MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 1592 November, 1996
C.B.C.L. Paper No. 140

Image Based Rendering Using Algebraic Techniques

Theodoros Evgeniou

This publication can be retrieved by anonymous ftp to publications.ai.mit.edu.

Abstract

This paper presents an image-based rendering system using algebraic relations between different views of an object. The system uses pictures of an object taken from known positions. Given three such images it can generate "virtual" ones as the object would look from any position near the ones that the two input images were taken from. The extrapolation from the example images can be more than 60 degrees of rotation. The system is based on the trilinear constraints that bind any three views of an object. As a side result, we propose two new methods for camera calibration. We developed and used one of them.

We implemented the system and tested it on real images of objects and faces. We also show experimentally that even when only two images taken from unknown positions are given, the system can be used to render the object from other viewpoints as long as we have a good estimate of the internal parameters of the camera used and we are able to find good correspondence between the example images.

In addition, we present the relation between these algebraic constraints and a factorization method for shape and motion estimation. As a result we propose a method for motion estimation in the special case of orthographic projection.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for this research was provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00000-00-A-0000. THE AUTHOR was also supported by NSF-0000000000.

1 Introduction

1.1 Motivation and Goal

In recent years image-based rendering systems have been an emerging subject of research within the computer graphics and computer vision communities. Applications range from generation of virtual reality effects, to object recognition by matching real images to virtual ones. The underlying idea of these systems is that the representation of the images is based purely on photometric observations and not on any indirect models. Traditionally objects and scenes have been approximated using either geometric models such as mathematical descriptions of the boundaries of the objects, or physics-based models, or, more typically, 3-D models. However the limitations of these approaches are apparent when one tries to render realistic images, since the complexity of the real-world does not permit accurate and efficient modeling.

Geometric modeling has the problem that it cannot be used for complicated surfaces such as faces or other flexible objects. Although it might work satisfactorily for some simple objects and scenes, it might not for other complicated ones. Similar problems arise with physics-based models. The complexity of such models makes them intractable. Furthermore, building 3-D models of objects, although it can work efficiently for simple planar surfaces, is often inefficient for more complicated ones. There is again a major trade off between the complexity of the model, ie. how well to approximate a non-planar surface using small patches, and its realism. Achieving high visual realism requires in many cases high complexity which leads to inefficient rendering.

Given these, the need for image based rendering systems is clear. When the representation of an objects or a scenery is based on its actual images the realism that can be achieved is higher than that when some approximation model is used. Moreover, the computation required for rendering using these systems is independent of the complexity of the scene and can be done efficiently enough for real-time use [4].

There are two new approaches in image based rendering. One is using morphing techniques, while the other is using algebraic techniques. Image morphing is the simultaneous interpolation of shape and texture. In this group, noticeable work has been the one by Beymer, Shashua, and Poggio [3] partly based on the ideas presented in [19]. They showed that morphing based on using correspondences between input images is applicable for generating virtual images of faces with changes in pose and expressions such as smiles. Other morphing techniques are described in [31], [21] and [5]. Despite their differences they are all based on the idea of interpolation, therefore they do not give much freedom for virtual camera motion, namely extrapolation.

The other new approach in image based rendering is to use "natural" algebraic relations that exist between different views of the objects in order to generate virtual images. The main difference from the aforementioned approach is that this one is only suitable for rigid transformations but also enables significant extrapolation from the example images. There has been a significant amount of work very recently done in this area ([11], [12], [13], [17], [18], [24]). Some of these approaches are presented briefly in the next section. Noticeable has been the work of [18] on rendering sceneries. Using cylindrical projections of a scenery taken from known positions, they manage to reconstruct the scenery from arbitrary positions by reconstructing the plenoptic function describing it. The plenoptic function is a parameterized function describing everything that is visible from a given point in space.

The goal of this paper is to develop a method similar to the one in [18] but which can be used for objects. Instead of having cylindrical images of a scenery, we are given pictures of an object taken from known positions around it. Then, using these images we want to be able to generate virtual views of the object as they would look from arbitrary positions. We want the system to be robust and to allow for arbitrary camera motion, ie zoom and in place rotation, as well as considerable extrapolation from the example images. The long term goal is to combine the system with existing ones for non-rigid transformations of objects (ie. face expressions) [8] in order to build a complete tool that can be used for rendering images of objects under any rigid or flexible transformation.

Moreover, part of the paper is to further study algebraic results used for computer vision. The goal is to find the hidden connection between the algebraic relations among images of an object and the estimation of the objects shape and motion. As a result of this study, a simple method for motion estimation of an object undergoing rigid motion with more than two images is developed.

1.2 Contribution of the paper

Summarizing, the contributions of the research described in this paper are briefly the following:

1. We developed a new method for image-based rendering and we tested it on real images.

2. We developed a new simple method for camera calibration.

3. We showed the relation between algebraic constraints connecting images and the shape and motion factorization method described in [29].

4. We proposed a simple method for motion estimation in the special case of orthographic projection and suggested possible extensions for the general perspective case.

1.3 Layout of the paper

The paper is organized as follows:

- In section 2 the necessary background as well as a brief description of related previous work and ideas are presented.
- In section 3 our method and the problems that our systems had to solve are presented in detail.
- Section 4 discusses the implementation of the system and shows results.
- In section 5 we refer to a different but related problem. The problem of recovering shape and motion using the ideas of the rest of the paper. A theoretical result is presented and a method for shape and motion estimation is proposed.
- Finally, in section 6 we state conclusions and suggestions for further work.

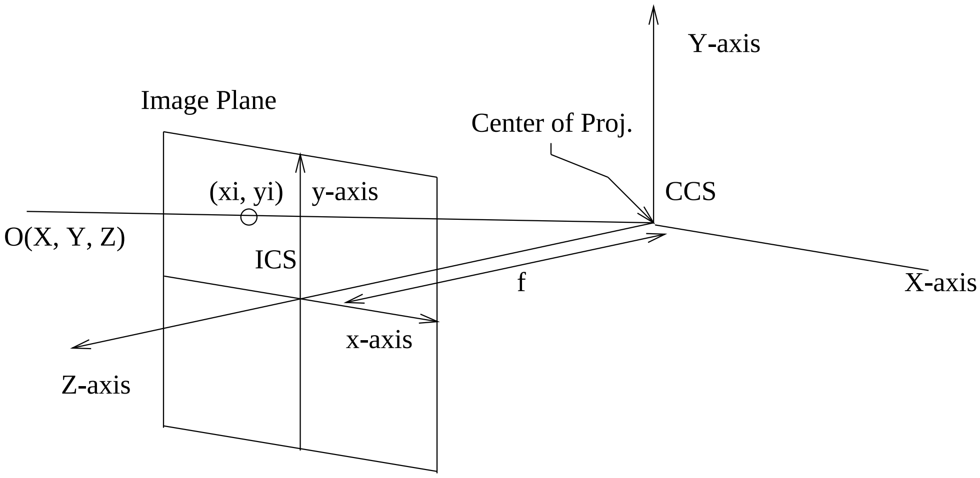

Figure 1: The Pinhole Camera Model

2 Background and Related Previous Work

2.1 Background

Before going to the main part of the paper we need to provide the necessary background. We present the basic ideas of projective geometry and its use in computer graphics.

