View Synthesis from Unregistered 2-D Images

Parag Havaldar, Mi-Suen Lee, and Gérard Medioni
Institute for Robotics and Intelligent Systems
Department of Computer Science
University of Southern California
Los Angeles, CA 90089-0273
U.S.A.

Phone: 213-740-6428 Fax: 213-740-7877
e-mail: havaldar, misuen, medioni@iris.usc.edu

Abstract

Synthesizing the image of a 3-D scene as it would be captured by a camera from an arbitrary viewpoint is a central problem in Computer Graphics. Given a complete 3-D model, it is possible to render the scene from any viewpoint. The construction of models is a tedious task. Here, we propose to bypass the model construction phase altogether, and to generate images of a 3-D scene from any novel viewpoint from prestored images. Unlike methods presented so far, we propose to completely avoid inferring and reasoning in 3-D by using projective invariants. These invariants are derived from corresponding points in the prestored images. The correspondences between features are established off-line in a semi-automated way. It is then possible to generate wireframe animation in real time on a standard computing platform. Well understood texture mapping methods can be applied to the wireframes to realistically render new images from the prestored ones. The method proposed here should allow the integration of computer generated and real imagery for applications such as walkthroughs in realistic virtual environments. We illustrate our approach on synthetic and real indoor and outdoor images.

Keywords: Image based rendering, projective invariants, epipolar geometry.

1 Introduction

The generation of photo-realistic images of a 3-D scene from varying viewpoints is a central problem in Computer Graphics. The traditional approach to this problem consists of constructing a 3-D geometric model of the scene, and then rendering it from any arbitrary viewpoint. These two steps are both very expensive:

• 3-D model construction is a tedious off-line operation. For a completely synthetic environment, one needs to specify all the objects in terms of facets and their relationships to each other. Methods for constructing models automatically from images are not yet mature. Data obtained from range finder has to be segmented and merged before it can be used [6]. In the case of intensity images, the difficult "structure from motion" problem [1],[17],[13] has to reliably be solved.

• Rendering realistic images is a computationally expensive process, with a complexity depending on the number of objects in the scene. Rendering for real time applications requires specialized hardware.

This complexity of rendering can be reduced by computing a few views and interpolating between them. Chen and Williams [4] have proposed a method for interpolating a computer generated scene from closely spaced viewpoints. Using a dense depth map of the scene and the positions of the cameras, they are able to generate pixel correspondences. With these correspondences, they map texture from an array of prestored images onto the new image. Greene [9] has developed a method to generate an image from the so-called environment maps, which are Z-buffered images rendered from a set of discrete viewpoints in 3D space. These methods assume full 3-D data is always available, which may not be so, for example when dealing with real world images. The requirement of 3-D models is a serious limitation.

Some researchers have tried to infer 3-D information directly from images. Various methods have been developed for recovering both the shape of an object and the motion relative to the camera from a sequence of images [1], [17]. These approaches essentially track feature points throughout the sequence of images and obtain 3-D information by solving the basic imaging equations. Limitation of these methods relate to the large number of close views required, and to high numerical instability due to nonlinear nature of the imaging equations.

Another approach is to perform image based rendering, in which case, no explicit 3-D model of object and

environment is required. Researchers have approached this by storing an array of 2D images of a scene in the form of a mosaic [18][19]. Recently, panoramic views and cylindrical projections of scenes have been obtained [5], [14]. These approaches offer limited navigation capabilities.

In this paper, we show that it is possible to sidestep the 3-D model construction phase. Using just two images of a scene, without knowing their camera positions, we can synthesize any novel viewpoint of the scene. To generate such new viewpoints, we use *projective invariants* which require correspondences of feature points between two initial images. These points are grouped to form planar surfaces for the purpose of rendering. The generation of wireframes is extremely fast and can be animated in real time on standard computing platforms. The appearance of the face is obtained by texture mapping one of the existing views of the face in a prestored image onto the new one. In this paper, we demonstrate results on polyhedral objects. Dealing with objects with curved surfaces will require obtaining fine point correspondences, which we currently do not address.

We make the following assumptions:

- Scenes are static. So, the transformation between the input images is only a camera displacement.
- Illumination is diffuse. Any specular components, if present, would be incorrectly mapped by the texture mapping process.
- The imaging process is well approximated by a pinhole camera model.

2 Geometric Invariants - a Brief Summary

The novelty of our approach lies in the use of projective geometry and invariant theory, namely the use of *cross ratios*. Invariant theory has been established for many years, but applications of invariants to other areas such as computer vision and graphics have been demonstrated only recently [15] [16].

An invariant, defined with respect to a transformation T, is a property which remains unchanged under this transformation. If T is a rotation of two lines in 2-D, as shown in figure 1, the angle between the lines remains unchanged, and hence is an invariant. Under orthographic projection, parallel lines always map to parallel lines. Here, parallelism is an invariant. Under orthographic projection, the ratio of points on a line also remains unchanged as illustrated in figure 1.

