground, distortions due to terrain undulation in mosaics formed from these images are severe. To reduce distortion, the images are first projected onto accurate digital elevation models (DEMs) of the terrain and then processed to form ortho-rectified mosaic images. However, sufficiently accurate DEMs do not exist in most cases. Therefore, the terrain needs to be determined from the imagery itself. This process involves a technique called Bundle Adjustment (BA). BA simultaneously refines the terrain model and the camera pose estimates in order to construct an accurate 3 dimensional model of the scene. Although here we concentrate on traditional photogrammetric terrain reconstruction, BA has been applied to 3D modeling of museum artifacts, archeological sites, and crime scene reconstruction to name a few. Two things distinguish our application from some of these others. First our image sequences are of significant length and designed with regular overlap of 40 to 50% between every adjacent image. Second, unlike typical photogrammetric data our initial pose estimates are very poor due to the nature of our navigation equipment. In our application a typical terrain feature for which we would like to estimate its location and elevation is imaged at most 4 times and typically twice, as the plane traverses back and forth across an area. Commercial photogrammetric software has been ineffective in terrain reconstruction from our image and navigation data due to both the size of our data sets, and the poor initial pose estimates. For this reason, we investigated two alternate implementations of the standard BA process (SBA). These two alternate BA formulations are termed Implicit BA (IBA) and Refined BA (RBA). These two alternate methods were formulated to reduce the parameter search space in hopes of reducing the computational burden for large data sets such as ours. It was also hoped that these methods might prove to have better convergence properties as well. This work summarizes our findings with regard to the convergence properties of these alternate formulations.

MINIMIZATION ALGORITHM

All of the formulations of BA discussed in this paper will require the minimization of a scalar cost function, c, that is a sum of squares (an inner product of a cost vector $\vec{C}$), and a function of a parameter vector $\vec{K}$:

$$c = \vec{C}^T \vec{C} = f(\vec{K}) \tag{1}$$

A standard minimization technique that works well for such cost functions and that is commonly used for Bundle Adjustment is the Levenberg-Marquardt (LM) algorithm (Gitl, 1981), an iterative technique that is a hybrid between gradient descent and the Gauss-Newton algorithm. We use a basic implementation of the LM algorithm. At each iteration step this algorithm updates the state vector $\vec{K}$ with a step:

$$d\vec{K} = \left(J^T J + \lambda I\right)^{-1} J^T \vec{c} \tag{2}$$

Here, J is the Jacobian of the cost vector, $\vec{c}$, with respect to the state vector $\vec{K}$.

$$J = \frac{\partial \vec{c}}{\partial \vec{K}} \tag{3}$$

λ is adjusted down or up based on the success or failure of the previously attempted step, respectively (Nielson, 1999). As lambda increases, the algorithm behaves more like gradient descent, which always takes steps that decrease the cost but can be slow close to the minimum. As lambda decreases, the algorithm behaves more like the Gauss-Newton algorithm, which can converge quickly but is not guaranteed to, especially if far away from the minimum.

BA FORMULATIONS

Standard Bundle Adjustment

The goal of BA is to refine camera pose estimates and feature location estimates to optimize the consistency between the feature locations and their appearance in images. Sometimes the internal camera calibration model (e.g. focal length, etc.) is also adjusted, but camera calibration will not be considered in this work. Instead it is assumed that cameras have been calibrated and images have been pre-processed to remove lens distortion. The somewhat standard implementation of BA (Triggs, 2000), SBA, measures scene consistency in each image using the difference between where the features appear and where the estimated scene and camera pose estimate predicts them to appear. This is accomplished by minimizing a cost function defined with these differences. Many of the quantities needed to define this cost function are depicted in Figure 1. To predict where a feature, defined by its location in the world coordinate frame $p_f^W = [x\ y\ z]^T$, appears in a camera's image, the feature is first expressed in the camera's coordinate frame as follows:

