Plane-based Projective Reconstruction

Robert Kaucic[1], Nicolas Dano[1] and Richard Hartley[2]
[1]GE Corporate Research and Development, Schenectady, NY
[2]Australia National University, Canberra, Australia

Abstract

A linear method for computing a projective reconstruction from a large number of images is presented and then evaluated. The method uses planar homographies between views to linearize the resecting of the cameras. Constraints based on the fundamental matrix, trifocal tensor or quadrifocal tensor are used to derive relationships between the position vectors of all the cameras at once. The resulting set of equations are solved using a SVD. The algorithm is computationally efficient as it is linear in the number of matched points used. A key feature of the algorithm is that all of the images are processed simultaneously, as in the Sturm-Triggs factorization method, but it differs in not requiring that all points be visible in all views. An additional advantage is that it works with any mixture of line and point correspondences through the constraints these impose on the multilinear tensors. Experiments on both synthetic and real data confirm the method's utility.

1 Introduction

It is commonly accepted that bundle adjustment is the Gold standard for projective reconstruction. However, when dealing with a large number of views, bundle adjustment methods face a dilemma. The computation of all cameras and all 3D points in a single adjustment neccessitates a good initialization. On the other hand, processing the views sequentially and bundle adjusting after each view is added becomes extremely expensive computationally as the number of images gets large. The factorization method [ST96] provides a good balance between the two extremes, however is limited by its restriction that all 3D points must be visible in all views. The planar-based projective reconstruction method presented here has no such restriction. The method utilizes the fact that planar homographies between views enable the linearization of the computation of the camera matrices. Further, the method is computationally efficient and when used as an initialization for bundle adjustment provides near optimal results.

The algorithm described in this paper is derived from a brief outline given in [HZ00]. No implementation details or analysis of results were given there.

2 Plane-based reconstruction

It is known that if four points visible in some number of images are known to be coplanar, then the computation of the multifocal tensors relating the image points becomes significantly more simple. For instance, the fundamental matrix for two views may be computed from two additional point correspondences. Using knowledge of planarity is the theme of [ST98], and is at the base of the plane-plus-parallax approach to vision geometry ([KAH94, IRP97, Saw94]). A major advantage of using knowledge of planity is that a tensor satisfying all its constraints may be computed using a linear algorithm. This is a particular advantage in the computation of the trifocal tensor or quadrifocal tensor, which must satisfy many constraints (8 and 51 respectively) imposed by the geometry.

In this paper, it is shown that the linear methods of computing the multi-view tensors may be extended to estimate all the camera matrices simultaneously, using a new linear algorithm. This algorithm is very rapid, in fact linear in the number of matched points known. A similiar approach is given in [RC01] where the cameras and projective structure are computed simultaneously.

The condition that four of the image correspondences are derived from coplanar points is equivalent to knowing the homographies between the images induced by a plane in space, since a homography may be computed from the four points. It is only the homographies that are important in the following approach, for that reason, we will henceforth suppose that image-to-image homographies are known between images in a sequence.

If $\mathtt{H}^i$ is the plane-induced homography that maps a point in the first image to its matching point in the i-th image, then the set of camera matrices can be assumed to have the form $\mathtt{P}^i = [\mathtt{H}^i|\mathbf{t}^i]$, where the $\mathtt{H}^i$ are known, but the final columns $\mathbf{t}^i$ are not. We may assume that $\mathtt{P}^1 = [\mathtt{I} \mid \mathbf{0}]$, so that $\mathbf{t}^1 = \mathbf{0}$. The set of all remaining $\mathbf{t}^i$ have $3m-4$ degrees of freedom, where m is the number of views, since the $\mathbf{t}^i$ are defined only up to a common scale. Now assume that several point or line correspondences across two or more views are known (three views are required for lines). In order to provide useful information, these correspondences must derive from 3D points or lines that do not lie in the reference plane (used to compute the $\mathtt{H}^i$). Each point cor-

respondence across two views leads to a linear equation in the entries of the fundamental matrix. Similarly, correspondences of points or lines across three or four views lead to linear equations in the entries of the trifocal or quadrifocal tensor.