Under perspective, the ratio of the ratio of points on a line, known as the *cross ratio*, is preserved. It is precisely this cross ratio which is used in our method. In figure 2, the line L_1 is shown with four points on it. L_1

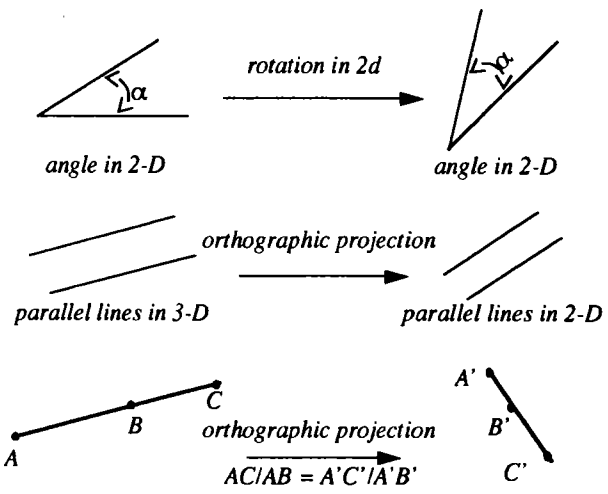


figure 1 Examples of invariants under orthographic transformations

maps to L_2 under a projective transformation. Although the relative positions of the points on the 2 lines change, the cross ratio, as defined below, remains constant. The proof and other properties of this projective invariant may be found in [15] or other books on projective geometry. Given four points on a line, the cross ratio is uniquely determined. Conversely, given a cross ratio and three points, the location of the fourth point can be uniquely computed. The choice of these points and the computation of the cross ratio is described in section 4.4.

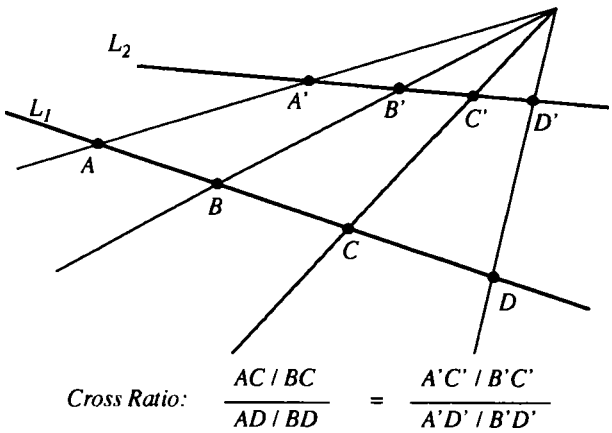


figure 2 The cross ratio is a projective invariant

3 Overview of the approach

The overview of our system is shown in figure 3. We first extract a set of "interesting" features in existing views. In case of polyhedral objects, such features should correspond to vertices and faces. The extraction of these features is performed off-line, semi-interactively using computer vision techniques and is explained in section