2.1.1 Camera Model and Projective Geometry

The camera model we use is that of a *pinhole camera* described in figure 1. Objects are viewed under *perspective projection*. The center of projection is the *optical center* of the camera, which we also define to be the origin of our camera coordinate system (CCS) - also called the *standard coordinate system of the camera*. The image plane - or *retinal plane* - is parallel to the XY-plane of our CCS, and at distance f from the optical center, called the *focal length* of the camera. We define an image coordinate system (ICS) such that the origin is the projection of the optical center on the retinal plane. If we define the X, Y axis of the ICS to be parallel to the X and Y axis of the CCS, respectively (see figure 1), then we can easily see that the image coordinates of the projection of an object point $O = (X, Y, Z)$ are given by:

$$\begin{aligned} x_i &= -f \cdot X/Z \\ y_i &= -f \cdot Y/Z \end{aligned} \tag{1}$$

Clearly any point on the line through O and the center of projection has the same image.

It seems that when one wants to compute image coordinates using the aforementioned approach one has to deal with ratios. Using ratios is not very computationally efficient and also makes the mathematics complicated. Fortunately there is a way to avoid these ratios. The traditional solution is to use *Projective Geometry* instead of the Euclidean one. After all, images are generated using *central projection.*

In projective geometry the 3-dimensional Euclidean space R^3 is considered to be a subspace of the 4-dimensional Projective space P^4. A point in P^4 is described by its coordinates up to a scale factor. This means that, ie. $(X, Y, Z, W) \simeq (2X, 2Y, 2Z, 2W)$. There is a natural injective mapping from R^3 to P^4 defined by:
$M : R^3 \mapsto P^4$ such that $(X, Y, Z) \mapsto (X, Y, Z, 1)$.
The remaining points of P^4, which are the ones of the form $(X, Y, Z, 0)$, are called *ideal points.* In the same way we have a similar relation between the 2-dimensional Euclidean space and the 3-dimensional Projective Space. A point (X, Y) in R^2 corresponds to $(X, Y, 1)$ in P^3 and points $(X, Y, 0)$ are the ideal points of P^3.

By modeling our spaces as Projective instead of Euclidean we gain the following:

1. First, we notice that points that are on the plane parallel to the image plane through the center of projection - called the *focal plane* - have no images (see figure 1). Modeling the image plane as a Euclidean 2-Dimensional plane is therefore not suitable since we cannot account for the "images" of these points. However, if we model it as a 3-dimensional Projective space, then we can define the images of these object points to be the ideal points of the retinal plane. This way all object points are treated the same.

2. There is a very important practical reason for using projective geometry. As stated above, using Euclidean geometry, the coordinates of a point on the image plane are given as ratios (equation 1). However, in the corresponding projective spaces, we have that the coordinates of an object point $(X, Y, Z, 1)$ (we can set the fourth coordinate to 1 so that we preserve the "natural" correspondence with Euclidean coordinates defined above. For the moment we also don't deal with the object points that have no image, therefore we can assume that z_i below is not zero) on the image plane are given simply by:

$$\begin{bmatrix} x_i/z_i \\ y_i/z_i \\ 1 \end{bmatrix} \simeq \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \begin{bmatrix} -f \cdot X \\ -f \cdot Y \\ Z \end{bmatrix} \tag{2}$$

Therefore we can use linear algebra and avoid the nonlinearities introduced by the aforementioned ratios.

If the coordinates of the object point are given relative to another coordinate system, which we call World Coordinate System (WCS), then, if

$$K = \begin{bmatrix} R & T \\ 0_3^T & 1 \end{bmatrix} \tag{3}$$

is the matrix of transformation between the WCS and the CCS, the coordinates of the object point in the CCS are given by:

$$\begin{bmatrix} R & T \\ 0_3^T & 1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{4}$$

therefore its projection on the image plane is given by:

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} R & T \\ 0_3^T & 1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = M \cdot D \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5}$$

The product $P = M \cdot D$ is called the *camera matrix.*

The matrix:

$$D = \begin{bmatrix} R & T \\ 0_3^T & 1 \end{bmatrix} \tag{6}$$

is called matrix of the *external parameters* of the camera. It describes the position of the camera relative to the WCS.

The matrix:

$$M = \begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{7}$$

is called matrix of the *internal - or intrinsic - parameters* of the camera. However, the matrix in equation 7 is a special case of the general form. Apart from the focal length, there are five other parameters that depend on the camera. First notice that in reality the line connecting the center of projection and the center of the image is not always perpendicular to the image plane. Therefore, we have to move the image coordinate system so that its origin is the projection of this perpendicular. This adjustment gives two more parameters: the coordinates of the actual origin of the image coordinate system in terms of the system described above. Furthermore, there is typically some scale distortion so that the unit lengths of the image axis are unknown - not the same as the corresponding ones in the camera coordinate system. This scale distortion gives two more factors - one for each axis. Finally, there is a last factor that gives the angle between the axis - in practice the X and Y axis may not be orthogonal. When one takes into account all these, one can easily find that the general form of the matrix of the intrinsic parameters of the camera is:

$$M = \begin{bmatrix} -fk_u & 0 & u_0 & 0 \\ 0 & -fk_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{8}$$

where k_u and k_v are the scaling factors for the X and Y axis respectively, and u_0, v_0 are the image coordinates of the actual origin of the ICS relative to the one we assumed at the beginning. For more information on the derivation of this matrix the reader can refer to [9].

Estimating the intrinsic parameters of the camera is called *calibration* of the camera. Various methods for doing camera calibration have been developed ([10], [25]). In the next section we describe a method that we developed and used.

Before changing topic, we should mention here a very usual approximation to the perspective projection model described above. Sometimes we assume that the Z coordinates of the object points are almost constant. This is the case when, for example, the objects are far relative to the focal length, therefore changes in the Z coordinates are negligible. In this case we can write the image coordinates in equation 1 as:

$$x_i = -f \cdot X/Z_0 = f' \cdot X \tag{9}$$

$$y_i = -f \cdot Y/Z_0 = f' \cdot Y \tag{10}$$

When we use this approximation we say that the image is generated under *orthographic projection*. In this case it is easy to see that the camera matrix is given by: (assuming f' is 1, all other internal parameters are 0, and the external parameters matrix is the identity one)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \tag{11}$$

so the image coordinates using equation 5 (in projective spaces) are given by $(x_i, y_i, z_i) = (X, Y, 1)$. This is a simplification that we will use in section 5.

Having this background we can now examine an algebraic relation between images that has been traditionally used in systems. The interested reader can refer to [9] and [16] for more information on projective geometry and its use in computer graphics.

2.1.2 Epipolar Geometry

Suppose we are given two images of the same object point O taken from two different known positions as shown in figure 2. Clearly, if we know the projection of the point on image 1, then the projection of the point on image 2 is constrained: it has to lie on line l_2 as shown in figure 2. This line is the intersection of the second image plane with the plane through O and the optical centers C_1 and C_2 of the two cameras. Symmetrically, its projection on image 1 has to lie on line l_1.