The key point (as explained in section 3) is that we may express the entries of the fundamental matrix (or trifocal or quadrifocal tensor) linearly in the entries of the vectors $\mathbf{t}^i$. Therefore each linear relation induced by a point or line correspondence may be related back to a linear relationship in terms of the entries of the $\mathbf{t}^i$. Thus, for example, a correspondence across views i, j and k gives rise to a set of linear equations in the entries of the three vectors $\mathbf{t}^i$, $\mathbf{t}^j$ and $\mathbf{t}^k$. A set of correspondences across many views can be broken down into correspondences across sets of consecutive views. Thus, for example, a single point correspondence across $m > 4$ views will give a set of equations of the form

$$\begin{bmatrix} \mathtt{X} & \mathtt{X} & \mathtt{X} & \mathtt{X} & & & & & & & & \\ & \mathtt{X} & \mathtt{X} & \mathtt{X} & \mathtt{X} & & & & & & & \\ & & \mathtt{X} & \mathtt{X} & \mathtt{X} & \mathtt{X} & & & & & & \\ & & & & & \ddots & & & & & & \\ & & & & & & \mathtt{X} & \mathtt{X} & \mathtt{X} & \mathtt{X} & & \\ \mathtt{X} & & & & & & & \mathtt{X} & \mathtt{X} & \mathtt{X} & & \\ \mathtt{X} & \mathtt{X} & & & & & & & \mathtt{X} & \mathtt{X} & & \\ \mathtt{X} & \mathtt{X} & \mathtt{X} & & & & & & & \mathtt{X} & & \end{bmatrix} \tag{1}$$

where each row represents a set of equations derived from a quadrifocal tensor relationship. Each $\mathbf{X}$ represents a block with 3 columns corresponding to one of the vectors $\mathbf{t}^i$. In the table above, we choose to wrap the equations around from the last to the first view to add greater rigidity. Otherwise, the values of the $\mathbf{t}^i$ can drift from the first to the last view. Other schemes for selecting groups of views are possible, and it is not necessary to restrict to consecutive views.

Linear relations may be generated between any subset of sufficiently many images (2, 3 or 4 depending on which tensor is used to generate the equations). One must trade off the added stability of the solution against the added computational cost of adding more equations. A mixture of bifocal, trifocal and quadrifocal constraints may be used in generating the set of all equations, and it is not necessary that all points be visible in all views. In addition, equations may be derived from line or line-point correspondences, through trifocal or quadrifocal constraints ([Har97].

3 Details of equation formation

Consider a pair of camera matrices, denoted for convenience $\mathtt{P} = [\mathtt{A}|\mathbf{a}]$ and $\mathtt{P}' = [\mathtt{B}|\mathbf{b}]$ and let the corresponding fundamental matrix be $\mathtt{F}$. A point correspondence $\mathbf{x}' \leftrightarrow \mathbf{x}$ between the two views leads to a well-known equation $\mathbf{x}'^{\top}\mathtt{F}\mathbf{x} = 0$ involving the entries of $\mathtt{F}$, and linear in those entries. For several (say n) point correspondences between the same two views, the set of all equations may be written as

$$\mathtt{S}\mathbf{f} = 0 \ , \tag{2}$$

where $\mathtt{S}$ is an $n \times 9$ matrix, and $\mathbf{f}$ is the vector containing the entries of $\mathtt{F}$.

Now, if left-hand 3×3 blocks $\mathtt{A}$ and $\mathtt{B}$ of the two camera matrices are known, then the entries of $\mathtt{F}$ may be written as linear expressions in terms of the last columns $\mathbf{a}$ and $\mathbf{b}$ of the two camera matrices. To be more precise, the entries of $\mathtt{F}$ are written in terms of the camera matrices as follows ([HZ00, Har95, Hey98]) :

$$F_{ji} = (-1)^{i+j} \begin{vmatrix} \sim\mathbf{P}^{(i)} \\ \sim\mathbf{P}'^{(j)} \end{vmatrix} . \tag{3}$$

where $\sim\mathtt{P}^{(i)}$ represents the matrix $\mathtt{P}$ with the i-th row removed. Vertical bars represent the determinant. For example,

$$F_{11} = \begin{vmatrix} \mathbf{A}^2 & a^2 \\ \mathbf{A}^3 & a^3 \\ \mathbf{B}^2 & b^2 \\ \mathbf{B}^3 & b^3 \end{vmatrix} .$$

Expanding this determinant down the last column gives

$$F_{11} = -a^2 \begin{vmatrix} \mathbf{A}^3 \\ \mathbf{B}^2 \\ \mathbf{B}^3 \end{vmatrix} + a^3 \begin{vmatrix} \mathbf{A}^2 \\ \mathbf{B}^2 \\ \mathbf{B}^3 \end{vmatrix} - b^2 \begin{vmatrix} \mathbf{A}^2 \\ \mathbf{A}^3 \\ \mathbf{B}^3 \end{vmatrix} + b^3 \begin{vmatrix} \mathbf{A}^2 \\ \mathbf{A}^3 \\ \mathbf{B}^2 \end{vmatrix} .$$