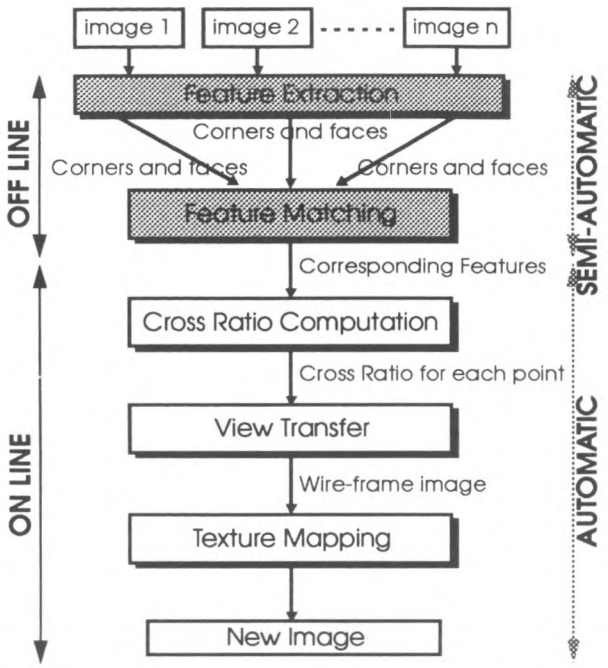


figure 3 Description of our approach

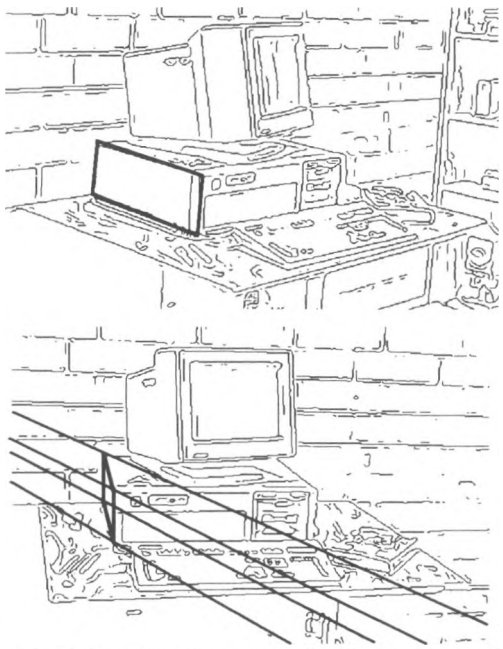


figure 4 Edge image of one view showing one detected face and vertices (above). Edge image of second view showing corresponding epipolar lines and matched vertices (below)

4.1. For each point and face, correspondences are generated in the other image. This stage, which is described in section 4.2, again uses well understood computer vision constraints, namely the epipolar constraint. User interaction is sometimes required to edit wrong correspondences.

The correspondences established enable computation of the cross ratio for every point, as explained in section 4.4. In section 4.5, we show how the cross ratio can be used to generate the image positions of these points in a novel view. Once the position of each face is known in the new scene, texture information can be mapped from the corresponding faces in the initial images. This is explained in section 4.7.

Finally, in section 5, we show experimental results of our approach. We first apply our technique to synthetic images, which enables us to quantitatively compare the predicted view to the accurately rendered view. We then show our approach on *real* indoor and outdoor images. In section 7 we discuss the limitations and extensions of our approach and the possible application scenarios.

4 Description of our system

We start with 2-D images of a scene. No 3-D information of the scene is directly available, nor do we have any information about the relative positions of the cameras. Using just two images, we are able to generate images of the scene from different viewpoints. The sections below give a detailed description of our approach for handling polyhedral objects.

4.1 Extracting points and faces

For polyhedral objects, the "interesting" points for which correspondences need to be established are the vertices of polygons in the image. We first start by detecting edges on the scene images [2]. The edges are then approximated by linear segments [10]. These linear approximations are used to extract vertices. Faces are detected by finding closed chain of vertices.

In theory, the process of extracting points, lines and detecting faces and vertices is straightforward. In practice, when dealing with real images, imperfect edges are obtained because of noise, image quantization effects, shadows and poor scene illumination. Two typical edge images are shown in figure 4. These images were obtained by applying the Canny edge detector [2] on the images shown in figure 10(a). Many of the edges shown here do not correspond to physical edges of the objects. Consequently, some of the relevant vertices and faces might not be detected. Therefore, we have developed a simple user interface which allows the user to edit these imperfections. The user can add and correct the vertices and faces that are not detected.

We now have vertices and faces in both images. Next, we compute correspondences for these features. Note that this step is also performed off-line.

4.2 Searching for correspondences

To establish correspondences, we need to search for the matching feature of the first image in the second image. This 2-D search problem can be simplified into a 1-D search problem through the use of the epipolar geometry [7], [8]. This is explained in the illustration of figure 5. Two cameras with centers O_1 and O_2 are shown with their respective image planes. Let P and Q be any two points in 3-D space. Their projections are p_1, q_1 in the first image and p_2, q_2 in the second image. P, O_1 and O_2 form a plane known as the *epipolar plane*. This plane intersects the two image planes in lines l_1 and l_2 respectively. These lines are known as *epipolar lines*. As seen from the geometry of the scene, p_1 and p_2 are constrained to lie on lines l_1 and l_2 respectively. Thus, if p_1 is given and its corresponding point p_2 is to be determined, we need to search for it only along epipolar line l_2 in the second image. The same reasoning applies for finding the correspondence of q_1 in the other image. This epipolar geometry is captured in a transformation matrix called the *fundamental matrix* which maps one image plane onto the other. Algorithms exist in the computer vision literature to compute this fundamental matrix from eight initial correspondences [8] or even five initial correspondences, although more stable results are obtained with more points [7]. Since this step is performed off line between existing views, the initial correspondences to set up the fundamental matrix may be given interactively.

One can also see from the figure that the projection of O_2 in the first image plane lies on the intersection of the two epipolar line l_1 and m_1, which are the projections of lines PO_2 and QO_2 respectively. This point o_1 is known as the one of the two *epipoles* and will be used for the computation of the cross ratio in section 4.4. The other epipole o_2, the projection of O_1 in the second image plane, is obtained similarly. Once the fundamental matrix is computed between two initial images, we can associate for every point in the first image, an epipolar line in the other image. The corresponding point in the other image should lie on this line.