If p_1 and p_2 are the image coordinates of the point in the two images, we must have that:

$$p_1 = P1 \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{12}$$

and

$$p_2 = P2 \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{13}$$

where P_1 and P_2 are the camera matrices as described above. If P_{11}, P_{21} are the top 3x3 submatrices of P_1 and P_2 respectively, then it is easy to see [9] that we must have:

$$p_2 = P_{21} \cdot P_{11}^{-1} \cdot p_1 \tag{14}$$

Moreover, if e_2 is the projection of C_1 on the second image and E_2 is the $3x3$ antisymmetric matrix representing the dot-product with e_2 (that is, $E_2 a = e_2 \times a$) then $E_2 p_2$ is a vector perpendicular to line l_2. This means that $p_2^T E_2 p_2 = 0$ which, using equation 14, gives that:

$$\begin{gathered} p_2^T \cdot E_2 \cdot P_{21} \cdot P_{11}^{-1} \cdot p_1 = 0 \Rightarrow \\ \Rightarrow p_2^T \cdot F \cdot p_1 = 0 \end{gathered} \tag{15}$$

where $F = E_2 \cdot P_{21} \cdot P_{11}^{-1}$.

The matrix F in equation 15 is called the *Fundamental Matrix* between images 1 and 2 and when we know it we say that we have *weakly-calibrated* the camera. Lines l_1 and l_2 are called *epipolar lines*. Since all epipolar lines are on a plane that goes through the two optical centers all these lines go through the same point on each image (e_1 and e_2 in figure 2): the projection of the optical center of the camera for the other image. This point is called the *epipole* - one for each of the two images.

This equation holds for any pair of corresponding points and it defines the *epipolar geometry* between the two images. Recovering this geometry has been the subject of research

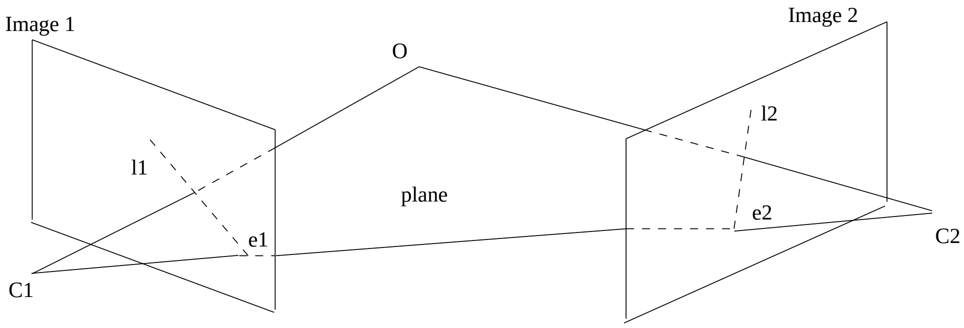

Figure 2: The epipolar geometry.

the recent years. For more information the reader can see [7], [9], [17]. We will use this geometry in the next section when we deal with the problem of occlusions.

2.2 Related Previous Work

The first results on algebraic relations between images of the same object appear in the work of Ullman and Barsi described in [30]. They first showed that under the special case of orthographic projection one can get linear relations between the image coordinates of an object point seen in three images. Later it was shown in [28] that "1.5" example views are enough to predict new images of an object (the "1.5 view theorem"). These results implied that in the special case of orthographic projection new views of an object can be generated as *linear combinations of example views.*

Later these results were generalized for the perspective case [23], [24]. However, in this case the relations become trilinear, and the number of parameters needed increases. These "new" relations were first tested on images in [22], [24]. The results of those experiments were encouraging ([22], [24]). Based on that work, a method for image-based rendering was developed in [1]. The method we develop is close to the one described in [1] developed independently and at the same time as this work.

Another approach in image-based rendering has been to use the epipolar geometry described in the previous section. Clearly one can use the epipolar geometry between views in order to predict where a point should project in a new view. For example, if we knew the epipolar geometry between view 1 and 3, and between view 2 and 3, then we could predict view 3 from views 1 and 2 by projecting a point of images 1 and 2 on the point of intersection of the two epipolar lines in image 3.

Using this idea [17] describes a method for reprojection in the weakly-calibrated case. The method is based on finding intersections of lines as described above. However, finding intersections of lines is noisy and in the case of epipolar lines in some degenerate cases impossible [17]. Therefore systems based on this approach were sensitive to noise and did not provide a wide range of virtual views. This were the main reasons we avoided such an approach.

Given this background we are now ready to describe our system.

3 Reprojection Using Trilinearity Constraints

3.1 The Method

We want to generate new views of an object given a set of example images. Through this discussion we assume that all views of the object are taken using the same camera, therefore the internal parameters do not change from image to image. We set the world coordinate system to be the standard coordinate system of camera 0. Let $\{I_0, I_1, ...I_n\}$ be the n reference views of an object, and P_i be the camera matrix for I_i. That is, if M are the coordinates of a point in the scene in the standard coordinate system of camera i, then the projection of the point on image i has image coordinates $P_i \cdot M$.

Define K_i to be a matrix that depends on the external parameters of camera i:

$$K_i = \begin{bmatrix} R_i & T_i \\ 0_3^T & 1 \end{bmatrix}^{-1} \tag{16}$$

where R_i is the rotation matrix of camera i relative to camera 0, and T_i is the translation vector between the position of camera i and camera 0. Clearly $R_0 = I_{3x3}$, and $T_0 = 0_{3x1}$. Then, if P_i is the camera matrix for image i, we get that $P_i = P_0 K_i$, since, if M are the coordinates of a point in the scene relative to the world coordinate system, then the coordinates of the same point relative to the standard coordinate system of camera i are given by $K_i M$. The coordinate system and the whole camera just moved, so, it is as if we just moved the point to a new position - which explains the inversion of the position matrix. Therefore, if x and x' are the images of M in the two views, then $x = P_0 M$ and $x' = P_0(K_i M) = (P_0 K_i)M = P_i M$. Matrix P_0 depends only on the internal parameters of our camera. So, if we estimate P_0, we can get the camera matrix for any position we want.

Suppose now that we want to generate the image of the object as it would be seen from a position which is related to the origin of our world coordinate system by the matrix:

$$K = \begin{bmatrix} R & T \\ 0_3^T & 1 \end{bmatrix}^{-1} \tag{17}$$

We get this view in two steps. First we have to find pairs of reference views whose matrices K_i^{-1} are "close" to K^{-1}. Close matrices in this case are the matrices of the reference cameras that are closer to the virtual one. However, for simplicity, in our case we used the system with only two reference images, therefore we did not have to do this step. Having a pair of reference images I_i, I_j, we can get the new image coordinates of a point appearing in both reference views using the trilinearity constrains described in [24]. The idea is that if a point appears in three images with image coordinates (x_1, y_1), (x_2, y_2), (x_3, y_3), then, as one would expect because of the low dimensionality of our space, the three pairs of image coordinates have to satisfy specific algebraic relations. Therefore, if we know the relations and two of the pairs, we are able to find the third pair of image coordinates. Different ways of getting these relations are described in [11] and in [13]. Moreover, in the special case of orthographic projection these relations become linear as first found in [30]. We show this in the Appendix.

