National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

Using Projective Vision to find Camera Positions in an Image Sequence *

Roth, G., Whitehead, A.
May 2000

* published in Vision Interface 2000. pp. 87-94, Montréal, Québec, Canada. May 2000. NRC 45873.

Canada

Using Projective vision to Find Camera Positions in an Image Sequence

Gerhard Roth *
Visual Information Technology Group
Building M 50, Montreal Road
National Research Council of Canada
Ottawa, Canada K1J 6H3
Gerhard.Roth@nrc.ca http://www.vit.iit.nrc.ca/roth/home.html

Anthony Whitehead
School of Computer Science
Carleton University
Ottawa, Canada
awhitehe@scs.carleton.ca http://www.scs.carleton.ca/~awhitehe

Abstract

The paradigm of projective vision has recently become popular. In this paper we describe a system for computing camera positions from an image sequence using projective methods. Projective methods are normally used to deal with uncalibrated images. However, we claim that even when calibration information is available it is often better to use the projective approach. By computing the trilinear tensor it is possible to produce a reliable and accurate set of correspondences. When calibration information is available these correspondences can be sent directly to a photogrammetric program to produce a set of camera positions. We show one way of dealing with the problem of cumulative error in the tensor computation and demonstrate that projective methods can handle surprisingly large baselines, in certain cases one third of the image size. In practice projective methods, along with random sampling algorithms, solve the correspondence problem for many image sequences. To aid in the understanding of this relatively new paradigm we make our binaries available for others on the web. Our software is structured in a way that makes experimentation easy and includes a viewer for displaying the final results.

Keywords

Projective Vision, Motion Problems, Structure from Motion, Camera Positions, Model Building

*All correspondence should be addressed to this author

1 Introduction

Recently, a great deal of research has been done in the field of projective vision [1, 2]. The idea is to compute information from image sequences without requiring prior camera calibration. A number of systems have been implemented [3, 4, 5] that can, at least in theory, compute a 3D model automatically from an uncalibrated image sequence.

While the results are impressive there are a number of outstanding issues. One is the fact that these programs are not available publicly in either source or binary form. The only exception we know of is [6]. In this paper we describe a system that uses the projective paradigm to go from an image sequence to a trifocal tensor, and then to a set of camera positions. Our implementation is, to our knowledge, the first publicly available software that computes the trifocal tensor. In this way we hope to make it possible for others to explore and experiment with this new paradigm, which we believe will have a significant influence on the field of computer vision. However, beyond just describing our experience in re-implementing published algorithms, we make a number of other contributions.

In most papers on projective vision, the goal is to compute a projective reconstruction, assuming that no camera calibration information is available [3, 4]. This is followed by a calibration process which enables the conversion of the projective reconstruction to metric (Euclidean) form [7]. The implication is that, if a calibration were available, one should use traditional structure-from motion-algorithms (SFM) to process the image sequence. We claim that this is not the case. Sur-

prisingly, for most image sequences it is not necessarily easier to compute reliable correspondences when calibration information is available. The reason is that the random sampling algorithms, which are the key to dealing with bad correspondences, are much easier to use in a projective framework than in a calibrated framework [8].

Using projective methods in combination with algorithms from the field of robust statistics [9, 10], one can automatically obtain very reliable correspondences for many image sequences, even those with considerable camera motion (i.e. a wide baseline). Producing such accurate correspondences is a multi-step process, where the final result is the trilinear tensor. We take a sequence of images and show that the correspondences that supporting the trilinear tensors are correct and accurate enough to be input directly to a photogrammetric package to compute a set of 3D camera positions, assuming we have a prior camera calibration.

The trilinear tensor holds only for a triple of images [1]. To improve the accuracy of projective reconstruction across an image sequence it is usual to perform a projective bundle adjustment. However, we believe, as do others [11], that the non-linear optimization inherent in the bundle adjustment is better done in metric space than projective space [12]. For this reason, we do not use the tensor to create a projective reconstruction, but only to produce a set of image correspondences. In this way, the tensor need only be accurate enough to identify individual matching features in adjacent images; a required accuracy of only a single pixel. This means that the cumulative error of the tensor over an image sequence is not an issue. The correspondences that support the tensor are used as input to a photogrammetric bundle adjustment program to accurately compute camera positions [13].