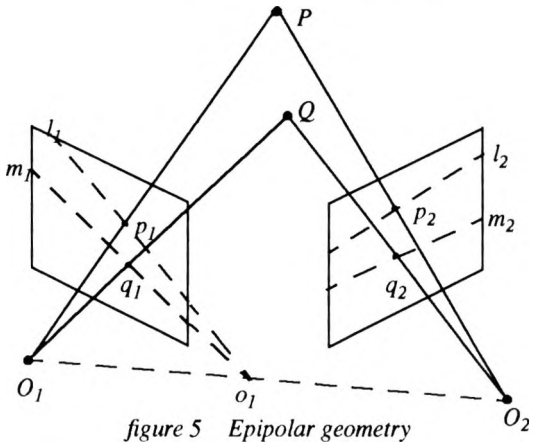


figure 5 Epipolar geometry

While searching for correspondences along the epipolar lines, exactly one, more than one, or no correspondences may be found. If exactly one vertex lies on the epipolar line, then we have found the corresponding point. If more than one vertex lies on the line, then we need local correlation to disambiguate the matches. If no vertex lies on the line, the face might have undergone a partial occlusion. The next section deals with such situations. It should be noted that, although theoretically straightforward, incorrect or no matches may be found in practice when working with real images. Therefore, in our current implementation, we have also developed an interface which allows the user to interactively correct an erroneous match if it occurs.

4.3 Dealing with occlusion

Occlusion of a vertex in one of the images may create a wrong match, or no match situation. Wrong matches are corrected interactively. In addition, for cases where there is no match found, we have developed a methodology to automatically handle occlusion as explained below with an example.

From figure 6, one can see that the darker face in view-1 gets occluded in view-2. Here the edge $A'B'$ is occluded in view-2 and shows up as $A'C'$. For point B in view-1, the epipolar line in view-2 intersects the corresponding edge $A'C'$ at B'. Since the corresponding point of B in view-2 lies on edge A'C' and the epipolar line, its position can be easily computed.

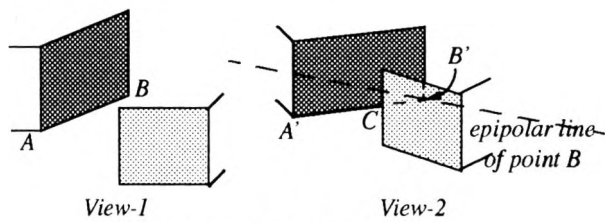


figure 6 Occlusion

4.4 Computing the cross ratio

Given any corresponding points in two images, we would like to predict where this point would appear in a new image from a different viewpoint. This can be accomplished with the cross ratio, since it is a projective invariant. For any point P, the cross ratio requires three other points collinear with P. We generate them using ideas similar to the one described in [16].

For simplicity of explanation, let us first assume that we have four non-coplanar points A, B, C and D in the 3-

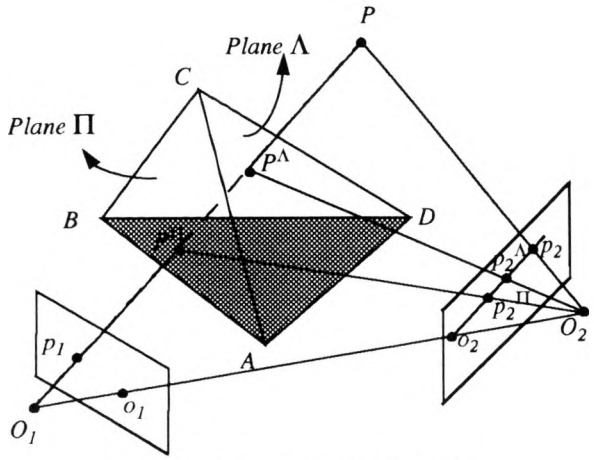


figure 7 Computation of cross ratio

D world, as shown in figure 7. To associate a cross ratio to a point P, we use the line defined by the center of the first camera O_1 and P. This line intersects the planes defined by A, B, C (plane Π) and B,C,D (plane Λ). Let us call these points of intersection P^Π and P^Λ respectively. These two points are different only if A, B, C, D are non-coplanar. From the four collinear points O_1, P^Π, P^Λ, and P, we can compute the cross ratio α_p.

In practice, however, we do not have access to these 3-D points A, B, C, D and P, but only to their projections in the two images. We show below how these can be used to compute projections of P^Π and P^Λ in the two initial images, and the cross ratio α_p.

In image 1, the points O_1, P^Π, P^Λ, P project to p_1, while in image 2 they project to o_2, p_2^Π, p_2^Λ, and p_2 respectively. The points A, B, C and D project to a_1, b_1, c_1 and d_1 in image 1, and a_2, b_2, c_2 and d_2 in image 2 respectively. o_1 and o_2 are the epipoles and can be computed as mentioned in section 4.1.

