Computing Camera Positions from a Multi-Camera Head

Roth, Gerhard

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

Computing Camera Positions from a Multi-Camera Head *

Roth, G.
June 2001

* published in Third International Conference on Recent Advances in 3D Imaging and Modelling. pp. 135-142, Québec, Québec, Canada. NRC 45871.

Canada

Computing Camera Positions from a Multi-Camera Head

Gerhard Roth
Visual Information Technology Group
National Research Council of Canada
Ottawa, Canada K1A OR6
Gerhard.Roth@nrc.ca http://www.vit.iit.nrc.ca/roth/home.html

Abstract

This paper presents a method of computing camera positions from a sequence of overlapping images obtained from a binocular/trinocular camera head. First, we find matching features among the images at each camera head position. Because the individual cameras are calibrated we can directly compute the 3D co-ordinates of these features using triangulation. Then we find matching features across adjacent images in the camera sequence. We compute the fundamental matrix between image pairs, and then the trilinear tensor between image triplets. The corresponding features that support the overlapping trilinear tensors are very reliable. Some of these matching features across the image sequence are also matching features among the images at each camera head location. This creates a potential set of matching 3D features between the adjacent images in the sequence. We compute the transformation between the camera positions in the image sequence using these matching 3D feature co-ordinates. Using multiple cameras has a number of advantages over computing the camera positions from a single camera. We directly obtain a Euclidean reconstruction of the camera path, we can reliably process very small motions, and there are no motion degeneracies.

1 Introduction

In traditional structure from motion (SFM) algorithms the goal is to find both the camera motion and the 3D structure of the scene from a series of overlapping 2D images. Our belief is that finding dense 3D structure from passive sensors is very difficult, because the texture necessary for passive methods to extract depth is not present everywhere. To acquire dense 3D structure active sensors are often necessary [1]. However, most scenes have enough texture to support the computation of sparse 3D structure, and this is sufficient to accurately compute the camera motion. In this paper we describe a method to compute the camera motion using a multi-camera passive head. There are many applications for robustly and accurately computing the camera motion - such as model building [2], tracking, and augmented reality.

There have been a number of systems that take overlapping images from a single camera and compute the camera path using a combination of random sampling and projective vision [3, 4]. First the fundamental matrix is computed between image pairs, and then the trilinear tensor between image triples. The correspondences that support the overlapping trilinear tensors have been shown to be very reliable. If we know the camera calibration then a reconstruction in metric space (up to a scale factor) of the camera positions can be computed. The camera calibration must either be known a-priori, or be obtained using an auto-calibration algorithm [5]. While successful, this approach still suffers from the problems inherent in any single camera reconstruction algorithm; the reconstruction has an unknown scale factor, there are problems handling degenerate motions[1], and poor accuracy for small camera motions. In this paper we describe the use of a passive binocular/trinocular head [6], consisting of two or all three cameras arranged in a triangle, to compute the camera motion. We believe that using multiple cameras to compute the camera positions overcomes the noted limitations inherent in the use of a single camera. Our idea is to use the 3D locations of the matching feature points of the binocular/trinocular images at each head location to register adjacent positions of the camera. This has a number of advantages.

We eliminate the scale factor so the reconstruction of the camera path is in Euclidean rather than metric space [4]. We can more accurately handle small camera motions which are very common, especially in video sequences, and we can also deal with degener-

[1] For example, when a single camera rotates around the center of projection the motion is degenerate. Another degeneracy occurs when imaging planar objects.

0-7695-0984-3/01 $10.00

ate motions. With a single camera it is necessary to explicitly test for such degeneracies, and in such cases to invoke special algorithms, a non-trivial process [7]. Multiple cameras also have the advantage of producing more reliable correspondences.