Once we have these camera positions, it is possible to rectify these images so that the epipolar lines are horizontal. Then one can compute dense depth maps using traditional stereo algorithms [3]. If the goal is to make 3D models, this is not necessarily the best approach as stereo algorithms will not work in regions without natural texture. For this reason, we believe that it is best to compute dense depth using active methods, since they will succeed even when there is no texture [14]. However, passive methods are sufficient for computing camera positions since this requires not dense depth, but sparse depth, which is much easier to obtain. Our ultimate goal is to create a modeling system that combines both passive and active sensors. The passive sensors will be used to find the position of the active sensors, which in turn will be used to actually obtain the dense depth necessary to make a 3D model. Therefore our goal is to go from an image sequence to a set of camera positions using only passive technology.

2 Projective paradigm

Structure from motion algorithms assume that a camera calibration is available. This assumption is removed in the projective paradigm. To explain the basic ideas behind the projective paradigm, we must define some notation. As in [15]; given a vector $\mathbf{x} = [x, y, \ldots]^T$, $\tilde{\mathbf{x}}$ defines the augmented vector created by adding one as the last element. The projection equation for a point in 3D space defined as $\mathbf{X} = [X, Y, Z]^T$ is:

$$s\tilde{\mathbf{m}} = P\tilde{\mathbf{X}}$$

where s is an arbitrary scalar, P is a 3 by 4 projection matrix, and $\mathbf{m} = [x, y]^T$ is the projection of this 3D point onto the 2D camera plane. If this camera is calibrated, then the calibration matrix C, containing the information particular to this camera (focal length, pixel dimensions, etc.) is known. If the raw pixel co-ordinates of this point in the camera plane are $\mathbf{u} = [u, v]^T$, then:

$$\tilde{\mathbf{u}} = C\tilde{\mathbf{m}}$$

where C is the calibration matrix. Using raw pixel co-ordinates, as opposed to actual 2D co-ordinates means that we are dealing with an uncalibrated camera.

Consider the space point, $\mathbf{X} = [X, Y, Z]^T$, and its image in two different camera locations; $\tilde{\mathbf{x}} = [x, y, 1]^T$ and $\tilde{\mathbf{x}}' = [x', y', 1]^T$. Then it is well known that:

$$\tilde{\mathbf{x}}^T E \tilde{\mathbf{x}}' = 0$$

Here E is the essential matrix, which is defined as:

$$E = [t] \times R$$

where t is the translational motion between the 3D camera positions, and R is the rotation matrix. The essential matrix can be computed from a set of correspondences between two different camera positions [16]. This computational process has been considered to be very ill-conditioned, but in fact a simple pre-processing step improves the conditioning, and produces very reasonable results [17]. The matrix E encodes the epipolar geometry between the two camera positions. If the calibration matrix C is not known, then the uncalibrated version of the essential matrix is the fundamental matrix:

$$\tilde{\mathbf{u}}^T F \tilde{\mathbf{u}}' = 0$$

Here $\tilde{\mathbf{u}} = [u, v, 1]^T$ and $\tilde{\mathbf{u}}' = [u', u', 1]^T$ are the raw pixel co-ordinates of the calibrated points $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}'$. The fundamental matrix F can be computed directly from a set of correspondences by a modified version of

the algorithm used to compute the essential matrix E. As is the case with the essential matrix, the fundamental matrix also encodes the epipolar geometry of the two images. Once E is known, the 3D location of the corresponding points can be computed. Similarly, once F is known the 3D co-ordinates of the corresponding points can also be computed, but in a projective space. The difficulty is that there are fewer invariants in a projective space than a Euclidean space. However, there are still many useful quantities, such as co-linearity, co-planarity and certain other invariants that can be computed in a projective space. Be that as it may, it is the case that the actual supporting correspondences in terms of pixel co-ordinates are identical for both the essential and fundamental matrices. Having a camera calibration simply enables us to move from a projective space into a Euclidean space, that is from F to E.