Given the corresponding projections of A, B, C, and D in the two images, we need to obtain the positions of p_2^Π and p_2^Λ, which can be computed as follows: The plane corresponding to image 1 can be projectively mapped to Π through a projective transformation. Similarly, Π can be projectively mapped onto image 2. Since projective transformations are transitive and invertible, there exists a projective transformation which maps the projection of points of Π on image 1 to the projection of points of Π on image 2. Therefore, if we can compute this projective transformation, we can obtain p_2^Π by applying this transformation to p_1. Computing a projective transformation requires four corresponding points in each image plane. Since we have to compute p_2^Π, we have to use correspondences of four points which are projections of points in the plane Π. The projections of A, B, C in the two images give three such points. To acquire a fourth point we make use of the following observation. Π intersects the line O_1O_2 at a point, say O'. O' thus lies on Π and projects onto o_1 and o_2 in the two image planes respectively. Hence the projections a_1, b_1, c_1, o_1 and a_2, b_2, c_2, o_2 can be used to compute this projective transform. Similarly using the projections of points b_1, c_1, d_1, o_1 and b_2, c_2, d_2, o_2, which are projection of points on Λ, a projective transformation can be computed which gives p_2^Λ when applied to p_1.

This explanation may appear confusing the first (few) time(s), as it involves intricate geometric construction, but is very straightforward to implement in practice. The central result is that, **for any point *P*, we can compute the cross ratio α_p, given the projection of the four points *A, B, C, D* in two images.** The issues related to the choice of these four points are discussed in section 7.

4.5 How to specify a new viewpoint

We want to use the results obtained in section 4.4 to generate a new view. As explained there, we need to use the projections of the four basis points A, B, C, and D to generate the new view. Although this is theoretically equivalent to another means of expressing a projective transformation, this specification is not intuitive at all. A more natural way to specify a new viewpoint would be to give the position and orientation of the image plane and the focal length. This, unfortunately, is not directly usable as we have no knowledge of the Euclidean 3-D transformation between the input views. To compute it would require a priori knowledge of the camera parameters. Furthermore, even with this information, its computation is difficult and unstable

The solution we propose is to generate an equivalent of the true depth of the four base points from the initial images, so that their projections can be computed given the intuitive representation of the new view. This is accomplished by computing the disparity of the four base points from their projections in the two initial images. Disparity is defined as the difference in image coordinates along the epipolar lines between matching points in these images, such as dp and dq in figure 8. As illustrated in the figure, disparity is inversely proportional to the actual depth [12] of the point relative to the camera.

In summary, we compute disparity for the four basis points A, B, C, and D from their projections in the two initial images. We specify any new viewpoint in a natural fashion, and infer the projection of the four basis points in this new viewpoint.

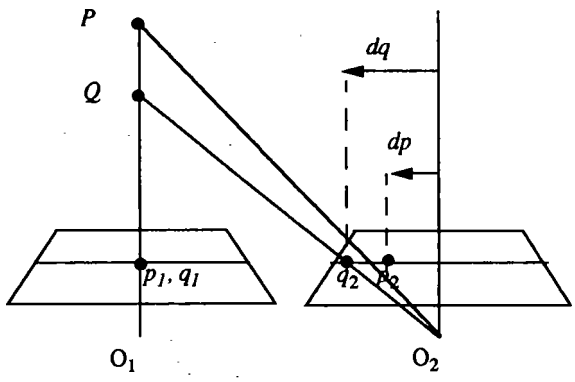


figure 8 Disparity and depth

4.6 Generating wireframe drawings

For a new camera position, we now have the projections of the four basis points in the new image. Cross ratios for all the other points in the scene have been computed from the two initial images using these four points basis points. For every point whose cross ratio is known, we thus can compute the location of the point in the new image using the method explained in section 4.4. The points are then linked to form wireframes. This process of generating wireframes for each new camera position require very few operations and hence wireframes can be generated at frame rate on a standard workstation.

4.7 Texture mapping

We have now generated the boundaries of faces in the new scene. Since we know which faces they correspond to in the two initial images, texture from these images can be mapped onto the new faces. We compute the areas of the faces in the two initial images. This help us in selecting the face with more texture information to map onto the face in the new image. Before mapping texture, the faces have to be ordered in a sequence corresponding to the way they occlude one another. Section 4.3 explained how to detected whether a face has become occluded. Texture mapping is first performed for faces which have become more occluded and then for the rest.

We first compute a projective transform which maps a face in one of the initial images to a face in the new image. This face is then broken down into triangles. Back projection using the computed projective transform, with bilinear interpolation is used to texture map each triangle of the face in the new image. Other efficient texture mapping techniques [3],[11] which are capable of creating more realistic and accurate results can also be used.