$$\vec{p}_f^{\,c} = R_{cw} \cdot (\vec{p}_f^{\,w} - \vec{p}_c^{\,w}) \tag{4}$$

where $\vec{p}_c^{\,w}$ is the position of the camera's center of projection (COP) and R_{cw} is a rotation matrix determined by the camera's pose estimate. A pinhole camera model is used to project the feature expressed in camera coordinates onto the image plane of the camera as follows:

$$\vec{p}_f'^{\,i} = \begin{bmatrix} u' \\ v' \end{bmatrix} = \frac{f}{\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \cdot \vec{p}_f^{\,c}} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \cdot \vec{p}_f^{\,c} \tag{5}$$

where $\vec{p}_f'^{\,i}$ is the prediction defined in image coordinates u and v, and f is the focal length of the camera. The difference between the prediction and the feature's observed location, $\vec{p}_f^{\,i}$, in the image defines two cost elements for each observation of a feature in an image:

$$\begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} = \vec{p}_f^{\,i} - \vec{p}_f'^{\,i} \tag{6}$$

The individual cost elements from every observation of every feature are grouped to form a cost vector:

$$\vec{C} = \begin{bmatrix} \Delta u_1 \ \Delta v_1 \ \Delta u_2 \ \Delta v_2 \cdots \Delta u_q \ \Delta v_q \end{bmatrix}^T \tag{7}$$

where q is the total number of observations of features in images. A scalar, sum of squares, cost function is defined with the inner product of the cost vector as shown in (1). This scalar is a function of the parameter vector, $\vec{K}$, which is adjusted to minimize the cost. These parameters consist of the elements that define $\vec{p}_c^{\,w}$ and R_{cw} for every camera and that define $\vec{p}_f^{\,W}$ for every feature. They are grouped into a single parameter vector $\vec{K}$ as follows:

$$\vec{K} = [\vec{p}_{f1}^{\,W} \ \vec{p}_{f2}^{\,W} \cdots \vec{p}_{fN}^{\,W} \ \vec{p}_{c1}^{\,W} \ \vec{\Omega}_{c1} \ \vec{p}_{c2}^{\,W} \ \vec{\Omega}_{c2} \cdots \vec{p}_{cM}^{\,W} \ \vec{\Omega}_{cM}\,]^T \tag{8}$$

where N is the number if features, M is the number of cameras and $\vec{\Omega}_c = [\omega \ \phi \ \kappa]^T$ is the orientation vector of the camera defining R_{cw} in (1).

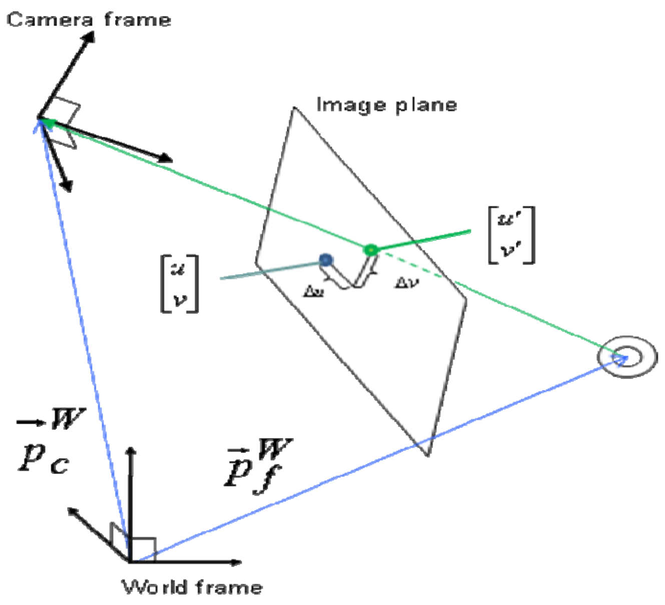

Figure 1. The basic quantities in Bundle Adjustment. A feature estimate is projected into the image plane of a camera and compared to its originally observed position in the image plane.