There is a similar but more elegant concept for three views, called the trilinear tensor. Assume that we see the point $\mathbf{X} = [X, Y, Z]^T$, in three camera views, and that 2D co-ordinates of its projections are $\tilde{\mathbf{u}} = [u, v, 1]^T$, $\tilde{\mathbf{u}}' = [u', v', 1]^T$, $\tilde{\mathbf{u}}'' = [u'', v'', 1]^T$. In addition, in a slight abuse of notation, we define $\tilde{\mathbf{u}}_i$ as the i'th element of $\mathbf{u}$; ie. $\mathbf{u}_1 = u$, and so on. It has been shown that there is a 27 element quantity called the trifocal tensor T relating the pixel co-ordinates of the projection of this 3D point in the three images [1]. Individual elements of T are labeled T_{ijk}, where the subscripts vary in the range of 1 to 3. If the three 2D co-ordinates $(\tilde{\mathbf{u}}, \tilde{\mathbf{u}}', \tilde{\mathbf{u}}'')$ truly correspond to the same 3D point, then the following four trilinear constraints hold:

$$
\begin{aligned}
u'' T_{i13} \tilde{\mathbf{u}}_i - u'' u' T_{i33} \tilde{\mathbf{u}}_i + u' T_{i31} \tilde{\mathbf{u}}_i - T_{i11} \tilde{\mathbf{u}}_i = 0 \\
v'' T_{i13} \tilde{\mathbf{u}}_i - v'' u' T_{i33} \tilde{\mathbf{u}}_i + u' T_{i32} \tilde{\mathbf{u}}_i - T_{i12} \tilde{\mathbf{u}}_i = 0 \\
u'' T_{i23} \tilde{\mathbf{u}}_i - u'' v' T_{i33} \tilde{\mathbf{u}}_i + v' T_{i31} \tilde{\mathbf{u}}_i - T_{i21} \tilde{\mathbf{u}}_i = 0 \\
v'' T_{i23} \tilde{\mathbf{u}}_i - v'' v' T_{i33} \tilde{\mathbf{u}}_i + v' T_{i32} \tilde{\mathbf{u}}_i - T_{i22} \tilde{\mathbf{u}}_i = 0
\end{aligned}
$$

In each of these four equations i ranges from 1 to 3, so that each element of $\tilde{\mathbf{u}}$ is referenced. The trilinear tensor was previously known only in the context of Euclidean line correspondences [18]. Generalization to projective space is recent [2, 1].

The estimate of the tensor is more numerically stable than the fundamental matrix, since it relates quantities over three views, and not two. Computing the tensor from its correspondences is equivalent to computing a projective reconstruction of the camera position and of the corresponding points in 3D projective space. One very useful characteristic of the tensor is image transfer (also called image reprojection). Given any two of $\tilde{\mathbf{u}}, \tilde{\mathbf{u}}', \tilde{\mathbf{u}}''$, and the tensor that holds between the three images, one can compute where the third point *must* be if these points correspond and the tensor is correct. The fundamental matrix and trilinear tensor can be calculated directly from pixel co-ordinates, and have many important and useful characteristics. We believe that there are four reasons for the recent rapid advances in the projective framework.

1. Basic theoretical work defining the fundamental matrix, trilinear tensor and their characteristics.
2. Simple and reliable linear algorithms for computing these quantities from a set of 2D image correspondences.
3. Robust random sampling algorithms for filtering noisy and inaccurate correspondences.
4. A suite of algorithms for doing auto-calibration using only the projective camera positions.

This combination of advances has made it possible theoretically to create a 3D VRML model of a scene from image sequence. It is important to note that, in practice, this process is divided into two distinct phases. The first phase computes from the overlapping image sequence a series of fundamental matrices and trilinear tensors, and consists of the following steps:

- Corner like points are found in each image using a local interest point operator [19, 20] (3.1).
- A feature matcher finds a set of potential corner pair matches between two adjacent images in the sequence [6] (3.2).
- These potential matches are pruned using some type of local consistency filter (3.3).
- A fundamental matrix is computed from the pruned matches using a random sampling algorithm [6, 8, 15] (3.4).
- A set of potential triple matches across three consecutive images are found from the supporting matches from the fundamental matrix (3.5).
- A trilinear tensor is computed from these potential triple matches, again using a random sampling algorithm [21] (3.5).