5 Experimental results

We first show results on a set of synthetic images. This sequence was chosen in order to make a quantitative comparison and demonstrate the accuracy of our approach. A sequence of two hundred images were generated using 3-D models. We wanted to recreate the entire sequence using only a few views of the rendered sequence. We chose six views showing different viewpoints of the scene, as shown in figure 9 (a). In this case, we had the exact locations of all the points in the rendered images and their correspondences. We used these correspondences in the six chosen views to compute the cross ratios of all the points as described in section 4.4. In figure 9(b) left, we show three rendered images from different viewpoints obtained from 3-D models. In figure 9(b) right we show the same views synthesized by using our method. For the first pair, the predicted wire frame superimposed on the rendered image is shown in figure 9(c). This shows that the view transfers are accurate. This was expected because the exact feature locations and correspondences were used. Figure 9(c) also shows an enhanced difference image for the first pair. This image was enhanced so the differences, which were below 20, could be visibly printed. Although the geometry is accurate, small differences exist because of artifacts in the bilinear interpolation.

In figure 10 we show our results on a real scene. Figure 10(a) gives the 2-D unregistered views used as input. Edge detection and feature extraction was performed on these images. 103 features points were automatically detected, amongst which 62 were manually rejected. Most of the discarded points consisted of texture features whose geometry can be captured or approximated by other points in the scene. Correspondences for 37 points were automatically obtained as described in section 4.1. We kept 32 correspondences to generate the viewpoints. Two more correspondences were interactively added for the completeness of the scene. Figure 10(b) shows three synthesized viewpoints generated using the above correspondences. Occlusion is handled well here as shown by the book on the table. To test the correctness of our approach for real images, a new image from a different viewpoint was captured. Figure 10(c) shows the predicted wireframe overlaid on the new image. Unlike the synthetic scenes, the positions of some of the predicted points are two or three pixels off because of localization errors in feature points. The right corners of the table and the PC, which are more erroneous are the ones which were manually added for completeness.

We applied our approach to a complex outdoor scene and were able to synthesize novel viewpoints for selected objects in the scene. In Figure 11(a) two such

outdoor views of a building are shown. One was taken from the roof of an adjacent parking structure, the other from ground level. Figure 11(b) shows synthesized images of the building from different viewpoints. Thus, using our approach, one can select objects from real images and put them into virtual environments to enhance realism.

6 Complexity Analysis

The first two stages, namely establishing correspondences between detected features in the two images and the computation of the cross ratio for each feature point, need to be performed only once, and are done off-line. Once the cross ratios are computed, the wireframe of any new image can be computed in real time. The complexity of this process is linearly proportional to the number of feature points. We can therefore generate an animated sequence of wireframes in real-time on a standard computing platform.

In our experiments, most of the burden for generating new images falls on the texture mapping stage, where triangles needed to be scanned. The choice of proper graphics hardware can overcome this burden.

7 Discussions and Applications

In this section, we discuss the limitations of our method and the possible enhancements that may improve performance of our implementation. We also highlight potential applications of our approach.

The method has the following limitations:

- The choice of the four non-coplanar points plays a role in the accuracy of the generated image. Different choices of points will give slightly different results. But once a set of points which approximate the cross ratios well is determined, the predictions remain accurate over a broad set of variations.
- The locations of the epipoles o_1 and o_2 (see figure 7) also affect the accuracy of the generated image. If the epipoles are far away, the cross ratio becomes prone to error and consequently the generated viewpoints are inaccurate. This happens when the image planes of the two initial images are coplanar.
- We perform texture mapping on the new image. Since we have no information of the location of viewpoint and the 3D geometry of the scene, specular reflections cannot be dealt with.

The accuracy of the image generated by using our method relies on the precision of the image feature location and the correctness of feature correspondences. Three images may be used to increase the reliability of finding correspondences. Given three images, as shown in figure 9, we can compute the fundamental matrix for each set of two images. If p_2 in image 2 and p_3 in image 3 are the hypothesized correspondences of p_1 in image 1, then p_3 in image 3 lies at the intersection of epipolar lines l_3 and l'_3 in image 3. This constraint can be used to verify the correspondences p_1 and p_2.

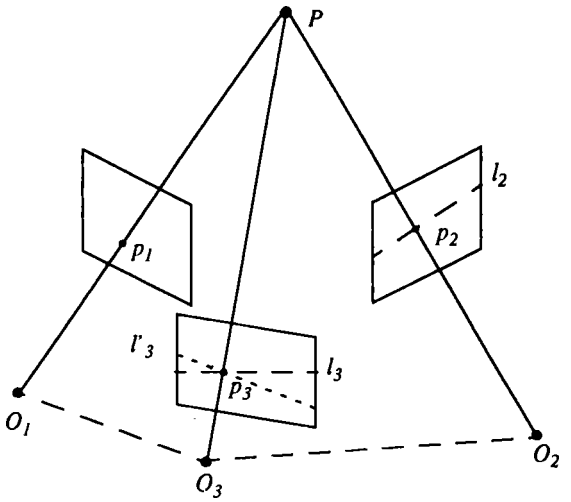


figure 9 Using three images

Also of interest are recent results, where a stable computation of trilinear tensors to establish correspondences is given for cases where three images are available [20].