Implicit Bundle Adjustment

Implicit Bundle Adjustment (IBA) uses the same cost function as SBA. However, the locations of the features are eliminated from the parameter vector, making them an implicit part of the process. This is accomplished by writing their triangulation as a function of the camera locations, camera poses, and their observations in the images ($\vec{p}_c^{\,w}, \vec{\Omega}_c, \vec{p}_f^{\,i}$ respectively).

$$\vec{p}_f^{\,w} = \vec{f}_g(\vec{p}_c^{\,w}, \vec{\Omega}_c, \vec{p}_f^{\,i}) \tag{9}$$

To define this function begin with $\vec{U}_1$ and $\vec{U}_2$, the pointing vectors in world coordinates from each camera's COP through the respective observed feature points u_i and v_i:

$$\vec{U}_1 = R_{cw1}\begin{bmatrix} u_1 \\ v_1 \\ f \end{bmatrix} \quad , \quad \vec{U}_2 = R_{cw2}\begin{bmatrix} u_2 \\ v_2 \\ f \end{bmatrix}. \tag{10}$$

Since the vector $\vec{r}$ spanning the shortest distance between these two camera rays will be perpendicular to both, the direction of $\vec{r}$, $\hat{r}$, if found with a cross product:

$$\hat{r} = \frac{\vec{U}_1 \times \vec{U}_2}{|U_1 \times U_2|}. \tag{11}$$

A vector loop involving the two camera positions, $\vec{U}_1$, $\vec{U}_2$ and $\hat{r}$ is given by

$$\vec{p}_1^{\,w} - \vec{p}_2^{\,w} = d_1\vec{U}_1 + d_3\hat{r} - d_2\vec{U}_2. \tag{12}$$

Here variables d_1 and d_2 are somewhat arbitrary scaling factors for $\vec{U}_1$ and $\vec{U}_2$, but d_3 is the actual length of the vector $\vec{r}$. (12) can be rewritten using matrix format:

$$\bar{p}_1^w - \bar{p}_2^w = \begin{bmatrix} \vec{U}_1 & \hat{r} & -\vec{U}_2 \end{bmatrix} \begin{bmatrix} d_1 \\ d_3 \\ d_2 \end{bmatrix}. \tag{13}$$

Solving (13) for the three unknown scalars gives

$$\begin{bmatrix} d_1 \\ d_3 \\ d_2 \end{bmatrix} = \begin{bmatrix} \vec{U}_1 & \hat{r} & -\vec{U}_2 \end{bmatrix}^{-1} \begin{bmatrix} \bar{p}_1^w - \bar{p}_2^w \end{bmatrix}. \tag{14}$$

Finally, the location of the feature is calculated as follows:

$$\bar{p}_f^w = \bar{p}_1^w + d_1\vec{U}_1 + \tfrac{1}{2} d_3 \hat{r}. \tag{15}$$

Plugging (15) into (4) eliminates the feature locations from the cost function. With this formulation, the parameter space is significantly reduced, at the expense of more complex cost element computation. More detailed analysis of this follows in the Performance Considerations section.