Combining stereo and SFM algorithms is not a new idea [8, 9, 10, 11]. However, our combination of Euclidean stereo and the projective approach to SFM is novel. We use only very simple and efficient binocular/trinocular matching algorithms. However this will always result in a significant number of matching errors, so not all the 3D data for each camera position will be reliable. An Iterative Closest Point (ICP) algorithm [12] would therefore fail to correctly register the camera positions if it used only this 3D data. The idea is to compute more reliable matches across the image sequence using the SFM algorithms. One possibility is to compute a projective reconstruction from the SFM algorithms and use this reconstruction to find better stereo matches [11]. By contrast, we do the opposite. We first find Euclidean features at each camera head location, and then use the projective SFM algorithms only to find reliable correspondences across the image sequence. We never compute a projective reconstruction. This issue is discussed in more detail in the conclusions.

2 Hardware Configuration

The camera we use is a commercially available trinocular head [6]. For each position of the head we have three orthogonal images labeled left, right and top. Assume that we have moved the camera head through a path in space. The images from the three cameras at the first camera head location are labeled left1, right1, top1; the second location are labeled left2, right2, top2; and for the last of the N head locations are labeled left$N$, right$N$, and top$N$ as shown in Figure 1.

The camera head is calibrated so the focal length and camera baselines are known. In our case the camera baseline is ten centimeters. The three images are rectified [4], which implies that the epipolar lines between the left and right images are horizontal, and between the right and top images are vertical. However, this is not necessary for our approach to succeed. All that we need to know is the epipolar geometry between the image pairs, which we can compute from the camera head calibration. The commercial head we use provides 3D data, but we have chosen to compute our own 3D feature points using either two cameras at each head position (binocular), or all three cameras at each head position (trinocular). We compute a relatively sparse set of matching features between the images at each head location, since this is sufficient for our application. One of our goals is to determine if the results with a three camera head are significantly superior to a two camera head.

3 Processing Steps

The processing steps that we use to compute the camera positions are as follows:

1. Compute a set of 2D corner points for two (binocular) or three (trinocular) images at each camera head position [13]. These corners are the image features that will be used in all further matching.
2. Find corresponding corner points *among* the two or three images at each camera head location [14].
3. Calculate the 3D co-ordinates of these matching corner points at each camera head location by triangulation.
4. Find corresponding corner features *across* the adjacent right images in the camera head sequence [4, 3].
5. For each corresponding corner feature *across* the right images check if it is also a member of the set of matching corners *among* the other images in each camera head location.
6. Use the 3D co-ordinates of these corner features *among* the images at each head location to compute the transformation between adjacent camera head positions.
7. Adjust all these computed camera head positions to be relative to the first head position by composing the aligning transformations.

We will now describe each of these steps in greater detail. In the first step we compute a set of corners in each of the left, right and possibly top images at each head location. These corners represent points of high texture in two directions and are good choices as potential match points. We use a well known corner operator [13], and set the threshold to produce a relatively large number of corners (1200). The final results are not sensitive to the corner threshold.

In the second step we match the corners among the images at each camera head location via a local correlation operation. All corner points within a certain distance of each other (in this case 40% of the image size) are matched. For binocular matching, we use

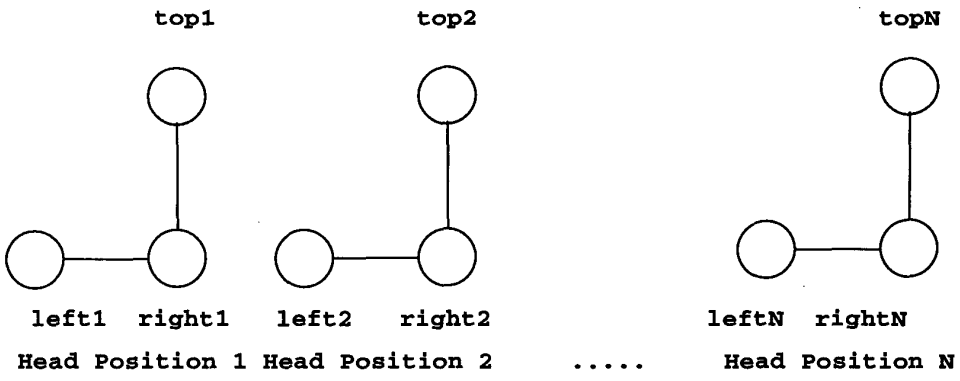

Figure 1: The binocular/trinocular image sequence of the moving camera head.

only the left and right images and ignore the top image. For trinocular matching we also match the right and top images in the same fashion, which produces a set of matching corners across three images. Matching across three images produces fewer but more reliable matches than matching across two images.