Producing the trilinear tensor is equivalent to creating a projective reconstruction of the camera position, along with a projective reconstruction of the matching corner points. What is impressive is that the final set of correspondences that support the tensor are in practice, error free, in the vast majority of cases. There are a number of reasons for this result. First, unlike the fundamental matrix, the tensor encodes the constraints among three image pairs. It can therefore

produce correct correspondences in the degenerate situation in which the epipolar lines of the two image pairs of the image triple happen to be collinear. The other reason for the reliable results is the use of robust methods to discard bad correspondences. The process begins with a large number of possibly unreliable corner matches and continually prunes these to a smaller set of more reliable matches.

If the final goal is to produce a dense reconstruction of the scene, then once the trilinear tensor is computed the next phase of the reconstruction process typically consists of the following steps:

- Rectify the image pairs in the sequence so that the epipolar lines are horizontal [3].
- Run a stereo algorithm to compute dense depth from the rectified image pairs [3].
- Auto-calibrate the image sequence to move from a projective to a metric reconstruction [7].

The set of steps in this last phase makes a number of assumptions. The first is that the goal is actually to create a dense 3D metric reconstruction of what has been viewed by the image sequence. However, for some applications, the output of the first phase, a set of projective camera positions, may be sufficient. An example occurs the field of augmented reality in which the goal is to place synthetic objects in an image of a real scene. In this case, the computed tensors can be used to place these synthetic objects in appropriate positions without having either dense depth or metric camera positions.

As previously noted, we believe that for obtaining dense depth it is best to use active, not passive methods. However, for obtaining the position of an active sensor it is feasible to use only an image sequence from a passive co-mounted sensor [22]. Also, in many model building applications it is not difficult to obtain camera calibration, and we assume this information is available. Our goal is to find the 3D camera positions from an image sequence using projective methods to solve the correspondence problem. The details of the procedure are described in the next section.

3 Processing Steps

We now describe the details of the process that takes an overlapping image sequence and computes a set of 3D camera positions. In doing so, we highlight the changes and additions that we have made over what is described in the literature.

3.1 Finding corners/interest points

The first step is to find a set of corners or interest points in each image. These are the points where there is a significant change in image gradient in both the x and y direction. We use the public domain Susan software for this function [19]. Instead of setting a corner threshold, we return a fixed number of corners. This tends to stabilize the results when the images have different contrast and brightness because the proper threshold is selected automatically. In the future we plan to make the required number of corners a function of the average image gradient. The final results are not particularly sensitive to the number of corners. Typically there are in the order of 800 corners found in an image.

3.2 Matching corners

The next step is to match corners between adjacent images. A local window around each corner is correlated against all other corner windows in the adjacent image that are within a certain distance. This distance represents an upper bound on the maximum disparity. We set this upper bound to 300 pixels in x and y for an image size of 750 by 500. Any corner pair between two adjacent images which passes a minimum correlation threshold and has less than the maximum disparity, is a potential match. All such potential matches must then pass a symmetry test. Consider a corner p in the left image of an image pair, and a corner q in the right image. Assume that the strongest match for p in the opposite image, the right image, is labeled $Right(p)$. Similarly for q the strongest match in the left image, is labeled $Left(p)$. The symmetry test requires the correlation be a maximum in both directions; from image a to image b, and vice-versa. In other words a match (p, q) is acceptable if and only if $q = Left(p)$, and $p = Right(q)$.

The symmetry test reduces the number of possible matches significantly and forces the remaining matches to be one-to-one. The total number of possible matches between images is therefore less than or equal to the total number of corner points. Typically, there are only 200 to 500 acceptable matches between 800 pairs of corners. Without the symmetry test constraint there are far more matches; but these matches are much less reliable. For wide baseline images, it is useful to relax the symmetry test and to accept the n best matches (usually in the order of 4). Even in this case we still require that the results be symmetric, that is that each of these matches actually be one of the n best in a symmetric fashion.