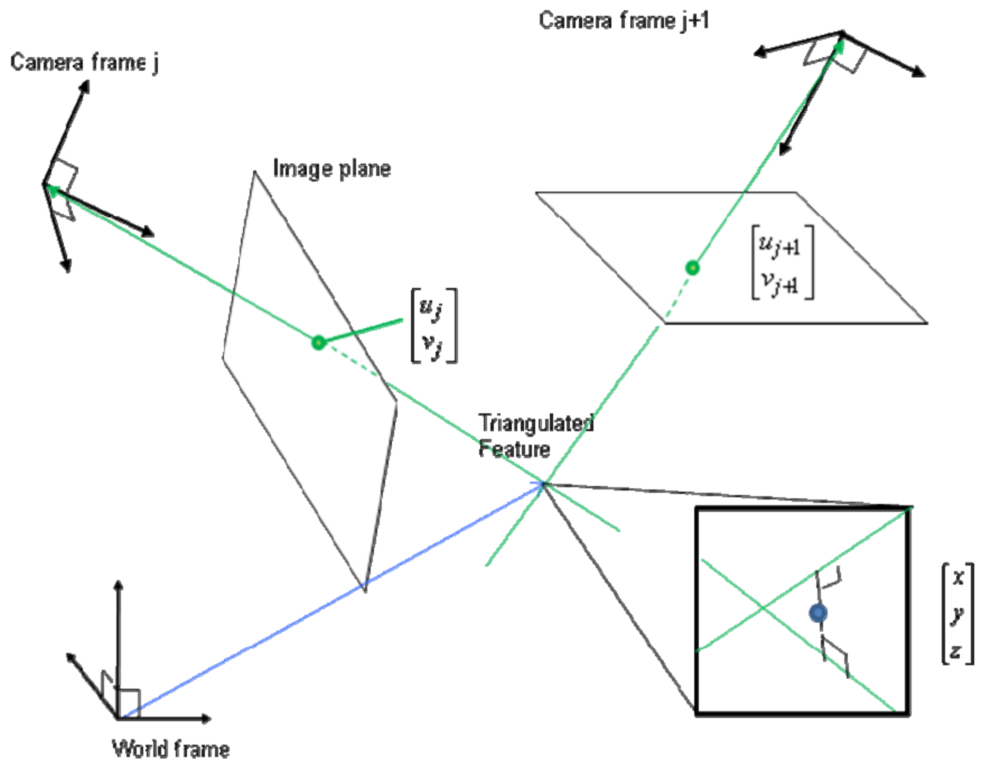
Camera frame j+1
Camera frame j
Image plane
$\begin{bmatrix} u_{j+1} \\ v_{j+1} \end{bmatrix}$
$\begin{bmatrix} u_j \\ v_j \end{bmatrix}$
Triangulated Feature
$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$
World frame

Figure 2. The conceptual process for feature triangulation with two observations of the feature. The zoomed in section shows that the lines are skew in general, so the midpoint is used.

Figure 2 shows the basic process for finding feature positions from triangulation. Since the rays projected from each camera's center of projection through corresponding image points will not in general intersect, the feature location in the world frame is taken to be the midpoint of the shortest line segment between the two rays. In the case of more than two cameras, the location is the average of all such points between pairs of cameras.

Reduced Bundle Adjustment

Reduced Bundle Adjustment (RBA) also eliminates the locations of the features from the parameter vector adjusted in minimization, but retains the computational simplicity of SBA. Instead of writing the feature locations as a function of the camera parameters, they are simply removed from the parameter vector and updated at each iteration by performing triangulation. These updated variables are used in the calculation of the cost function but not accounted for in the parameter vector. This means the Jacobian matrices of SBA and RBA are the same except that there are no entries in RBA for feature locations. While this Jacobian is not technically correct, this type of Bundle Adjustment still gave respectable results in our simulations. It does not however completely converge, but chatters at the end of the minimization process. It has the advantage of an easily computed Jacobian matrix and smaller parameter space.

Alternate Cost Bundle Adjustment

Alternate Cost Bundle Adjustment (ABA) also eliminates the locations of the features from the parameter vector, but does so with the definition of an alternative cost function. In this formulation the cost is calculated in world space using the minimum distance between the rays of two cameras pointing to the same feature. An element of the cost vector is calculated as follows:

$$C_i = (\vec{p}^{\,w}_{ca} - \vec{p}^{\,w}_{cb}) \bullet (\hat{r}) . \tag{16}$$

where $\hat{r}$ is the unit vector orthogonal to both of the rays as defined in equation 11. This is the same as d_3 in (12).

This formulation has the same number of parameters to search as the IBA and RBA, but never explicitly triangulates feature locations. It is however biased towards smaller scenes. That is, it can always reduce the cost by shrinking the scene. Making everything smaller reduces the distance between the camera rays pointing to the same feature thus reducing the cost.