Thus, F_{11} is a linear expression in terms of the unknown entries a^i and b^i. The other entries of the fundamental matrix may be similarly expressed, leading to a linear relationship

$$\mathbf{f} = \mathtt{T}\mathbf{t} \tag{4}$$

where $\mathbf{t}$ is the 6-vector consisting of the elements of $\mathbf{a}$ and $\mathbf{b}$, and $\mathtt{T}$ is a 9×6 matrix representing the linear relationship. Combining this equation with (2) yields a set of equations of the form $\mathtt{ST}\mathbf{t} = \mathbf{0}$, where $\mathtt{ST}$ is a $n \times 6$ matrix. Note that $\mathtt{S}$ is determined by the set of all point correspondences between the pair of images, and $\mathtt{T}$ is computed from the known 3×3 blocks $\mathtt{A}$ and $\mathtt{B}$ of the two camera matrices.

Given several (say m) images, and corresponding points $\mathbf{x}^i_j$ in the set of images, one may form such a set of equations for each pair of images. All such equations involve the unknown last columns of the camera matrices, and may be combined together to form a large set of equations of the form $\mathtt{E}\mathbf{t} = 0$, where now $\mathbf{t}$ is the $3m$-vector consisting of the final columns of all camera matrices.

It will not normally be necessary to include all pairs of views when forming such equations. If only pairs of consecutive images are used, then a sparse equation set results,

where E has the following form:

$$\begin{bmatrix} \mathtt{X} & \mathtt{X} & & & & \\ & \mathtt{X} & \mathtt{X} & & & \\ & & \mathtt{X} & \mathtt{X} & & \\ & & & \ddots & & \\ & & & & \mathtt{X} & \mathtt{X} \\ \mathtt{X} & & & & & \mathtt{X} \end{bmatrix} \tag{5}$$

An entry X represents a non-zero block of coefficients, and empty slots represent blocks of zeros.

It was observed in [HZ00] that it is not sufficient to use only consecutive pairs of views[1] ; in fact it is advisable to include also alternate pairs of views, resulting in a further set of equations with the block-form as follows:

$$\begin{bmatrix} \mathtt{X} & & \mathtt{X} & & & \\ & \mathtt{X} & & \mathtt{X} & & \\ & & \mathtt{X} & & \mathtt{X} & \\ & & & \ddots & & \\ \mathtt{X} & & & & \mathtt{X} & \\ & \mathtt{X} & & & & \mathtt{X} \end{bmatrix}.$$

In solving the total set of equations, one may (and should) assume that the first camera matrix has zeros in the last column. Thus, the first three entries of $\mathbf{t}$ are zero, and the equation matrix is reduced in dimension by deleting its first three columns, leaving a set of equations in $3(m-1)$ variables.

4 Equations derived from the trifocal and quadrifocal tensors

The above discussion gave the general outline of how to derive equations based on the fundamental matrix. It is also possible to use three and four-view relations based on the trifocal and quadrifocal tensors.

Trifocal constraints. Given a point correspondence across three views $\mathbf{x} \leftrightarrow \mathbf{x}' \leftrightarrow \mathbf{x}''$, a set of relationships of the form

$$\mathbf{x}^i \mathbf{x}'^r \epsilon_{jru} \mathbf{x}''^s \epsilon_{ksv} \mathcal{T}_i^{jk} = 0_{uv}$$

exists ([HZ00, Har97]). Each choice of the free indices u and v gives a single linear relationship in the entries of $\mathcal{T}_i^{jk}$, a total of 9 equations, but only 4 of them are linearly independent. Given several point correspondences across the three views, a set of equations of the form $\mathtt{E}\mathbf{t} = 0$. Here $\mathbf{t}$ is a 27-vector consisting of the entries of the trifocal tensor, and E is a $4n \times 27$ matrix (or $9n \times 27$ if all 9 equations are included).

[1] However experiments with a tableau of the form (5) seem to give good results.

As with the fundamental matrix, the entries of the trifocal tensor may be written in terms of the last columns of the camera matrices. The analogous formula to (3) in the trifocal case is

$$\mathcal{T}_i^{qr} = (-1)^{i+1} \begin{vmatrix} \sim\mathbf{P}^{(i)} \\ \mathbf{P}'^q \\ \mathbf{P}''^r \end{vmatrix}. \tag{6}$$

where $\mathbf{P}'^i$ means the i-th row of $\mathtt{P}'$ and as before $\sim\mathtt{P}^{(i)}$ represents the matrix P with the i-th row removed. Expanding this formula in cofactors down the last column as before results in a linear expression for the entries of $\mathcal{T}_i^{jk}$ in terms of the final columns $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ of the three camera matrices.