For computational purposes we use the trilinear relations as derived in [11] that we include in the Appendix. Here we present only one of those equations. Let the rows of P_i be 1, 2, 3, the rows of P_j be 4, 5, 6, and the rows of the new matrix $P_{new} = P_0K$ be 7, 8, 9. Let a point X in the scene project on I_i, I_j and the new image on $p_i = [x, y]$, $p_j = [x', y']$ and $p_{new} = [x'', y'']$ respectively. Then the following equations hold:

$$\begin{aligned} x'' &= \frac{(xx', -x, yx', -y, x', -1) \cdot ([2367], [2347], [3167], [3147], [1267], [1247])}{(xx', -x, yx', -y, x', -1) \cdot ([2369], [2349], [3169], [3149], [1269], [1249])} \\ y'' &= \frac{(xy', -x, yy', -y, y', -1) \cdot ([2368], [2358], [3168], [3158], [1268], [1258])}{(xy', -x, yy', -y, y', -1) \cdot ([2369], [2359], [3169], [3159], [1269], [1259])}, \end{aligned} \tag{18}$$

where $\cdot$ stands for dot product and [1234] represents the determinant of matrix [1234].

So, once we compute the new camera matrix (given P_0 and the new external parameters) we can predict the position of any point of the reference images in the new image.

3.2 Important Issues

3.2.1 Finding Correspondences Between the Reference Images

It is clear from the aforementioned that once we are given two reference images it is important to find the pairs of image points p_i, p_j mentioned above that correspond to the same object point. Finding these correspondences is a very critical step as one can easily see.

There are many ways to define correspondence between two images. In this paper, a *dense*, *pixel-wise* correspondence between two images is defined: for every pixel in image A, we associate a flow vector that refers to the corresponding pixel in image B. The flow vectors are relative to the current pixel position, so for a pixel in image A at position (i, j), the corresponding pixel in image B lies at position $(i + \Delta x(i, j), j + \Delta y(i, j))$, where Δx and Δy are arrays that contain the x and y components of the flows, respectively. Δx and Δy are the same size as the images A and B.

There are also many ways to obtain such a dense, pixel-wise correspondence between the example images. We use the optical flow algorithms developed by Bergen and Hingorani [2]. For details the reader can refer to that source.

3.2.2 Camera Calibration

Another crucial step is the calibration of the camera. As we see, the image coordinates of a point in the new view depend directly on the camera matrices and therefore on the internal parameters of the camera (equation 18). It is important, therefore, to have a good estimate of these parameters. We propose two ways for camera calibration.

The first way is to use directly the tensor relating the three example images. In the first step one can compute this tensor by finding corresponding points in the three images and solving the trilinear equations for the unknown coefficients. Since every set of corresponding points provides four independent trilinear equations, 7 points are enough to compute the tensor. However, least square estimates can be computed using more points. A robust method to compute the tensor relating three images is presented in [1].

Once the tensor is computed the internal parameters of the camera can be found iteratively so that given the matrices of external parameters the computed tensor for given internal parameters matches the actual tensor computed in the first step. Notice, however, that the relation between the tensor and the internal parameters is not linear (determinants in 18). So one would expect many solutions. However, if a good initial guess is given (typically focal length 500 and principal point at the center of the image) one would expect the method to converge to the correct solution.

The second method for camera calibration that we propose and we also implemented is as follows.
Given three images of the object, I_0, I_1 and I_2, taken from known positions, we use two of them, say I_0 and I_1, to generate the third one, I_{2new} using our method for any given set of internal parameters. As in the previous method, we start with a good initial guess and we determine the actual internal parameters iteratively so that the "distance" between predicted I_{2new} and actual I_2 is minimized. For better results we use only the middle third of the image. There are various ways to measure the distance between two images. In this paper we measure it as follows:
For each pixel in the middle third of I_0 and it's corresponding pixels in I_1 and I_2 we compute the predicted corresponding pixel in I_{2new} and we get the euclidean distance between the predicted pixel and the actual one in I_2 - assuming we have correct correspondence between I_0 and I_2. We do this for all pixels in the middle third of I_0 and we sum the distances.
The minimization is done using Powells' multivariable minimization method [20]. In figure 3 we show the three example images and the predicted third one. The initial guess of the

matrix of the internal parameters was

$$\begin{bmatrix} 500 & 0 & 115 & 0 \\ 0 & 500 & 210 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{19}$$

and the computed one was

$$\begin{bmatrix} 1099 & 0 & 241 & 0 \\ 0 & 1458 & 285 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{20}$$

3.2.3 Noise in the External Parameters

Our method assumes that the positions of the camera for the example images are known accurately. Clearly, in reality, it is difficult to know exactly where the camera is located. Therefore we should expect some noise.

In this paper we did not account for this noise. We simply assumed the locations are accurately known. However, when we tested the program by adding noise to the locations of the example images, we found that it is not sensitive to such error. Instead, we found that the calibration of the camera was giving different internal parameters in each case. However we did not examine this further. It is still an open question how errors in one set of parameters influences the estimation of the other.

In the next section experimental results regarding noise in external parameters are discussed further.

3.2.4 Solving for occlusion

Clearly not all points appearing in the reference views will appear in the new image. Moreover, not all points appear in both reference images. Generally detecting these occlusions is an unsolved problem. In this paper we used a heuristic similar to the one used in [31]. This heuristic however solves only some occlusion cases. Specifically, in the case that two pixels are reprojected onto the same pixel in the new image, we choose the one that is closer to the epipole in the first example corresponding to the new camera position. We estimate the position of the epipole using again the camera matrices directly. By [11], the location of the epipole in example one should be (in terms of the camera matrices as described in the previous section):

$$epipole_{1new} = ([1789], [2789], [3789]) \tag{21}$$

Figure 3: Top: Example Images I_0 and I_1. Bottom: Example Image I_2 and Predicted Image I_{2new} after 16 iterations.

where, like in equation 18, the rows of the first camera matrix are 1, 2, 3, the rows of the new camera matrix are 7, 8, 9, and $[abcd]$ represents the determinant of the $4x4$ matrix whose rows are rows a, b, c, d.

However this method does not handle occlusions such as the ones between the example images ([31]). The visibility problem is very important in many cases (depending on the object), and further research needs to be done.

4 Implementation and Experimental Results

We implemented the method as described in the previous section. However, experimentally we found out several sources of noise. In this paper we did not study the exact reasons of these problems. Below we present the problems, we suggest reasons for them, and we describe the way we solved them in our program.

4.1 Experimental Issues

4.1.1 Noise Due to the Choice of the Trilinear Equations

Experimentally we noticed that the program was giving different results when we were using different trilinear relations (as shown in the Appendix, there are 9 such relations). Specifically, when our example images were only from the XZ-plane, the system would not be able to generate views outside this plane when it used the relations described in the previous section. If instead we would use the following one for the y-coordinates:

$$y'' = \frac{(xy', yy', y', -x, -y, -1) \cdot ([2367], [3167], [1267], [2357], [3157], [1257])}{(xy', yy', y', -x, -y, -1) \cdot ([2369], [3169], [1269], [2359], [3159], [1259])} \tag{22}$$

it would work better. We postulate that the reason for this was that when using the equations of the previous section we try to estimate the y-coordinates of an image point using mainly the almost negligible changes in the y-coordinates between our examples. Therefore very small errors in the y-coordinates of the points in the example images - due to the fact that we use digital images - lead to big errors in the new y-coordinates. If however one uses the equations above, one uses the larger changes in the x-coordinates of the examples to estimate the new y-coordinates. A way to overcome this problem is to make the program so that different trilinearity equations are used depending on the position of the new virtual camera. So, for example, a rule of thumb could be that if the virtual camera is "close" to the line connecting the cameras for the two example images we use equations 18, while in case the virtual camera is "far" from that line, we use equation 22.