Our method can be used in applications where 3-D models are impractical or impossible to obtain, such as complex real interior environments, or remote or dangerous environments where it is impossible to obtain models for. If a few views can be obtained, our approach could provide smooth animations.

Our method can also be used to quickly synthesize images of complex modelled environments. Virtual reality walkthroughs can be done using a small number of precomputed high quality images with radiosity lighting. For models with many polygons, our approach can offer a performance advantage.

8 Conclusion

We have developed an approach to generate new images without going through the tedious steps of obtaining 3-D models and camera positions. This work is based on firm mathematical principles and uses mature Computer Vision techniques. Even though the whole process is not fully automated, human intervention is required to augment and correct automatic steps, namely point out hard to detect features and refine correspondences. These steps are performed off line and only for a sparse set of existing views. It is then possible to generate wireframes

in real time and to texture map the new scene for an accurate and realistic look.

Acknowledgment

We thank Dr. Ulrich Neumann for stimulating discussions on this work and his help in organizing this paper. We also thank Ms. Yolanda W.H. Chen for her effort in implementing some of the ideas.

References

[1] Ali J. Azarbayejani, Alex Pentland - *Recursive Estimation for CAD Model Recovery.* Proceedings of the second IEEE CAD based Vision Workshop., 1994. pages 90-97.

[2] J. Canny, *A Computational Approach to Edge Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8, 1986, pages 679-698.

[3] E. Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces* (Ph. D. Thesis), Department of Computer Science, University of Utah, Tech. Report UTEC-CSc-74-133, December 1974.

[4] S. Eric Chen and Lance Williams - *View Interpolation for Image Synthesis.* SIGGRAPH 93 Proceedings. pages 279 - 288.

[5] S. Eric Chen. QuickTime VR - An Image-Based Approach to Virtual Environment Navigation. SIGGRAPH 95 Proceedings.

[6] Y. Chen and G. Medioni, *Object modelling by registration of multiple range images*, Image and Vision Computing, 10, 1992, pages 145-155.

[7] R.Deriche, Z. Zhang, Q.T. Luong, O. Faugeras. *Robust recovery of the epipolar geometry for an uncalibrated stereo rig.*European Conference on Computer Vision, vol I, pages 567-576, 1994.

[8] O. Faugeras, N. Ayache, and B. Faverjon. *Building visual maps by combining noisy stereo measurements.* In Proceedings of IEEE Conference on Robotics and Automation, Apr 1986.

[9] N. Greene. *Environment Mapping and other Other Applications of World Projections.* IEEE CG &A, Vol 6, No. 11, November 1986.

[10] P. Havaldar, G. Medioni and F. Stein. *Extraction of groups for recognition.* Proceedings of the European Conference on Computer Vision, volume I, pages 251-261, Stolkholm, 1994.

[11] P.S Heckbert. *Fundamentals of Texture Mapping and Image Warping*, Masters Thesis, Dept. of EECS, UCB, Technical Report No. UCB/CSD 89/516, June 1989.

[12] B.K.P. Horn, Robot Vision, MIT Press, 1986.

[13] M.S. Lee and G. Medioni, *Structure and Motion from a Sparse Set of Views*, Proc. of IEEE International Symposium on Computer Vision, pages 73-78, Coral Gables, Florida, Nov 95.

[14] L. McMillan and G. Bishop. *Plenoptic Modeling: An Image-Based Rendering System.* SIGGRAPH 95 Proceedings.

[15] J. Mundy and A. Zisserman. Appendix - *Projective Geometry for Machine Vision.* In J. Mundy and A. Zissserman, editors, Geometric Invariance in Computer Vision. MIT Press, Cambridge, 1992

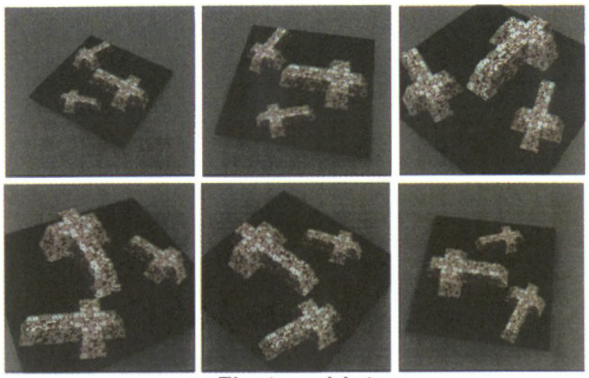
[16] A. Shashua. *Projective Depth: A Geometric Invariant for 3D Reconstruction From Two Perspective Orthographic Views and for Visual Recognition.* Proceedings of the International Conference on Computer Vision, pages 583-590, Berlin, Germany, May 1993.