3.3 Local consistency testing

The next step is to perform some type of local filter on these matches. The idea is that just by looking at the local consistency of a match relative to its neighbors it is possible to prune many false matches. This is not always done in the literature, but is sensible, since the computational cost of using a local filter is low. One possible approach to prune matches is to use relaxation [15]. We use a simpler relaxation-like process to prune false matches, one based on the concept of disparity gradient [23]. The disparity gradient is a measure of the compatibility of two correspondences between an image pair. Assume the first correspondence maps a corner (a_{lx}, a_{ly}) in the left image to another corner (a_{rx}, a_{ry}), in the right image. Similarly, a second correspondence maps corners (b_{lx}, b_{ly}) into (b_{rx}, b_{ry}). The disparity of these two correspondences are the vectors $\mathbf{d_a} = (a_{rx} - a_{lx}, a_{ry} - a_{ly})$ and $\mathbf{d_b} = (b_{rx} - b_{lx}, b_{ry} - b_{ly})$. The point a_z is the midpoint of the line segment joining a_l, and a_r, and similarly b_z is the midpoint of the line segment joining b_l, and b_r. The vector that joins a_z and b_z is called the cyclopean separation, $\mathbf{d_{cs}}(a, b)$. The disparity gradient is the ratio of the magnitude of these two vectors; $d_{gr} = |\mathbf{d_a} - \mathbf{d_b}|/|\mathbf{d_{cs}}(a, b)|$.

Corner points that are close together in the left image should have similar disparities, and the disparity gradient is a measure of this similarity. Thus, the smaller the disparity gradient, the more the two correspondences are in agreement and vice-versa. The disparity gradient measure has been used in many stereo algorithms to prune invalid correspondences. Typically, these algorithms reject any correspondence with a disparity gradient greater than 1.5. In our case, we compute the disparity gradient of each correspondence with respect to every other correspondence. The sum of all these disparity gradients is a measure of how much this particular correspondence agrees with its neighbours. We iteratively remove correspondences until they all satisfy the condition that the correspondence with maximum disparity gradient sum is within a small factor (usually 2) of the correspondence with minimum disparity gradient sum. Using this simple disparity gradient heuristic we are able to remove significant numbers of bad correspondences at a very low computational cost. Typically, at least 20% of the total number of correlation matches are removed by this process.

3.4 Computing the fundamental matrix

The original matches between image i and j produced by the correlation process are labeled as the set M_{ij}, and the filtered matches that pass the disparity gradient test as the set DM_{ij}. The next step is to use these filtered matches to compute the fundamental matrix which is the uncalibrated version of the essential matrix. This process must be robust, since it can not be assumed that all of the correspondences in DM_{ij} are correct. Robustness is achieved by using concepts from the field of robust statistics, in particular, random sampling. Random sampling is a "generate and test process" in which a minimal set of correspondences, in this case the smallest number necessary to define a unique fundamental matrix (7 points), are randomly chosen [9, 10, 24, 8, 15]. A fundamental matrix is then computed from this best minimal set. The set of all corners that satisfy this fundamental matrix is called the support set. The fundamental matrix F_{ij}, with the largest support set SF_{ij} is returned by the random sampling process.

While this fundamental matrix has a high probability of being correct, it is not necessarily the case that every correspondence that supports the matrix is valid. This is because the fundamental matrix encodes only the epipolar geometry between two images. A pair of corners may support the correct epipolar geometry by accident. This can occur, for example, with a checkerboard pattern when the epipolar lines are aligned with the checkerboard squares. In this case, the correctly matching corners can not be found using only epipolar lines (i.e. computing only the fundamental matrix). This type of ambiguity can only be dealt with by computing the trilinear tensor.