4.1.2 Noise Due to Holes

It is unavoidable to have some noise which leads to overlapping pixels in some places of the new image, and therefore to holes in other. Moreover, in case that we zoom in, clearly we will have to deal with holes due to lack of information.

Initially we solved the problem of hole filling like in [8]. Once we generate the new image we fill the holes by interpolating the pixel values of the edges of the holes along the x-axis or the y-axis. However this approach was not satisfactory.

In the final version we solved the problem of holes as follows. For every four neighbor pixels forming a rectangle in one of the example images, we find their projection in the new image and then we fill the interior of the projected rectangle by using a bilinear interpolation of the pixel values of the four corners. This approach was significantly better than the previous one.

4.2 Summary of the Implementation

Summarizing, the program is organized as follows:

1. Calibrate the camera using at least 3 example images.
2. Given a new position of the camera, find the corresponding matrix of external parameters and then the new camera matrix.
3. Choose two example images that are close to the one we want to generate.
4. Find the dense pixel-wise correspondence between the example images.
5. Using the new camera matrix and the camera matrices for the two example images, compute the coefficients for the trilinear equations.
6. For every four neighbor pixels forming a rectangle in the first example and their corresponding pixels in the second example, find their new coordinates in the virtual image using equations 18. Fill the interior of the rectangle in the new example as described above.
7. In the case that a pixel in the new image is written more than once, choose the pixel value according to the occlusion method described in the previous section.

4.3 Experimental Results

In this section we show the experimental results of the method. We tested the method using several set of images for several objects. Because of space constraints here we include only the results for one object. The reader can also refer to the World Wide Web page

http://www.ai.mit.edu/people/theos/theos.html

for more results of the system.

We tested the program on real images of a statue that we collected in the lab. We calibrated the camera using the examples as shown in the previous section. Having the internal parameters we generated the virtual images shown in figure 4. The two example images we used are the first two images on the left top.

4.4 Robustness of the system

Experimentally we noticed that the method was not very sensitive to noise in the computed internal parameters of the camera or in the assumed viewpoints of the example images. Changes in these two sets of parameters had the following effects:

1. Small change of the internal parameters did not have almost any visual effects. However, big changes introduced some perspective effects in the virtual images, as expected.

2. Assuming different external parameters for the example images had only the following effect for the virtual images. Suppose that the first image is from the origin and the second one is from 5 degrees rotation around the x-axis, 2 degree rotation around the y-axis. Let $G_{5.2}$ be the image generated from 10 degrees around the x-axis and 10 degrees around the y-axis. Suppose that we assume now (due to error) that the second example image is from 10 degrees rotation around the x-axis, 4 degree rotation around the y-axis instead (100 percent error). Let $G_{10.4}$ be the image generated from 20 degrees around the x-axis and 20 degrees around the y-axis given the new assumption about the position of the second image. Then we observe that $G_{10.4}$ and $G_{5.2}$ are almost identical. There is only a small prespective difference when we use the same internal parameters in both cases. Moreover, if we generate an image $G_{10.4.2}$ from 10 degrees around the x-axis and 10 degrees around the y-axis given the erroneous estimate of the position of the second image, then $G_{10.4.2}$ will be displaced "half-way" relative to $G_{5.2}$ which should be the correct one. All these are shown in image 5.

   In other words, we noticed that all that matters is not the exact positions of the camera for the example images, but instead the "relative" positions.

4.5 How well can 2 example images do?

Having these observations in mind, one would naturally ask the following question:
"If the external parameters of the camera for the example images are not very important for reprojection, and the system is not very sensitive to small errors in the internal parameters, how well can the system do if only two example images from unknown positions are given?"

Experimentally we found that when we are given only two examples we can still generate good virtual images as long as we have a good guess for the internal parameters of the

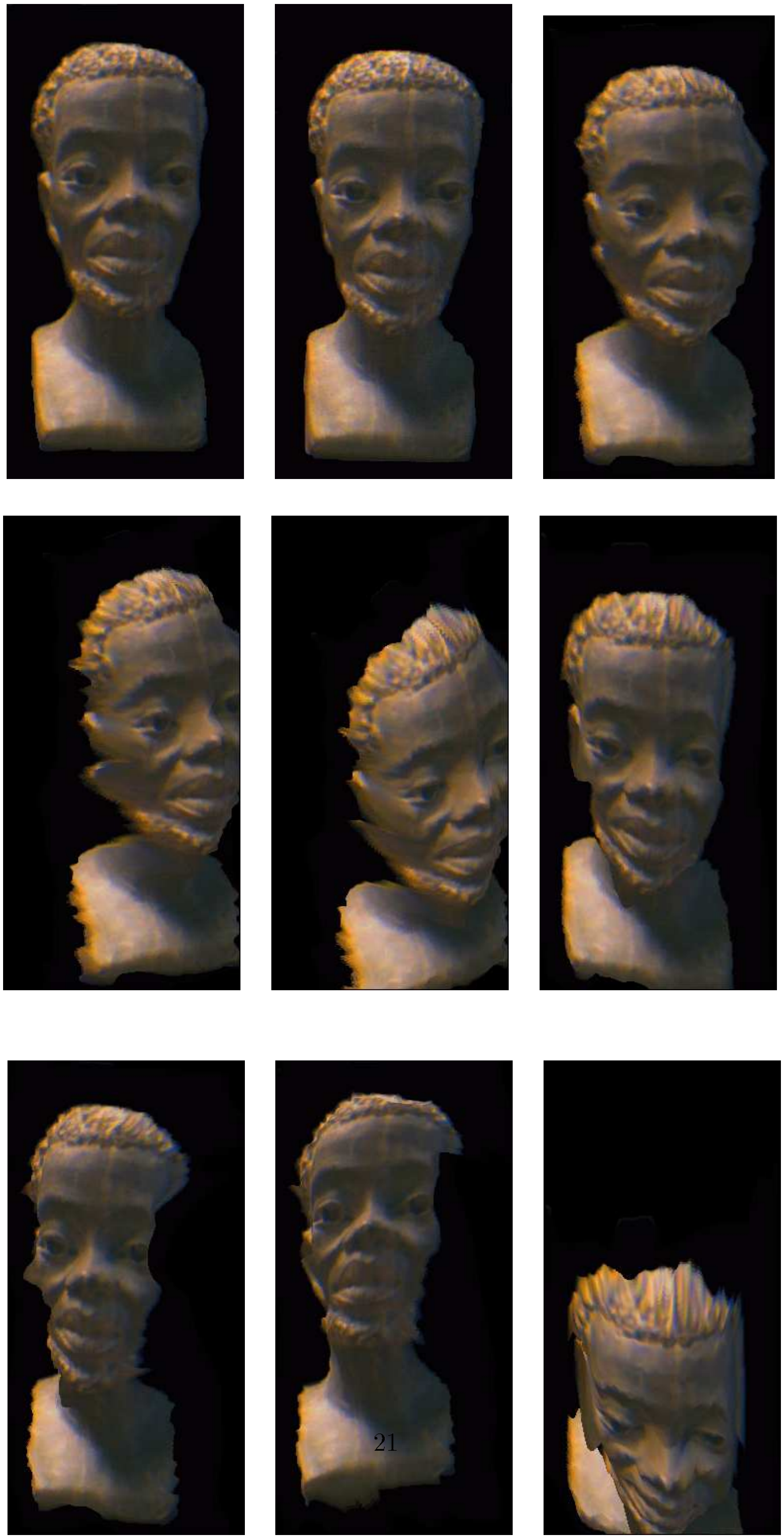

Figure 4: Statue Movie: The only real images are the first two ones on the left top.