Combining the sets of equations from successive triples of views into a single equation set results in an equation matrix with the block-form

$$\begin{bmatrix} \mathtt{X} & \mathtt{X} & \mathtt{X} & & & & & & \\ & \mathtt{X} & \mathtt{X} & \mathtt{X} & & & & & \\ & & \mathtt{X} & \mathtt{X} & \mathtt{X} & & & & \\ & & & & \ddots & & & & \\ & & & & & & \mathtt{X} & \mathtt{X} & \mathtt{X} \\ \mathtt{X} & & & & & & & \mathtt{X} & \mathtt{X} \\ \mathtt{X} & \mathtt{X} & & & & & & & \mathtt{X} \end{bmatrix} \tag{7}$$

Quadrifocal constraints. The method works just as well with quadrifocal constraints. The basic quadrifocal constraint involving a point correspondence across 4 views is

$$x^i \epsilon_{ipt} x'^j \epsilon_{jqu} x''^k \epsilon_{krv} x'''^l \epsilon_{lsw} Q^{pqrs} = 0_{tuvw} \ .$$

Each choice of free index gives a single equation, but the full set of equations has rank 16.

The formula for the quadrifocal tensor is

$$Q^{pqrs} = \begin{vmatrix} \mathtt{P}^p \\ \mathtt{P}'^q \\ \mathtt{P}''^r \\ \mathtt{P}'''^s \end{vmatrix} \tag{8}$$

Expanding this determinant down the last column gives a linear formula for each entry of Q in terms of the last columns of the four camera matrices. Consecutive quadruples of views give a sparse set of equations of the form (1).

5 Efficiency considerations

The full set of equations created by the algorithm here described may be very large. For instance, consider a set of n points seen in m views, all points being visible in all views. In the trifocal case, a total of 4 (or 9 non-independent) equations are generated for each point in each triple of images, a total of $4mn$ equations in total. The complete set of equations has dimension $4mn \times 3(m-1)$. In the quadrifocal

case this becomes $16mn \times 3(m-1)$. Typically we run this algorithm with 500 points (comfortably found using the Kanade-Lucas point tracker [LK81]) and 20 or more views. The total set of equations is of size 40000×57 in the case of the trifocal algorithm or 160000×57 in the quadrifocal case – a relatively large set of equations. Typically, this equation set is solved in a least-squares sense using the Singular Value Decomposition (SVD), where the solution is the singular vector corresponding to the smallest singular value. A full-scale computation of the SVD, $\mathtt{E} = \mathtt{UDV}^\top$ in which both $\mathtt{U}$ and $\mathtt{V}$ are computed will be very expensive. According to formulae quoted by Golub ([GVL83]) a full SVD of an $m \times n$ matrix requires $4m^2n + 8mn^2 + 9n^3$ flops. However, if only the matrices $\mathtt{V}$ and $\mathtt{D}$ are required, then only $4mn^2 + 8n^3$ flops are required. This is the present case, since the solution is the last column of $\mathtt{V}$. For a 160000×57 matrix, the full SVD would require 5841 Gflops, whereas only 2.08 Gflops are used if $\mathtt{U}$ is not accumulated. Clearly, it is essential to use an implementation of the SVD in which one has the option not to accumulate $\mathtt{U}$.

Further improvements. One may take further advantage of the sparseness of the system. For concreteness, consider the quadrifocal case. The total set of equations to be solved has the form given in (1). One achieves greater efficiency by carrying out orthogonal row reductions on the sets of equations derived from a single quadruplet, represented by a single row of blocks in (1). Thus, the set of equations derived from a single quadruplet of views has dimension $16n \times 12$, each point correspondence contributing 16 equations in the 12 (last columns of the camera matrices) related to the four views in question. Clearly, this set of equations can not have rank greater than 12, the number of rows. Consequently, it may be reduced to a 12×12 set of equations. In order to obtain the same numeric result, this should be done by orthogonal row operations.

By orthogonal row operations is meant multiplying the equation matrix $\mathtt{E}$ on the left by an orthogonal matrix $\mathtt{U}$ so that $\mathtt{UE}$ has block form

$$\mathtt{UE} = \begin{bmatrix} \mathtt{E}' \\ 0 \end{bmatrix}$$

where $\mathtt{E}'$ is a square matrix. The matrix $\mathtt{E}'$ is then used instead of $\mathtt{E}$. Orthogonal row reduction is most efficiently carried out using Householder matrices (as in the first step of an SVD algorithm [GVL83]). It may also be accomplished (though with some inefficiency) by using SVD, without accumulation of $\mathtt{U}$. If $\mathtt{E} = \mathtt{UDV}^\top$, then $\mathtt{E}' = \mathtt{DV}^\top$ is the required orthogonally row-reduced matrix.