Since the camera head is calibrated we know the baseline between the cameras along with the internal camera parameters. In the third step we use the camera calibration along with the pixel co-ordinates of the matching binocular or trinocular features to find the 3D co-ordinates of these features relative to the current camera head position. In Figure 2 we see the three camera views, and the reconstructed 3D features and camera locations for a single position of the binocular/trinocular head. In the left hand side of this figure we see features computed using two cameras and in the right hand side features computed using three cameras. This process is repeated until we have computed the 3D features for each camera head location using both two and three cameras.

In the fourth step we compute the matching corners across the right image at each camera head location; that is across images right1, right2, up to right$N$ where N is the last position of the camera head in the sequence. To summarize the process, we first compute the fundamental matrix between the adjacent right image pairs, and then use these correspondences as the basis of the computation of the trilinear tensor between adjacent image triples. The result is a set of reliable correspondences across the right image of the camera head sequence. The basic method has been implemented by a number of researchers [14, 7, 3, 4]. We have made some improvements to the approach [3]. The executables for our software, which we call the Projective Vision Toolkit, are available on our website.

In the fifth step we find the 3D feature that is associated with each of the matching 2D features computed across the right images in the sequence. Some of the matching 2D corners across the right images in the sequence are also members of the matching corners among the left, right (and possibly top) images at each camera head location. More formally, assume that a 2D corner at pixels $(r1x, r1y)$ in image right1 matches a 2D corner at pixels $(r2x, r2y)$ in image right2. If corner $(r1x, r1y)$ is a member of a set of matching 2D corners among the images at camera head location one then this pixel also represents a 3D data point, which we label $(x1, y1, z1)$. Similarly if corner $(r2x, r2y)$ is a member of a set of matching 2D corners among the images at camera head location two then this pixel also represents an associated 3D data point, which we label $(x2, y2, z2)$. These two 3D points therefore represent the same feature in space. However, $(x1, y1, z1)$ is the location of this feature relative to camera head location one, and $(x2, y2, z2)$ is the location of this feature relative to camera head location two. We find the associated pair of matching 3D data points for each matching 2D feature point across the right images in the sequence

In step six we take the set of matching 3D points for every pair of adjacent camera head locations and compute the rigid transformation that aligns them using the quaternion algorithm [15]. However, to deal with potential errors in the matching process we make this process more robust by performing random sampling using the LMeDs approach [16]. For each pair of adjacent camera head positions we have a set of potentially