COMPUTATIONAL CONSIDERATIONS

Calculation of the Jacobian and solution of the matrix equation $(J^T J + \lambda I)d\vec{K} = J^T \vec{c}$ for $d\vec{K}$ are the two computationally expensive parts of the minimization. For a scene with N features and M images, there will be $6M + 3N$ total parameters for SBA, corresponding to $[x \quad y \quad z \quad \omega \quad \phi \quad \kappa]$ for each image and $[x \quad y \quad z]$ for each feature. This will be reduced to 6*M* for IBA, RBA and ABA. For the first three methods (SBA, IBA, and RBA) there are $2 \cdot q$ cost vector elements, where $q \le N \cdot M$ is the number of observations of features in the images. There are u and v cost element for each feature and image combination. Generally, for an aerial survey, where areas of overlap are limited to two images, *q* will be on the order of 4*N*. The ABA does not use the image space error, but rather a single world space cost element for each pair of feature observations. So the total number of cost vector elements possible is the number of possible combinations of pairs of observations. Again, for an aerial survey the overlap will be limited to two images generally. Therefore the number of elements will generally be on the order of *N*.

In addition to the size, the complexity of the Jacobian and the cost function are also a concern. Implicitly calculating feature locations in IBA has a major disadvantage in terms of the complexity of calculations. In SBA, feature locations have no influence on each other directly in the equations, nor does the position of one camera directly the influence the position of another. The parameters influence each other through the minimization process. This means that the Jacobian for this formulation is sparse and relatively easy to compute, although it is large. IBA, however, uses triangulation from each camera. This means that a variation in one parameter affects the feature's location in every other camera which significantly complicates the calculations. The function in (9) is actually quite complicated and the additional calculations in the Jacobian are very significant. While the size of the Jacobian is smaller it loses its sparse nature and becomes much more difficult to calculate. Also, adding cameras in this method not only increases Jacobian size, but increases its complexity to compute because the cost in several images must be considered for the variation of the parameters in one image. Although, the computation time is implementation dependent, in our implementation we found that the iterations in SBA are less computationally intensive than IBA.

RBA addresses the issues discussed in the previous paragraph at the cost of using an approximate Jacobian. As explained before, it still uses triangulation to calculate the feature locations, but does not take this into account when computing the Jacobian. It simply assumes that changing the camera parameters will not affect the feature location or cost in any other camera. The Jacobian then becomes exactly the SBA Jacobian without the entries for feature locations, making it again sparse and easier to compute. Also, the complexity of the Jacobian calculation no longer depends on the number of cameras as in IBA.

ABA results in the lowest computational complexity. This implementation has the same smaller parameter search space, and an even smaller Jacobian. In addition, it does not triangulate feature locations. However, the cost function is defined in the world space, resulting in a biased estimator because reducing the size of the entire scene is rewarded in cost function.

Table 1. Summary of Computational Considerations

	Size of Jacobian	Search space	Computational complexity
SBA	(order of 4*N*) x (6*M*+3*N*)	cameras and features	Med
IBA	(order of 4*N*) x (6*M*)	cameras only	High
RBA	(order of 4*N*) x (6*M*)	cameras only	Low
ABA	(order of *N*) x (6*M*)	cameras only	Lowest

REGION OF CONVERGENCE

One of the motivations for our investigation of alternative formulations of BA was lack of convergence with commercial packages implementing standard BA and our poor initial pose estimates from inexpensive inertial navigation systems. Here we will discuss experiments to test what we call the “region of convergence” for each method. This is a measure of the ability of the algorithm to converge to the “correct” solution with increasing errors in the initial estimates of camera pose. To test the region of convergence we used artificially generated scenes of two cameras, multiple sets of features, and many randomly generated initial errors in pose. We only considered initial error in the orientation variables for one of the cameras, although all parameters were allowed to change. This error is a 3 dimensional vector. We generated 100 random unit error vectors to be used for each run. For each run, we normalized these 100 vectors to specific values between 0.1 and 0.5 radians. The same sets of vectors were used for all formulations. Since the original scene was known, convergence was determined by subtracting the final ground point locations found from bundle adjustment from the original ones, summing these errors and measuring it against a small threshold. However, since the algorithms may converge on any scaled or rotated version of the original scene, the final points needed to be transformed appropriately before the difference was taken. The starting camera positions were also known, so the correct rotation can be obtained by using the opposite of the rotation that one camera experienced during the minimization. The scale can likewise be found by using the ratio of the distances between the cameras before and after the adjustment. With no noise in the image point measurement, the error should be virtually zero if the correct solution was found. However, in the case of the RBA algorithm the incorrect Jacobian does not converge on the perfect solution, but instead chatters about this solution.