The complete algorithm is then as follows (described below for trifocal tensor implementation, but also valid for fundametal-matrix and quadrifocal tensor cases):

Objective Generate set of equations from n views of m points.

Algorithm

1. For groups of 3 views, use all points visible in all three views to generate a $4n \times 27$ set of linear equations $S\mathbf{t} = \mathbf{0}$.
2. Generate the 27×9 matrix $\mathtt{T}$ expressing $\mathbf{t}$ in terms of the vector of last columns of the three views : $\mathbf{t} = \mathtt{T}\mathbf{a}$.
3. Form the $4n \times 9$ matrix $\mathtt{E} = \mathtt{ST}$.
4. Orthogonally row-reduce $\mathtt{E}$ to get $\mathtt{E}'$.
5. From the equation matrices $\mathtt{E}'$ formed from all groups of 3 views used, generate a complete set of equations of dimension $9m \times 3(m-1)$, as in (7).
6. Solve this set of equations (using SVD) to find the solution, namely the last columns of all the camera matrices.

Total complexity Using this method, the total complexity of all the SVD computations may be computed as follows. For the purposes of this computation, we suppose that there are m views of n points and that all points are visible in all views. This is of course not necessary for the algorithm to work. It is furthermore assumed that m groups of 2, 3 or 4 views are used, as for instance in (1). Finally, for simplicity in computing the computational cost, it is assumed that the orthogonal row reduction is done using SVD. Denote by $\mathrm{SVD}(a,b)$ the cost of carrying out a singular value decomposition of a matrix with a rows and b columns, namely $\mathrm{SVD}(a,b) = 4ab^2 + 8b^3$.

The computation cost involved in computing the SVDs (the major algorithmic cost) then consists of

1. For each of the m groups of views, multiplication $\mathtt{E} = \mathtt{ST}$ to form the set of equations for this group.

   bifocal case $n \times 9 \times 6$ multiply/adds.

   trifocal case $4n \times 27 \times 9$ multiply/adds.

   quadrifocal case $16n \times 81 \times 12$ multiply/adds.

2. For each of m groups of views, orthogonal row reduction of a set of equations derived from n point matches.

   bifocal case n equations in $r = 6$ unknowns.

   trifocal case $4n$ equations in $r = 9$ unknowns.

   quadrifocal case $16n$ equations in $r = 12$ unknowns.

3. SVD of the complete set of equations of dimension $mr \times 3(m-1)$ equations to solve for the last columns of the camera matrices.

This gives a total complexity of

bifocal case $54mn + m \times \text{SVD}(n, 6) + \text{SVD}(6m, 3(m-1))$.

trifocal case $972mn + m \times \text{SVD}(4n, 9) + \text{SVD}(9m, 3(m-1))$.

quadrifocal case $15522mn + m \times \text{SVD}(16n, 12) + \text{SVD}(12m, 3(m-1))$.

For the case of 500 points in 20 views, this gives 5.0 Mflops, 26.6 Mflops and 252.5 Mflops for the bifocal, trifocal and quadrifocal cases respectively.

Comparison with factorization algorithm. In the factorization algorithm, one is faced with the task of carrying out a full SVD (including accumulation of `U` and `V`) for a $3m \times n$ matrix of image measurements. This requires approximately $4(3m)^2n + 8(3m)n^2 + 9n^3$ flops. In the case of 500 points, 20 views, this gives 76.3 Mflops.

Note that the complexity of the factorization algorithm grows as the square of the number of points, but the planar-based algorithm grows linearly in the number of points. The result is that the factorization algorithm is faster than the planar-based methods for small numbers of points (less than about 200), but is slower for large numbers of points.

6 Computation of the homographies

Correlation and point-based methods were implemented with good results. Both methods have their proponents, and without attempting to make a definitive pronouncement we make the following remarks. The correlation methods work well, but are more difficult to implement in cases where a well-defined plane is not visible in the image, or there is more than one significant plane. It is possible that an approximate homography may exist between images, but that this homography may be one induced by an actual plane; note that not all image homographies are induced by a plane ([HZ00], chapter 12). In addition, the point-based method has the advantage that planar points may be found by applying RANSAC to the same set of points subsequently used to compute the reconstruction, thereby avoiding significant additional computational burden (RANSAC is fast).

7 Other reconstruction methods used for comparison

The new plane-based reconstruction method has been compared with several other popular reconstruction methods.

Factorization method. The factorization method used is based on the method of [ST96, Tri96] which is similar in style to the Kanade-Tomasi factorization method ([TK92]), but applied to projective cameras. The method is not an exact method, since it relies on an estimate of the "projective depths" of the points, which will not in general be exact. In the original paper, Sturm and Triggs propose an initialization scheme for the weights based on the fundamental matrix, but in the implementation we used, the weights were set initially all to 1, and reestimated by reprojection. This method has been observed by other authors to give good results.