3.5 Computing the trilinear tensor

The trilinear tensor relates the image coordinates of matching corners in three images instead of two images. It is therefore inherently a more stable, and more discriminating quantity that the fundamental matrix [1]. We compute the trilinear tensor from the correspondences that form the support set of two adjacent fundamental matrices in the image sequence. Consider three adjacent images, i, j and k and their associated fundamental matrices F_{ij} and F_{jk}. Each of these matrices has a set of supporting correspondences, which we call SF_{ij} and SF_{jk}. Say a particular element of SF_{ij} is (x_i, y_i, x_j, y_j) and similarly an element of SF_{jk} is (x'_j, y'_j, x'_k, y'_k). Now if these two supporting correspondences overlap, that is if (x_j, y_j) equals (x'_j, y'_j) then the triple created by concatenating them is a member of CT_{ijk}, the possible support set of tensor T_{ijk}. The set of all such possible supporting triples is the input to the random sampling process that computes the tensor. The result is the tensor T_{ijk}, and a set of triples (corner in the three images) that actually support this tensor, which we call ST_{ijk}.

We have gone from a set of corner points C_i, C_j, and

C_k; to a set of matches M_{ij}, and M_{jk}; to a set of filtered matches DM_{ij}, and DM_{jk}; to a pair of fundamental matrices F_{ij}, F_{jk} and support SF_{ij} and SF_{jk}; to a set of computed tuples CT_{ijk}, to a tensor T_{ijk} with support ST_{ijk}. Note that the cardinality of the supporting matches always decreases, but the confidence that each match is correct increases. The entire process begins with many putative matches, and refines these to a few high confidence matches. The final matches that support the tensor, that is ST_{ijk}, range in cardinality from 20 to 100, and in practice, have a very high probability of being correct.

As we stated in the introduction, the goal is to compute the 3D camera positions from the image sequence, not to compute dense depth. We therefore do not perform the steps in the second phase; rectification, stereo and auto-calibration. Instead, we take the correspondences that supports the overlapping tensors and send them to a photogrammetric bundle adjustment program. Assume that we have a sequence of images numbered from 1 to n, and have computed a set of overlapping tensors $T_{123}, T_{234}, \ldots, T_{(n-2)(n-1)(n)}$. Consider the tensors T_{ijk} and T_{jkl} which have supporting correspondences $(x_i, y_i, x_j, y_j, x_k, y_k)$ in ST_{ijk} and $(x'_j, y'_j, x'_k, y'_k, x'_l, y'_l)$ in ST_{jkl}. Those correspondences for which (x_j, y_j, x_k, y_k) equals (x'_j, y'_j, x'_k, y'_k) represent the same corner in images i, j, k and l. In such cases we say that this corner identified in T_{ijk} is continued by T_{jkl}. Each 2D corner point is given a unique identifier, and its continuation is tracked as long as possible in the sequence of tensors. The input to this tracking process is the set of supporting correspondences for the overlapping tensors. The output is a file of numbered corners, where each corner is identified by its 2D co-ordinates in a set of consecutive images. This corner list is then sent directly to the commercial bundle adjustment program Photomodeler [13] using a Visual Basic routine that communicates through a DDE interface. Since we know the camera calibration we use these correspondences to compute the 3D camera positions, along with the 3D co-ordinates of the matching features by running the Photomodeler program.

4 Experiments

We have performed a number of experiments in which we automatically compute the correspondences for a sequence of images using the projective paradigm. We have noticed that the photogrammetric bundle adjustment software will sometimes not converge if the spacing between the image sequence is too small. For this reason we manually skip images in a sequence to maintain an average disparity of at least twenty pixels.

Figure 1: Four images in the castle sequence.

Because of space limitations we describe only two experiments. The first uses the castle sequence that appears in other projective vision papers. This is a set of approximately thirty images in a video sequence of a castle [7]. In Figure 1 are some of the original images in the sequence, while in Figure 2 are the camera positions and the 3D locations of the corner points. In all the images showing the final reconstruction the corner points are the bright dots, and the camera positions are the box-like objects. There are only six camera positions, since we discard intermediate images in the video sequence to make the baselines wide enough for the bundle adjustment program to converge. It should be noted that we have chosen the camera calibration in a rather ad-hoc fashion, simply to make the walls of the castle appear to be 90 degrees. However, even with this ad-hoc calibration our reconstruction compares well to the figures in the literature (see Figure 7.2 of [7]). Therefore it seems that we have computed approximately the same camera positions from this image sequence.