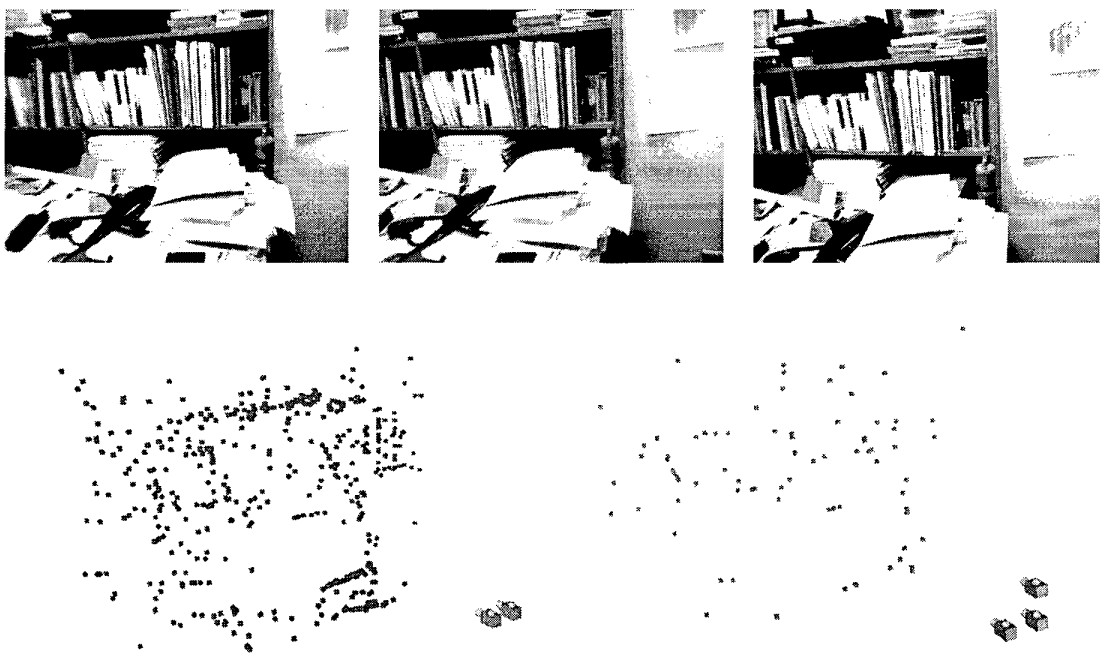

Figure 2: The three camera views and the reconstructed 3D features and camera positions (using two or three cameras) for a single position of the binocular/trinocular camera head.

matching 3D points, usually in the range of fifty to one hundred such points. Using three randomly chosen matching 3D points from this set we compute a putative transformation that aligns all the remaining matching 3D points. After applying this transformation we compute the residuals of all the corresponding 3D points, and find their median. This random sampling process is repeated a number of times (typically thirty times) and the aligning transformation with the smallest median residual error is saved. Those corresponding 3D points whose residual error is greater than a certain multiple of the median residual error are likely to be errors and are discarded. This robust 3D alignment algorithm produces a correct transformation even when up to fifty percent of the matching 3D points are incorrect. This process is repeated for each set of matching 3D points across each pair of adjacent camera head locations. The result is a series of transformations that aligns two adjacent camera head positions; i.e. we say that transformation T_i aligns camera position $i+1$ with camera position i.

The seventh, and last step is to cascade these aligning transformations to place them all in the same coordinate frame. This is done by applying the aligning transformations for a camera pair to that camera and to all successive camera positions in the sequence. For example, transformation T_1 which aligns the camera head at position two to position one is applied to all the camera head positions from two to N. This process must be repeated iteratively for each transformation from T_1 to T_{N-1}. Once this is done all the camera positions are in a consistent co-ordinate frame and are in the correct Euclidean relationship to each other. We have finished the process and now know the sequence of camera positions of the binocular/trinocular head in Euclidean space.

4 Experiments

Here we describe four experiments that demonstrate the utility of the approach. In each case we move the camera head through a number of positions (in the range of ten to sixty). We then compute the camera positions using 3D features from two images (binocular) and from three images (trinocular). The first sequence is a series across the top of a desk. Here the camera is moving in an approximately horizontal path and there are sixty camera positions. The second series is across a bookshelf. Here the camera motion

Figure 3: Views of the camera path reconstruction for the bookshelf (left) example and the wall (right) example along with the supporting 3D features.

is vertical, and there are fifty camera positions. The third is a series across a bookshelf with rotational motion, and there are fifty camera positions. The fourth series is the across a set of wall posters. Here the camera is moving in a horizontal path and there are nine camera positions. In Figure 3 we show the computed path of the right camera and the reconstructed 3D features for the one of the bookshelf examples and the wall example. In Figure 4 we show a more detailed pictorial view of the desk example. In the top part of this figure are the right images for the first, middle and last position of the camera head in the sequence. Below this are four different views of the reconstruction of the camera positions and the associated 3D features. In both Figures we only show the reconstructed 3D features actually used to compute the transformations between the camera positions.