For the factorization algorithm to be applied, all points used must be visible in all views.

Incremental reconstruction The algorithm is also compared with an incremental method based on two and three-view reconstruction. The method is as follows (described for the trifocal-tensor implementation):

1. Choose three views containing sufficiently many common matched points, and carry out projective reconstruction based on the trifocal tensor ([Har97].
2. Compute the 3D locations of all points visible in at least two of the views. At the end of this step, some of the 3D points have been computed (said to be "reconstructed") and some are not, since they are visible in only one of the three initial views.
3. Select the one of the remaining views that see the largest number of reconstructed points, and compute the camera matrix for this view using the DLT algorithm ([Sut63, HZ00]).
4. Reconstruct all additional points that are now visible in two of the views. Return to the previous step if any camera matrices remain to be computed. Steps 2 and 3 are repeated until completion.

A similar method was tried, in which the initial projective reconstruction was carried out using just two views, using the fundamental matrix ([HGC92]).

There is a trade-off when using this incremental method in choosing views for the initial reconstruction. If the views are close together in sequence, then there may be many matched points, but the base-line may be short, resulting in inaccurate initial reconstruction, and ultimate failure. On the other hand, if the base-line is wider, then the number of common points may be small. In the examples we used, this was not a problem, since it was possible to find many points visible across all views, and so we chose the widest possible base-line for the initial reconstruction – namely first, last and middle views.

Obviously variations on this method are possible, such as an adjustment to the reconstruction after each new view is computed. However, the unavoidable problem with such incremental methods is that they are strongly dependent on the initial reconstruction, which depends on a small subset of views.

8 Results

The algorithm was tested on both real and synthetic data. The experiments on synthetic data enabled a quantitative comparison of the various reconstruction methods in typical settings. Towards this end, rather than using purely synthetic data, real sequences served as the basis for the synthetic experiments (e.g. the boat sequence in figure 1). A 3D projective reconstruction was computed using the linear resection algorithm followed by a complete bundle adjustment. "Ground truth" 2D data was taken to be the projection of the 3D points into each of the views.

8.1 Synthetic results

Varying amounts of Gaussian noise were added to the "ground truth" 2D data. A projective reconstruction was then computed from the noisy data using the various algorithms. The methods were compared according to their residual error which was computed by taking the root-mean-square of the reprojected 3D points from the ground truth 2D data (in pixels). A comparison of the planar reconstruction algorithm using different multi-linear (bifocal, trifocal, quadrifocal) constraints is shown in figure 2 for the room sequence of figure 1. One hundred trials per method used.

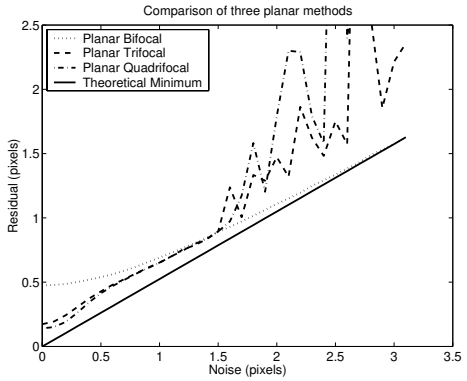

Figure 2: *Comparison of the three planar methods (bifocal, trifocal and quadrifocal) applied to the room sequence (without bundle adjustment). The theoretical minimum is shown in black. All three methods perform well, within about 10% of the theoretical minimum. The trifocal and quadrifocal methods perform slightly better than the bifocal method at low noise levels. The deviations at high noise levels are due to occasional failure of the algorithm, but these noise levels are beyond practical noise limits.*

A second set of experiments was conducted comparing the new planar-based method with two common reconstruction approaches–the incremental and factorization methods discussed in section 7. It is difficult to provide a direct comparison of the methods, because the incremental method is highly dependent on the initial views chosen and the factorization method applies only to 3D points seen across all views, whereas the new method is not similarly restricted. Accordingly, the comparisons were accomplished under conditions most favorable to the other methods, that is, using (RANSAC weeded) features which were visible in all views. Further, the incremental method was initialized with the widest possible baseline–the first, last, and middle images in the sequence. Results are shown in figure 3.

8.2 Real Imagery

The algorithm was also tested on several real sequences. Here the residual error was taken to be the root-mean-square of the reprojected 3D points from the 2D maximum likelihood estimates. Figure 1 shows the actual residuals for the three sequences in figure 4 using the various algorithms, before and after bundle adjustment.