[17] Tomasi and Kanade. *Shape and motion from image streams under orthography*: a factorization method. International Journal of Computer Vision, vol. 9, no. 2, pages 137-154, November 1992.

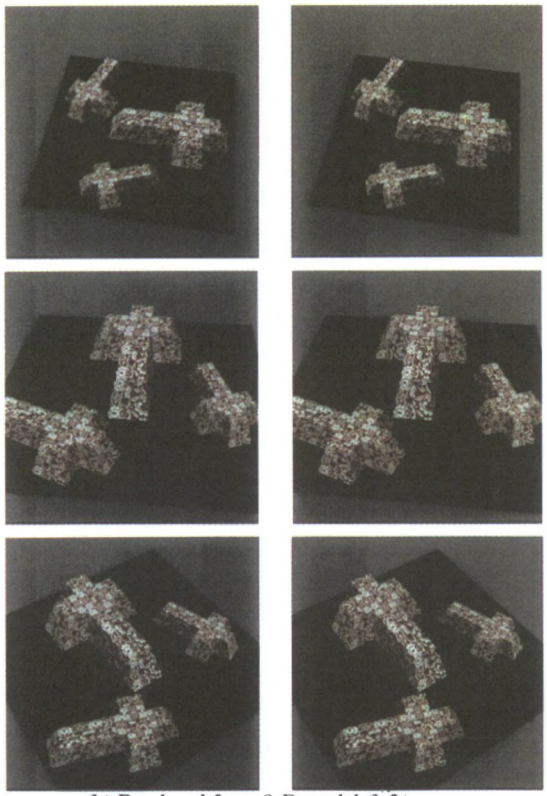
[18] L. Teodosio and W. Bender. *Salient Video Stills: Content and Context Preserved.* Proceedings of First ACM International Conference on Multimedia, pages 39-46, Anaheim, California, Aug 93.

[19] L. Teodosia and M. Mills. *Panoramic Overviews For Navigating Real-World Scenes.* Proceedings of First ACM International Conference on Multimedia, pages 359-364, Anaheim, California, Aug 93.

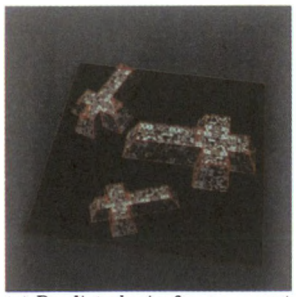
[20] M. Werman and A. Shashua. *Trilinearity of Three Perspective Views and its Associated Tensor.* Proceeding of the International Conference on Computer Vision, pages 920-925, Boston 1995.



(a) The six model views



(b) Rendered from 3-D model (left),
Synthesized from model views (right)



(c) Predicted wireframe superimposed on rendered view (left),
Difference image of first pair (right).

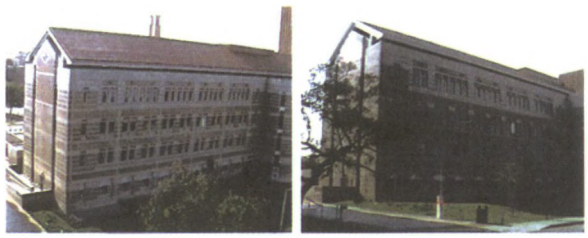
Figure 9 Synthetic Scene.

(a) Two views of an indoor scene.

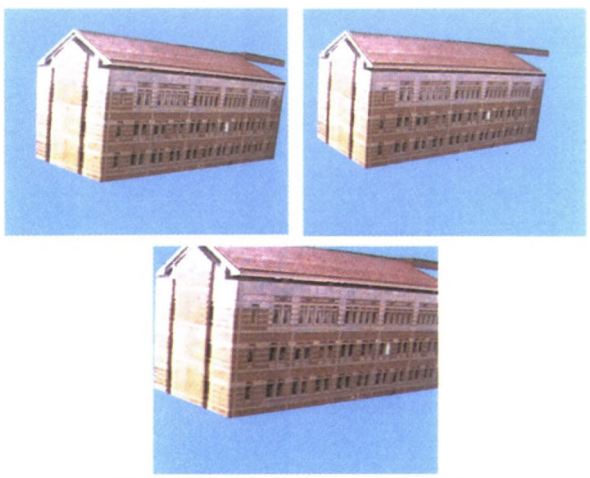
(b) Synthesized views of the indoor scene.

(c) Predicted wireframe superimposed on a new view.

Figure 10 Indoor Scene.



(a) Two views of an outdoor scene.



(b) Synthesized views of the building.

Figure 11 Outdoor Scene.