All these examples are in unstructured environments with no targets, using only natural features. While the results look good visually, we need a more formal way of evaluating the performance. Ideally, we would have a calibrated camera with a known trajectory, and compare our computed camera positions with the known camera positions. We plan to do such experiments but for now we have some simple but effective ways to evaluate the results. Consider a set of matching 3D features that we have computed across the image sequence. Is there a way to check whether there are gross errors in these correspondences? There are two good measures of matching reliability. The first is the distance in 3D space between the matching 3D features. We call this measure the average 3D correspondence error. The second, which we feel is even more indicative of matching reliability, is the 2D reprojection error. This reprojection error is known to be a good measure of the accuracy of the results for any reconstruction algorithm [17]. After the matching 3D features have been aligned we project them back into their associated 2D images. Then we compute the distance in pixels between the 2D reprojections of these 3D matches and their original 2D location in the images. If the average reprojection error for all the 3D matches is low (less than a pixel) and there are a significant number of matching 3D features (at least fifty) for each adjacent camera position then we can be confident the computed camera positions are correct.

Table 1 shows the reconstruction results using two cameras and Table 2 the results using three cameras. We also show three additional measures; the average number of matching 3D features between adjacent camera positions, the average distance of the 3D features from the camera, and the average translational motion of the camera. All measurements are in millimeters. In both the trinocular and binocular case the quality measures (especially the reprojection error) indicate that the camera positions are correct. The second example is obtained for a planar scene, and the fourth is from a rotational camera motion. Both are degeneracies for traditional structure from motion single camera algorithms, but cause no problems here. As can be seen from the Tables, the camera