Finally, a full metric reconstruction of the boat sequence was done using the trifocal plane-based projective reconstruction method followed by self calibration. The resultant VRML is shown in figure 5.

Figure 5: *Euclidean VRML*

9 Conclusions

The plane-based method described in this paper is one of the few methods in which the full set of images are handled uniformly and simultaneously to obtain a projective reconstruction. The only other general procedure that does this is the factorization method of Sturm-Triggs [ST96].

It may be seen from the experimental evaluation that the planar-based methods perform very well, when applicable. Of course this is only in cases where a plane may be identified in the images, but this covers a wide class of real-world sequences including the boat and canyon sequences illustrated here.

In cases where neither the factorization nor plane-based methods are applicable, the reconstruction problem is harder, and some sort of incremental method must be used, such as the one discussed here. However, without considerable care, these methods can be unstable and were inferior to the planar and factorization methods on the sequences used here. The planar based method also has the computational advantage of being linear in the number of points used, unlike the factorization method that is quadratic in the

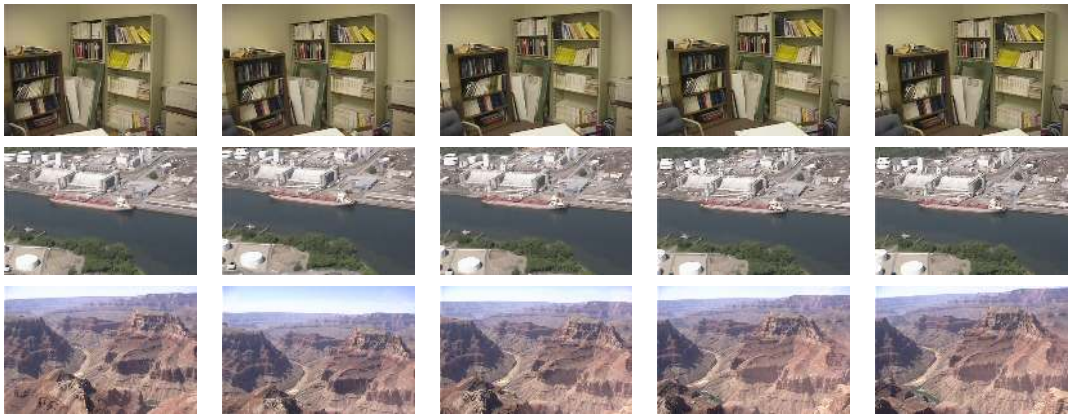

Figure 1: *Snippets from three of the image sequences used to evaluate the algorithms. All three sequences were obtained using a hand-held camcorder–the last two from a helicopter flying overhead.*

number of points. In addition, it does not need points defined in all views, and may even be applied to line matches.

References

[GVL83] G. Golub and C. Van Loan. *Matrix Computations*. John Hopkins University Press, 1983.

[Har95] R. I. Hartley. Multilinear relationships between coordinates of corresponding image points and lines. In *Proceedings of the Sophus Lie Symposium, Nordfjordeid, Norway (*not published yet*)*, 1995.

[Har97] R. I. Hartley. Lines and points in three views and the trifocal tensor. *International Journal of Computer Vision*, 22(2):125–140, 1997.

[Hey98] Anders Heyden. A common framework for multiple view tensors. In *Proc. 5th European Conference on Computer Vision, Freiburg, Germany*, pages 3–19, 1998.

[HGC92] R. I. Hartley, R. Gupta, and T. Chang. Stereo from uncalibrated cameras. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1992.

[HZ00] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[IRP97] M. Irani, B. Rousso, and S. Peleg. Recovery of ego-motion using region alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):268–272, March 1997.

[KAH94] R. Kumar, P. Anandan, and K. Hanna. Shape recovery from multiple views: a parallax based approach. In *ARPA Image Understanding Workshop, Monterey, CA,*, 2929 Campus Drive, Suite 260, San Mateo, California 94403o, November 1994. ARPA, Image Understanding, Morgan Kauffmann Publishers.

[LK81] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of the 7th International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[RC01] C. Rother and S. Carlsson. Linear multi view reconstruction and camera recovery. In *Proc. 8th Int. Conf. on Computer Vision*, 2001.

[Saw94] H. S. Sawhney. Simplifying motion and structure analysis using planar parallax and image warping. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[ST96] P. Sturm and W. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *Proc. European Conference on Computer Vision*, pages 709–720, 1996.

[ST98] R. Szeliski and P. H.S̃. Torr. Geometrically constrained structure from motion : Points on planes. In *Proc. of SMILE98, 3D Structure from Multiple Images of Large-Scale Environments*, pages 171–186, 1998.