Figure 5: Starting from the left, we show images $G_{10.4}$, $G_{5.2}$ and $G_{10.4.2}$ as described in the text. The example images are the same as in figure 4

camera. In other words, *two images, a good guess of the internal parameters of the camera and any estimate of the position of the camera for the two example images are good enough for the system to generate virtual images from other viewpoints.* This is in agreement with the observations stated above.

Big errors in the estimate of either the internal or the external parameters can potentially create some perspective effects to the results, which are not important when the errors are small. Moreover, as mentioned before, errors in the estimation of one set of parameters, say external, can partly be "balanced" by adjusting the other set of parameters, in this case internal. However we have not studied how these two sets of parameters interact with each other for reprojection using the method we developed.

5 Shape and Motion Estimation

5.1 Motivation and Background

Computing the motion and shape of a moving object is an important task for many applications such as navigation and robot manipulation. Traditionally algorithms compute shape by using depth values estimated by methods such as triangulation. However these approaches are typically noise-sensitive. Significant has been the work done by Tomasi and Kanade [29]. In [29] they developed a robust method for the special case that the images are taken under orthographic projection. Their method is based on the factorization of the matrix that contains image coordinates of selected features of the moving object. This factorization leads to a product of two matrices: one contains information about the shape of the camera, and the other about the motion of the camera. In [27] the method is extended to the paraperspective case, which is a first order approximation to the general perspective case. Moreover, in [6] a variation of the method for the multi-body case is presented.

The purpose of this section is to examine the relation between the factorization method of [29] and the algebraic relations described in the previous sections. We present this relation in the special case of orthographic projection using two different approaches and we propose a new method for shape and motion estimation when we have two or more images. Finally we suggest possible generalizations for the perspective case.

5.2 Theory

5.2.1 Problem Statement

We are given a sequence of F images of an object taken with a camera that is allowed to undergo any rotation and translation. More specifically, we have a set of P object points $p_i = (X_i, Y_i, Z_i)$, $i \in \{1, 2, ..., P\}$ projected on F images (frames), $f \in \{1, 2, ..., F\}$, taken under orthographic projection. Frame f is obtained by the camera whose orientation is described by the orthonormal unit vectors i_f, j_f and k_f, where k_f points along the camera's line of sight, i_f corresponds to the camera image plane x-axis, and j_f corresponds to the camera image plane y-axis. The camera position is described by the vector t_f from the origin of the world coordinate system to the camera's focal point. The situation is shown in figure 6. We assume that the focal length of the camera is 1 and all other internal parameters are 0. Finally, let x_{fi} and y_{fi} be the image coordinates of point p_i in frame f. The problem then is the following. Given x_{fi} and y_{fi}, compute the shape of the object, which in this case means p_i for each of the P points, and the motion of the camera, which in this case means i_f, j_f, $i_f \cdot t_f$ and $j_f \cdot t_f$ for each of the F frames. We assume that all P points are visible in all frames.

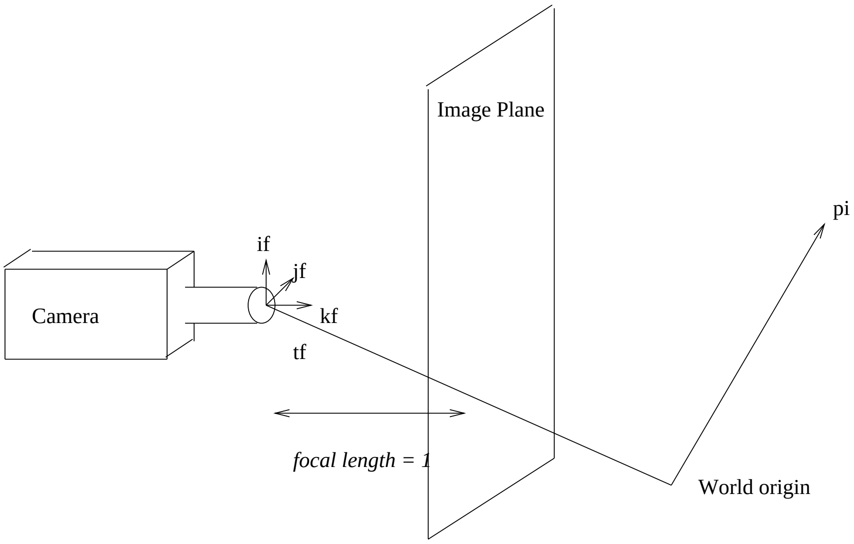

Figure 6: Coordinate system

5.2.2 Shape and Motion estimation

We organize all of the feature point coordinates into a $2F \times P$ matrix W:

$$W = \begin{bmatrix} x_{11} & \cdots & x_{1P} \\ \cdots & \cdots & \cdots \\ x_{F1} & \cdots & x_{FP} \\ y_{11} & \cdots & y_{1P} \\ \cdots & \cdots & \cdots \\ y_{F1} & \cdots & y_{FP} \end{bmatrix} \tag{23}$$

Each column of the measurement matrix contains all the observations for a single point, while each row contains all the observed x-coordinates or y-coordinates for a single frame.

As we can see from figure 7, under orthographic projection a point $p_i = (X_i, Y_i, Z_i)$ relative to the world coordinate system is projected on the image plane of frame f at image coordinates (x_{fi}, y_{fi}) given by:

$$\begin{aligned} x_{fi} &= i_f \cdot (p_i - t_f) = i_f \cdot p_i - i_f \cdot t_f \\ y_{fi} &= j_f \cdot (p_i - t_f) = j_f \cdot p_i - j_f \cdot t_f \end{aligned} \tag{24}$$

where the notation is as defined in the problem statement section.