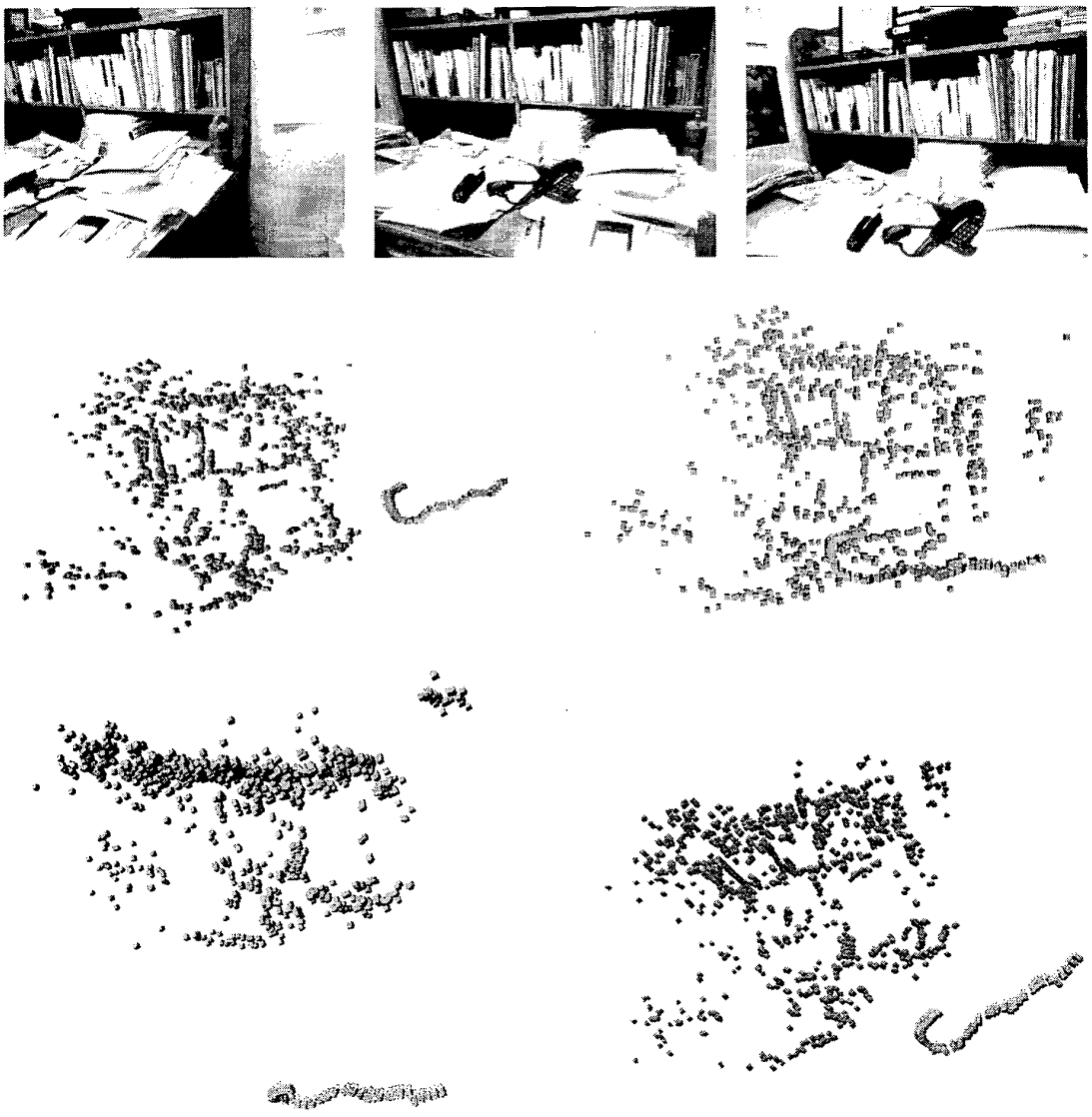

Figure 4: The first, middle and last image of the right camera of a sequence across the top of a desk. Below are four different views of the camera path reconstruction along with the supporting 3D features.

Name of the Example	Desk	Bookshelf1	Bookshelf2	Wall
Total # of images	60	50	50	10
Avg. # matching features per image pair	83	93	152	72
Avg. 3D feature correspondence error (mm)	3.90	0.64	1.03	0.95
Avg. 2D feature reprojection error (pixels)	0.54	0.20	0.22	0.34
Avg. 3D feature distance from camera (mm)	1069	548	660	422
Avg. 3D translational camera motion (mm)	12.67	2.33	1.14	5.75

Table 1: Binocular results for four examples.

Name of the Example	Desk	Bookshelf1	Bookshelf2	Wall
Total # of images	60	50	50	10
Avg. # matching features per image pair	13	22	34	13
Avg. 3D feature correspondence error (mm)	1.76	0.38	0.53	0.37
Avg. 2D feature reprojection error (pixels)	0.34	0.15	0.17	0.18
Avg. 3D feature distance from camera (mm)	1128	546	660	425
Avg. 3D translational camera motion (mm)	16.13	2.49	1.42	5.93

Table 2: Trinocular results for four examples.

motion is relatively small in all four examples. This would also cause a problem for a single camera structure from motion algorithm. However, the multiple camera approach successfully handles small motions.

We have a choice as to whether to use binocular or trinocular features, that is whether to use a two or three cameras to produce the initial 3D features. In both cases the remaining parts of the processing sequence as described in the last section are unchanged. We see from the tables that there are more binocular features, but they tend to be slightly less accurate than the trinocular features. However, it seems that there are significantly more binocular features (four times on the average) which gives us considerably more confidence in the binocular results.

5 Discussions and Conclusions

This method of computing camera positions can easily be generalized to other types of sensors as long as they provide co-registered 3D data. Consider a sequence of overlapping 2D images, and in particular consider one image, I. We define $I(i,j)$ as the 2D pixel at index i,j in this image. Now assume that there are three co-registered 3D images, $X(i,j)$, $Y(i,j)$ and $Z(i,j)$, that define the locations of the associated 3D data point of the 2D pixel $I(i,j)$. In other words $x = X(i,j), y = Y(i,j)$ and $z = Z(i,j)$ is the location in 3D of the point in space that is intersected by the ray from the camera origin through pixel i,j in the camera image plane. If there is no 3D data available for that 2D pixel, then we simply define $X(i,j) = Y(i,j) = Z(i,j) = -1$.

This model can accommodate all types of 3D sensors, from passive to active systems. Some of these systems will produce sparse 3D data, and some dense 3D data. If dense 3D data exists then it is possible to use this directly to deal with very large camera motions [18]. In our case we assume that the motion in the camera sequence is more constrained. More precisely we assume that the disparities of the matching feature points between the images across the sequence is less than one third of the image size.

While there are more 3D points when we use binocular features, the chance of a gross mismatch error producing an invalid 3D feature point is clearly higher than when we use trinocular features. However, because we compute the 3D transformation between images robustly, we are able to discard invalid matching 3D feature points. Therefore because of the robustness of our approach the results indicate that we can compute the camera positions very reliably using a binocular system, a trinocular system is not necessary.