The second example is of a longer sequence of images of an indoor room. In Figure 3 are eight equally spaced images, out of a total of 33 actually used in the experiment. Figure 4 shows two views of the camera position reconstruction. The first image (of the Apple computer) in the sequence is the leftmost camera position. The set of compact dots on the left hand side of the reconstruction figure represent the corner points found on the bookshelf.

There is no doubt that reliable results can be obtained for a wide variety of images, both indoor and outdoor. The software runs on most Unix systems, along with Windows NT, and Windows 98. The input is a sequence of overlapping images, and the output is a series of fundamental matrices and trilinear tensors.

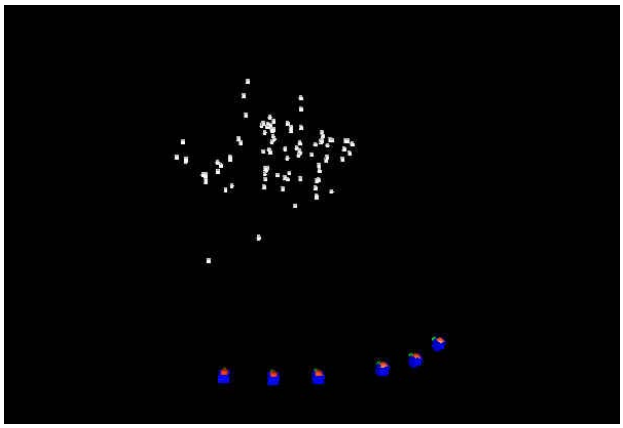

Figure 2: The 3D reconstruction of the camera positions and the corner points for the castle sequence.

The binaries and a number of examples are available on the web site http://www.vit.iit.nrc.ca/roth/home.html. The software is written in a modular form so that each step of the process outputs a file, which is then used as the input to the next step. The software is also able to read many different types of input file formats. We invite the reader to download and test the software on their own image sequences.

5 Conclusions and Discussions

We have implemented a modular system for computing a reconstruction of the camera position in an image sequence. Since our goal is to find the metric camera position we do not need to create a dense 3D reconstruction. We assume that we have a camera calibration available, but do not use this calibration when computing the correspondences. Instead reliable correspondences are computed using the un-calibrated projective method. However, the calibration information is used for computing the 3D camera positions from these correspondences. The final correspondences, those that support the trilinear tensor, are error free in the vast majority of cases. The results are demonstrated experimentally on a number of examples.

In performing our experiments we have drawn some conclusions about the best approaches for each step of the projective reconstruction process. We have also described a new way to locally filter invalid correspondences based on the disparity gradient. We believe that projective methods in combination with random sampling solve the correspondence problem for many image sequences. The support set of the fundamental

Figure 3: Eight equally spaced images (out of thirty-three) in the room sequence.

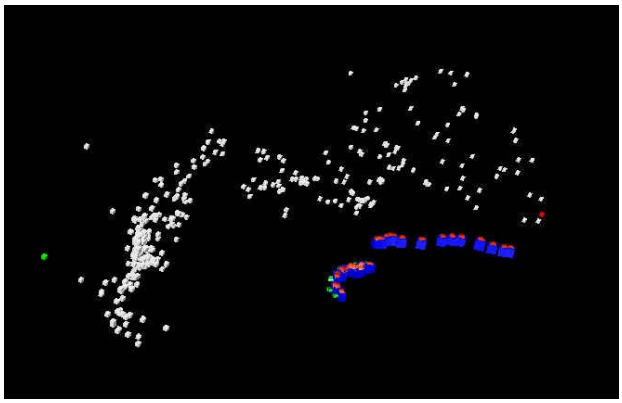

Figure 4: The 3D reconstruction of the camera positions and the corner points for the room sequence.

matrix and trilinear tensor are correct correspondences in the vast majority of cases. If the goal is to compute the metric camera positions and the camera calibration is known, we believe that it is best to send the supporting correspondences of the tensor directly to the photogrammetry software. Our justification for this is that a bundle adjustment process is necessary to compute these positions accurately and we believe that this is better done in metric space, rather than projective space. In the future we plan to perform more systematic experiments where the ground truth is known.

Acknowledgements

The authors would like to thank Dr. Toshio Ueshiba of ETL in Japan for many helpful discussions and useful software when starting our work on projective vision.