[Sut63] I. E. Sutherland. Sketchpad: A man-machine graphical communications system. Technical Report 296, MIT Lincoln Laboratories, 1963. Also published by Garland Publishing, New York, 1980.

[TK92] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization approach. *International Journal of Computer Vision*, 9(2):137–154, November 1992.

[Tri96] W. Triggs. Factorization methods for projective structure and motion. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 845–851, 1996.

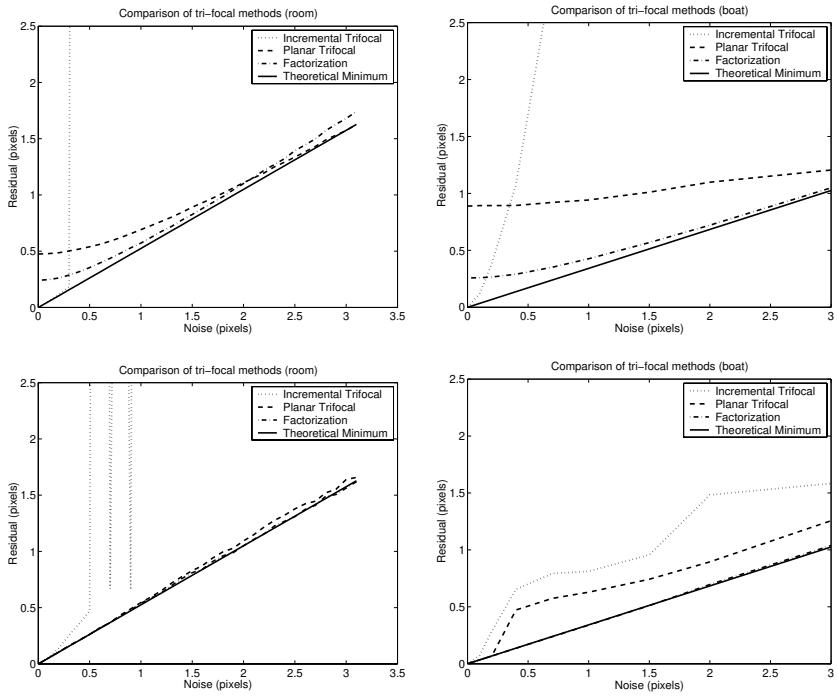

Figure 3: *Comparison of the trifocal-planar method with the factorization and incremental-trifocal methods, as well as the theoretical minimum ground-truth error. Above are the results for the method alone, below the results after bundle adjustment. The graphs on the left correspond to the room sequence and the ones on the right to the boat sequence. Note that both the planar and factorization methods are better than the incremental method for both sequences. The factorization method performs slightly better than the planar method, but after bundle adjustment both are nearly optimal at low noise, and at all noise levels in the room sequence.*

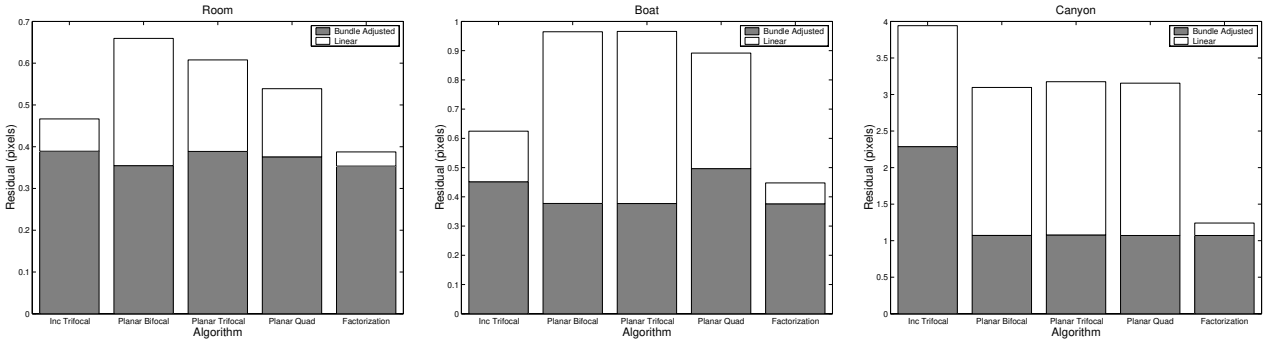

Figure 4: *Results of running the various algorithms on the real room, boat, and canyon sequences in figure 1. The total height represents the residual error after running just the algorithm. The grey bars represent the residual error after bundle adjusting the results of the algorithm. The far left and far right plots correspond to the incremental trifocal and factorization methods while the center three plots within each graph correspond to the various planar methods. Note that although the factorization method initially performs better (Linear error), the bundle adjusted errors are nearly identical.*