We have yet to perform systematic experiments by moving the camera head through motions for which the ground truth is know. We plan to do this by mounting the camera head on a calibrated track. How-

ever, the reprojection error alone is a strong indicator of the quality of any reconstruction process. From the small value of this reprojection error we can see experimentally that the approach has achieved reliable results. An important characteristic of our method is that the accuracy of the estimated camera positions depends only on the accuracy of the 3D data points computed at each camera position, which in turn depends on the baseline of the cameras in the binocular/trinocular head. This accuracy is independent of the distance between the cameras across the image sequence, which is what governs the accuracy of the results when a single camera is used. However, since we only compute a pair-wise registration of the 3D points as the number of camera positions in the sequence increases we will accumulate errors. To reduce these errors we must compute a final 3D transformations for *all* the camera positions in the sequence simultaneously. This would reduce the cumulative errors. Such group ICP algorithms exist and we are in the process of implementing one for this application [12].

There is similar work that uses the ideas of combining easier correspondence from motion with the more accurate results from a stereo head [11]. However, this approach must first compute an accurate projective reconstruction, which is difficult to do with small camera motions. We compute the 3D features for each camera head position, which can can be done efficiently with binocular/trinocular matching, but which produces errors. Then we use the more reliable projective matches produced by the trilinear tensor across the motion to align the associated 3D features at each camera head position. Our approach is simpler than [11], can work with any algorithm that produces co-registered depth at each camera position, and works well.

In summary, having a multi-camera head enables us to directly produce 3D data points at each camera head position. Using these 3D points to compute the camera position means that we know the positions in Euclidean space, we can reliably process very small motions and handle motion degeneracies. We have shown that excellent results can be obtained by using only a binocular system. A trinocular system is not necessary for reliably computing camera motion. The main drawback over using a single camera is the cost and weight of the extra camera. There is almost always enough texture for a passive system to get a sufficient number of 3D data points to compute the camera pose.

References

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

[2] G. Roth, "Building models from sensor data: an application shared by the vision and graphics community," in *Nato Meeting: Confluence of Computer Vision and Computer Graphics*, pp. 329–338, 1999.

[3] G. Roth and A. Whitehead, "Using projective vision to find camera positions in an image seqeuence," in *Vision Interface 2000*, (Montreal, Canada), pp. 87–94, May 2000.

[4] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge Uinversity Press, 2000.

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

[6] *Digiclops by Point Grey Research*. www.ptgrey.com.

[7] P. Torr, A. Fitzgiobbon, and A. Zisserman, "The problem of degeneracy in structure from motion and motion recovery from uncalibrated images," *International Journal of Computer Vision*, vol. 32, pp. 27–44, August 1999.

[8] K. Hanna and N. Okamoto, "Combinng stereo and motion analysis for direct estimation of scene structure," in *Proceedings of the Fourth International Confernce on Computer Vision*, pp. 357–365, 1993.

[9] Z. Zhang, "Motion and structure of four points from one motion of a stereo rig with unknown extrinsic parameters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 12, pp. 1222–1227, 1995.

[10] P.-K. Ho and R. Chung, "Stereo-motion with stereo and motion in complement," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 215–220, 2000.

[11] F. Dorniaka and R. Chung, "Cooperative stereo-motion: matching and reconstruction," *Computer Vision and Image Understanding*, no. 79, pp. 408–427, 2000.

[12] R. Benjemaa and F. Schmitt, "A solution for the registration of multiple 3d point sets using unit quaternions," in *Computer Vision-ECCV'98*, pp. 34–50, 1998.

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

[14] 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.

[15] B. Horn, "Close-form solution of absolute orientation using quaternions," *Journal of the Optical Society of America*, vol. 5, no. 7, pp. 1127–1135, 1988.

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

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

[18] G. Roth, "Registering two overlapping range images," in *Second International Conference on Recent Advances in 3D Imaging and Modelling*, (Ottawa, Canada), pp. 302–311, October 1999.