Moreover, if we define the world coordinate system to be such that the origin is at the center of mass of the object we get that:

$$p_1 + p_2 + ... + p_P = 0 \tag{25}$$

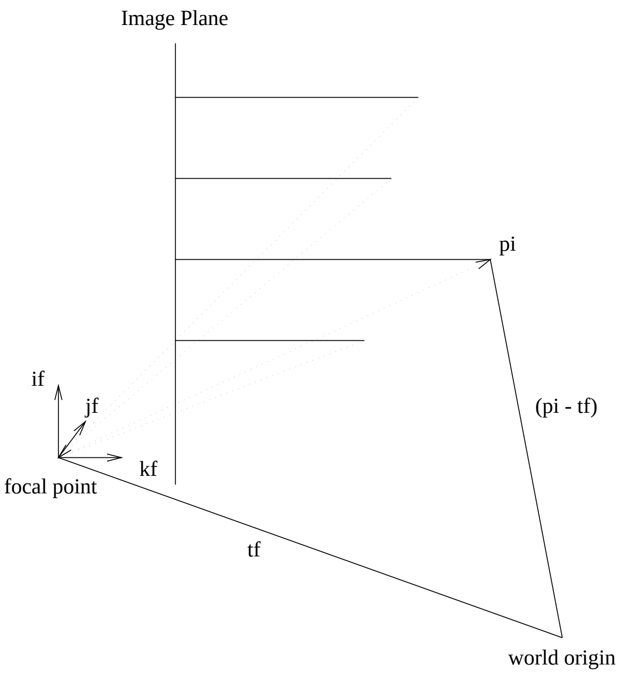

Figure 7: Orthographic Projection. Dotted lines indicate perspective projection. The solid perpendicular lines to the image plane indicate orthographic projection.

which in combination with equation 24 gives:

$$\begin{array}{c} x_{f1} + x_{f2} + ... + x_{fp} = i_f \cdot (p_1 + p_2 + ... + pP) + P(i_f \cdot t_f) = 0 + P(i_f \cdot t_f) \Rightarrow \\ \Rightarrow i_f \cdot t_f = \frac{1}{P}(x_{f1} + x_{f2} + ... + x_{fp}) \\ and \\ y_{f1} + y_{f2} + ... + y_{fp} = j_f \cdot (p_1 + p_2 + ... + pP) + P(j_f \cdot t_f) = 0 + P(j_f \cdot t_f) \Rightarrow \\ \Rightarrow j_f \cdot t_f = \frac{1}{P}(y_{f1} + y_{f2} + ... + y_{fp}) \end{array} \tag{26}$$

Therefore we can compute the camera position (translation) directly from the image coordinates by taking the average of each row of matrix W.

Once we find the camera translation, we can factor it out by subtracting from each entry of W the average of the entries of the row it is in. Once we do this we are left with a new matrix W' with entries:

$$\begin{array}{c} x'_{fi} = i_f \cdot p_i \\ y'_{fi} = j_f \cdot p_i \end{array} \tag{27}$$

The question now is how to estimate p_i, i_f and j_f. The problem, however, becomes simple after we make the following observations.

First, we can consider matrix W' as the matrix of image coordinates of the P points taken by a camera that is located at the origin of the world coordinate system and that is allowed to make only rotations - **no** translation. So we can start solving the problem of shape and motion estimation from the beginning for this new scenario. Notice that this situation is not realistic, since it would have been impossible to see all points in every frame if a camera were located at the center of the mass of the object. However, we can still use our knowledge for orthographic projection assuming, like at the beginning, that all points are visible in all frames.

Second, from equation 27, we get that the new camera matrix for frame f is:

$$K_f = \begin{bmatrix} i_f & 0 \\ j_f & 0 \\ 0_3^T & 1 \end{bmatrix} \tag{28}$$

Finally, we know that in the case of orthographic projection, "1.5 views" of an object are enough to predict other images of the object [28]. In other words, as we also see in the appendix, we know that there are α_{f1}, α_{f2}, α_{f3}, α_f and β_{f1}, β_{f2}, β_{f3}, β_f such that :

$$\begin{array}{c} (x'_{f1} \cdots x'_{fP}) = \alpha_{f1}(x'_{11} \cdots x'_{1P}) + \alpha_{f2}(y'_{11} \cdots y'_{1P}) + \alpha_{f3}(x'_{21} \cdots x'_{2P}) + \alpha_f \\ (y'_{f1} \cdots y'_{fP}) = \beta_{f1}(x'_{11} \cdots x'_{1P}) + \beta_{f2}(y'_{11} \cdots y'_{1P}) + \beta_{f3}(x'_{21} \cdots x'_{2P}) + \beta_f \end{array} \tag{29}$$

Furthermore, in this case $\alpha_f = \beta_f = 0$, since there is no translation from frame to frame. So the relations above immediately give us that matrix W' has rank 3. This is the *rank 3 theorem* proved in [29]. Using equation 29 we get another simple proof of that theorem - which is anyway clear from equation 27.

Using these equations we can approach the factorization problem of [29] in a different way. We can write W' as a product of a $2F \times 3$ and a $3 \times P$ matrix as follows;

$$W' = \begin{bmatrix} x_{11'} & \cdots & x'_{1P} \\ x'_{21} & \cdots & x'_{2P} \\ x'_{31} & \cdots & x'_{3P} \\ \cdots & \cdots & \cdots \\ x'_{F1} & \cdots & x'_{FP} \\ y'_{11} & \cdots & y'_{1P} \\ y'_{21} & \cdots & y'_{2P} \\ \cdots & \cdots & \cdots \\ y'_{F1} & \cdots & y'_{FP} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \\ \cdots & \cdots & \cdots \\ \alpha_{F1} & \alpha_{F2} & \alpha_{F3} \\ 0 & 1 & 0 \\ \beta_{21} & \beta_{22} & \beta_{23} \\ \cdots & \cdots & \cdots \\ \beta_{F1} & \beta_{F2} & \beta_{F3} \end{bmatrix} \times \begin{bmatrix} x'_{11} & \cdots & x'_{1P} \\ y'_{11} & \cdots & y'_{1P} \\ x'_{21} & \cdots & x'_{2P} \end{bmatrix} \tag{30}$$

This is one of the possible factorizations as mentioned in [29]. Indeed, the crucial question is how to factorize W'. Notice that for any invertible matrix A, from a factorization $W' = MS$ of W' we can get another factorization by simply replacing M with MA and S with $A^{-1}S$. Indeed: $W' = MS = (MA)(A^{-1}S)$ for any invertible 3x3 matrix A. So the question is *how* *to factorize W' in order to get the correct motion and shape matrices.* The factorization that we want to get is:

$$W = \begin{bmatrix} x'_{11} & \cdots & x'_{1P} \\ x'_{21} & \cdots & x'_{2P} \\ x'_{31} & \cdots & x'_{3P} \\ \cdots & \cdots & \cdots \\ x'_{F1} & \cdots & x'_{FP} \\ y'_{11} & \cdots & y'_{1P} \\ y'_{21} & \cdots & y'_{2P} \\ \cdots & \cdots & \cdots \\ y'_{F1} & \cdots & y'_{FP} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ i_{21} & i_{22} & i_{23} \\ i_{31} & i_{32} & i_{33} \\ \cdots & \cdots & \cdots \\ i_{F1} & i_{F2} & i_{F3} \\ 0 & 1 & 0 \\ j_{21} & j_{22} & j_{23} \\ \cdots & \cdots & \cdots \\ j_{F1} & j_{F2} & j_{F3} \end{bmatrix} \times \begin{bmatrix} X_1 & \cdots & X_P \\ Y_1 & \cdots & Y_P \\ Z_1 & \cdots & Z_P \end{bmatrix} \tag{31}$$

since we have that $x_{fi} = i_f \cdot [X_i, Y_i, Z_i]^T$ and $y_{fi} = j_f \cdot [F_i, Y_i, Z_i]^T$.