Early tests comparing ABA to commercial packages led us to believe that alternative forms of bundle adjustment may actually have a greater region of convergence than SBA. However, this hypothesis was not confirmed. IBA was the poorest performing of all. However, the RBA performed almost as well as SBA (the best of all) even with its approximate Jacobian. However, it does not completely converge. Results are shown in Figures 3 and 4.

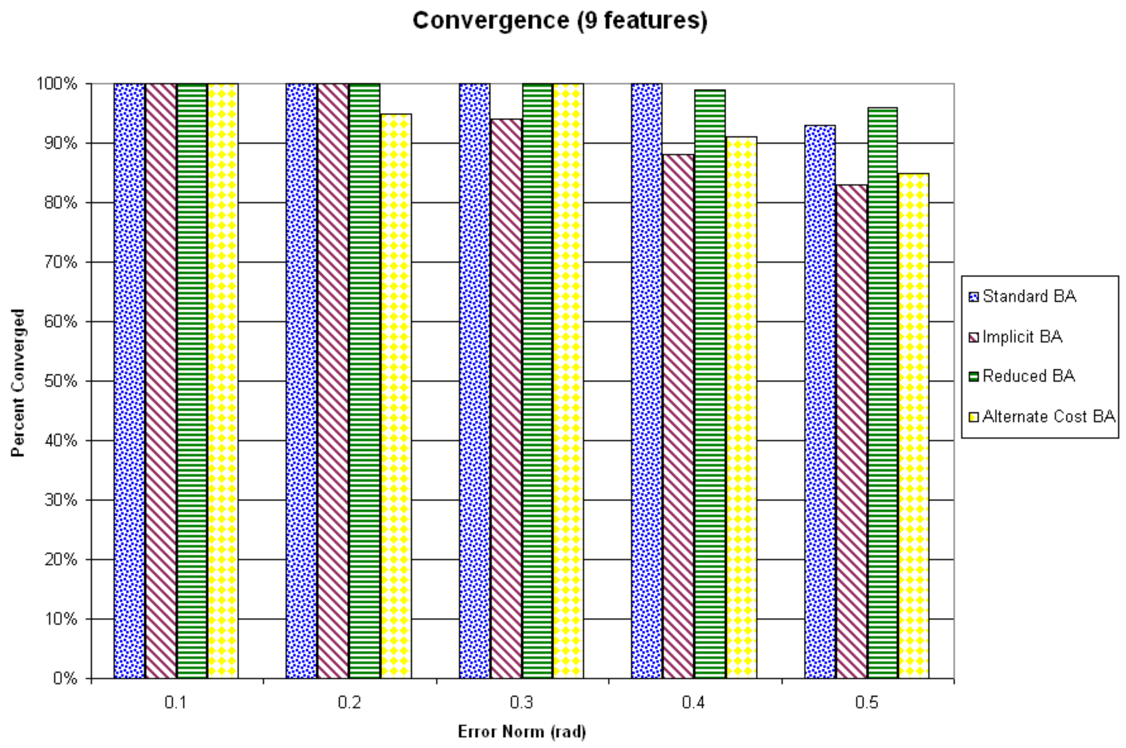

Figure 3. Convergence graph for 9 features in a grid.