References

[1] A. Shashua, "Algebraic functions for recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 779–789, 1995.

[2] R. Hartley, "A linear method for reconstruction from lines and points," in *Proceedings of the International Conference on Computer Vision*, pp. 882–887, Cambridge, Mass., June 1995.

[3] R. Koch, M. Pollefeys, and L. VanGool, "Multi viewpoint stereo from uncalibrated video sequences," in *Computer Vision-ECCV'98*, pp. 55–71, 1998.

[4] M. Pollefeys, R. Koch, M. Vergauwen, and L. VanGool, "Automatic generation of 3d models from photographs," in *Proceedings Virtual Systems and MultiMedia*, 1998.

[5] A. Fitzgibbon and A. Zisserman, "Automatic camera recovery for closed or open image sequences," in *ECCV'98, 5th European Conference on Computer Vision*, (Freiburg, Germany), pp. 311–326, Springer Verlag, June 1998.

[6] Z. Zhang, R. Deriche, O. Faugeras, and Q.-T. Luong, "A robust technique for matching two uncalibrated images through the recovery of the unkown epipolar geometry," *Artificial Intelligence Journal*, vol. 78, pp. 87–119, October 1995.

[7] M. Pollefeys, *Self-calibration and metric 3d reconstruction from uncalibrated image sequences*. PhD thesis, Catholic University Leuven, 1999.

[8] P. Torr and D. Murray, "Outlier detection and motion segmentation," in *Sensor Fusion VI*, vol. 2059, pp. 432–443, 1993.

[9] P. J. Rousseeuw, "Least median of squares regression," *Journal of American Statistical Association*, vol. 79, pp. 871–880, Dec. 1984.

[10] R. C. Bolles and M. A. Fischler, "A ransac-based approach to model fitting and its application to finding cylinders in range data," in *Seventh International Joint Conference on Artificial Intelligence*, (Vancouver, British Colombia, Canada), pp. 637–643, 1981.

[11] H. Sawhney, Y. Guo, J. Asmuth, and R. Kumar, "Multi-view 3d estimation and applications to match move," in *1999 IEEE Workshop on Muli-View Modelling and Analysis of Visual Scenes*, pp. 21–28, 1999.

[12] P. Mclauchlan, "Gauge invariance in projective 3d reconstruction," in *IEEE Workshop on Multi-View Modelling and Analysis of Visual Scenes*, pp. 37–44, IEEE Computer Society, 1999.

[13] *Photomodeler by EOS Systems Inc.* http:/www.photomodeler.com.

[14] P. Besl, "Active, optical range imaging sensors," *Machine Vision and Applications*, vol. 1, no. 1, pp. 127–152, 1988.

[15] G. Xu and Z. Zhang, *Epipolar geometry in stereo, motion and object recognition*. Kluwer Academic, 1996.

[16] H. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, pp. 133–135, 1981.

[17] R. Hartley, "In defence of the 8 point algorithm," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, 1997.

[18] M. Spetsakis and J. Aloimonos, "Structure from motion using line correspondences," *International Journal of Computer Vision*, vol. 4, pp. 171–183, 1990 1990.

[19] S. Smith and J. Brady, "Susan - a new approach to low level image processing," *International Journal of Computer Vision*, pp. 45–78, May 1997.

[20] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Ivey Vision Conference*, pp. 147–151, 1988.

[21] P. Torr and A. Zisserman, "Robust parameterization and computation of the trifocal tensor," *Image and Vision Computing*, vol. 15, no. 591-605, 1997.

[22] P. Jasiobedski, "Fusing and guiding range measurements with colour video images," in *Proceedinsg International Conference on Recent Advances in 3-D Digital Imaging and Modelling*, (Ottawa, Ontario), pp. 339–347, IEEE Computer Society Press, 1997.

[23] R. Klette, K. Schluns, and A. Koschan, *Computer Vision: three-dimensional data from images*. Springer, 1996.

[24] G. Roth and M. D. Levine, "Extracting geometric primitives," *Computer Vision, Graphics and Image Processing: Image Understanding*, vol. 58, pp. 1–22, July 1993.