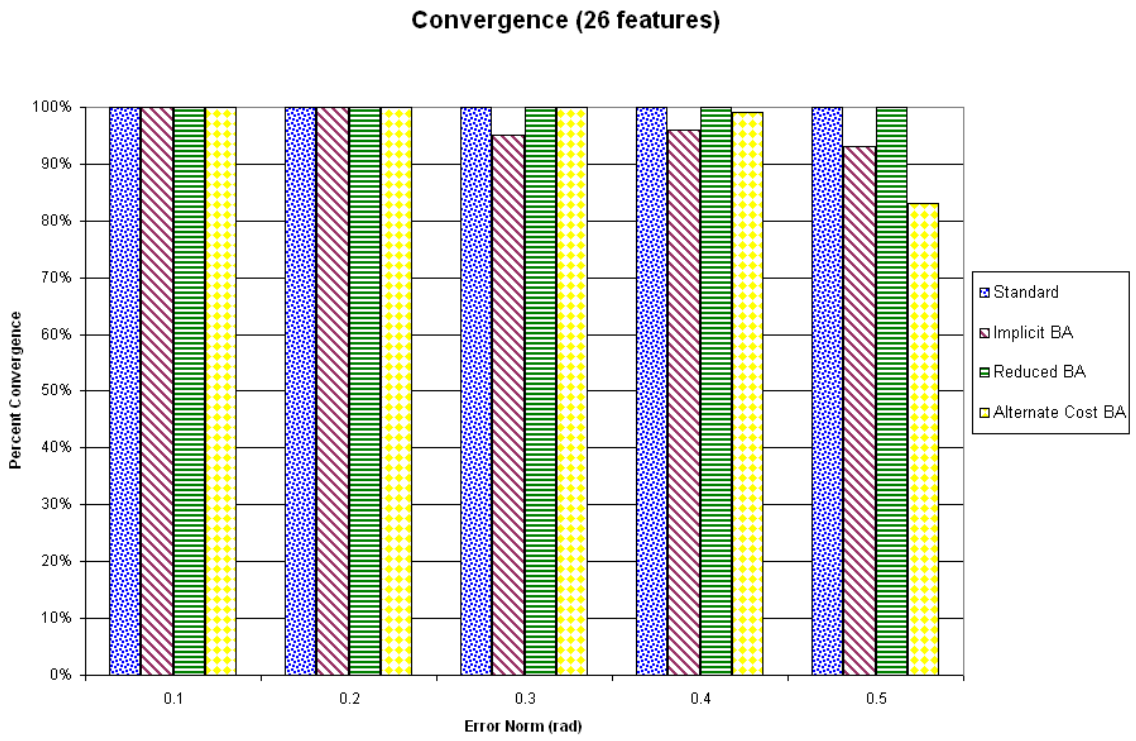

Figure 4. Convergence graph for 26 features in a pyramid.

CONCLUSION

Although it was hoped that a larger region of convergence could be achieved with alternative formulations of the BA, this was not confirmed in our tests. It was hoped that by reducing the search space the minimization would have a tendency to find the correct solution even with large initial errors. Given the complexity of IBA the authors see no benefit to this formulation at all. SBA appears to be the best overall formulation given that it converges to the correct solution, does not chatter near the minimum as does RBA, is of medium complexity and is not biased as is ABA. Perhaps RBA or ABA would have some use in real time applications where computation is a very significant consideration and complete convergence could not be expected.

REFERENCES

Gilt, P.E., W. Murray, M.H. Wright, 1981. *Practical Optimization*, Academic Press.

Nielson, H.B., 1999. Damping parameter in Marquardt's method, *Technical Report IMM -REP-1999-05*, Department of Mathematical Modeling, DTU.

Triggs, B., P.F. McLauchlan, R.I. Hartley and A. Fitzgibbon, 2000. Bundle adjustment -- A modern synthesis, In *Vision Algorithms: Theory and Practice*, volume 1883 of Lecture Notes in Computer Science. Pages 298-372. Springer-Verlag.