To get the correct factorization we need one more step. From section 3 we know exactly the relations between the entries of the motion matrix we have and the ones we want to have! We can get them from the relation of the α_f, β_f and the camera matrices. For simplicity, let's define the world coordinate system so that each axis is parallel to the corresponding axis of the camera coordinate system of the first camera. In the general case - since we don't know the orientation of the first camera - the approach is exactly the same although the algebra is more complicated.

With this "alignment" the new camera matrix for the first frame is clearly:

$$K_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{32}$$

Let the rows of K_1 be 1, 2, 3, the rows of K_2 be 4, 5, 6, and the rows of matrix K_f be 7, 8, 9. Using the results of Faugeras [11] or simple geometric arguments we can find that:

$$\begin{array}{cc} \alpha_{f1} = \frac{T_{111}}{T_{313}} = \frac{i_{f1}i_{23} - i_{f3}i_{21}}{i_{23}}, f > 2 & \alpha_{f2} = \frac{T_{211}}{T_{313}} = \frac{i_{f2}i_{23} - i_{f3}i_{22}}{i_{23}}, f > 2 \\ \alpha_{f3} = -\frac{T_{331}}{T_{313}} = \frac{i_{f3}}{i_{23}}, f > 2 & \beta_{f1} = \frac{T_{112}}{T_{313}} = \frac{j_{f1}i_{23} - j_{f3}i_{21}}{i_{23}}, f > 1 \\ \beta_{f2} = \frac{T_{212}}{T_{313}} = \frac{j_{f2}i_{23} - j_{f3}i_{22}}{i_{23}}, f > 1 & \beta_{f3} = -\frac{T_{332}}{T_{313}} = \frac{j_{f3}}{i_{23}}, f > 1 \end{array} \tag{33}$$

where T_{ijk} is the determinant of the matrix that is made by choosing the ith element of the first column the jth of the second and the kth of the third of the following "matrix":

$$\begin{bmatrix} (23) & 4 & 7 \\ (31) & 5 & 8 \\ (12) & 6 & 9 \end{bmatrix} \tag{34}$$

using this last step we get that:

$$W = \begin{bmatrix} x'_{11} & \cdots & x'_{1P} \\ x'_{21} & \cdots & x'_{2P} \\ x'_{31} & \cdots & x'_{3P} \\ \cdots & \cdots & \cdots \\ x'_{F1} & \cdots & x'_{FP} \\ y'_{11} & \cdots & y'_{1P} \\ y'_{21} & \cdots & y'_{2P} \\ \cdots & \cdots & \cdots \\ y'_{F1} & \cdots & y'_{FP} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \\ \cdots & \cdots & \cdots \\ \alpha_{F1} & \alpha_{F2} & \alpha_{F3} \\ 0 & 1 & 0 \\ \beta_{21} & \beta_{22} & \beta_{23} \\ \cdots & \cdots & \cdots \\ \beta_{F1} & \beta_{F2} & \beta_{F3} \end{bmatrix} \times \begin{bmatrix} x'_{11} & \cdots & x'_{1P} \\ y'_{11} & \cdots & y'_{1P} \\ x'_{21} & \cdots & x'_{2P} \end{bmatrix} = (x_{11} = X_1, y_{11} = Y_1) \tag{35}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \\ \cdots & \cdots & \cdots \\ \alpha_{F1} & \alpha_{F2} & \alpha_{F3} \\ 0 & 0 & 1 \\ \beta_{21} & \beta_{22} & \beta_{23} \\ \cdots & \cdots & \cdots \\ \beta_{F1} & \beta_{F2} & \beta_{F3} \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ i_{21} & i_{22} & i_{23} \end{bmatrix} \times \begin{bmatrix} X_1 & \cdots & X_P \\ Y_1 & \cdots & Y_P \\ Z_1 & \cdots & Z_P \end{bmatrix} = \tag{36}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ i_{21} & i_{22} & i_{23} \\ i_{31} & i_{32} & i_{33} \\ \cdots & \cdots & \cdots \\ i_{F1} & i_{F12} & i_{F3} \\ 0 & 1 & 0 \\ j_{21} & j_{22} & j_{23} \\ \cdots & \cdots & \cdots \\ j_{F1} & j_{F2} & j_{F3} \end{bmatrix} \times \begin{bmatrix} X_1 & \cdots & X_P \\ Y_1 & \cdots & Y_P \\ Z_1 & \cdots & Z_P \end{bmatrix} \tag{37}$$

This is a natural result from the relation (equation 33) between the α_f and β_f with i_f and

j_f. Indeed, with simple manipulation of those equations, we get that:

$$\begin{aligned} i_{f1} &= \alpha_{f1} + \alpha_{f3} \cdot i_{21} \\ i_{f2} &= \alpha_{f1} + \alpha_{f3} \cdot i_{21} \\ i_{f3} &= \alpha_{f1} + \alpha_{f3} \cdot i_{21} \\ j_{f1} &= \beta_{f1} + \beta_{f3} \cdot i_{21} \\ j_{f2} &= \beta_{f1} + \beta_{f3} \cdot i_{21} \\ j_{f3} &= \beta_{f1} + \beta_{f3} \cdot i_{21} \end{aligned} \tag{38}$$

There is one last problem. As we see in the equation above, in the general case we need to know the orientation of the first and second camera! As a first step we can either assume that we are given this motion, or estimate it using any of the existing methods for motion estimation. The key, though, is that once we estimate shape and motion from the first two images, we can get the motion for any subsequent image without paying much! The motion comes directly as a result of the linearity constraints described in the appendix. To summarize, in the general case we manage to change the question of shape and motion estimation from many images to the question of shape and motion estimation from two images. In the general case that we don't know the first and second camera orientations the equations above become:

$$W = \begin{bmatrix} x'_{11} & \cdots & x'_{1P} \\ x'_{21} & \cdots & x'_{2P} \\ x'_{31} & \cdots & x'_{3P} \\ \cdots & \cdots & \cdots \\ x'_{F1} & \cdots & x'_{FP} \\ y'_{11} & \cdots & y'_{1P} \\ y'_{21} & \cdots & y'_{2P} \\ \cdots & \cdots & \cdots \\ y'_{F1} & \cdots & y'_{FP} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \\ \cdots & \cdots & \cdots \\ \alpha_{F1} & \alpha_{F2} & \alpha_{F3} \\ 0 & 1 & 0 \\ \beta_{21} & \beta_{22} & \beta_{23} \\ \cdots & \cdots & \cdots \\ \beta_{F1} & \beta_{F2} & \beta_{F3} \end{bmatrix} \times \begin{bmatrix} x'_{11} & \cdots & x'_{1P} \\ y'_{11} & \cdots & y'_{1P} \\ x'_{21} & \cdots & x'_{2P} \end{bmatrix} = (x_{11} = X_1, y_{11} = Y_1) \tag{39}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \\ \cdots & \cdots & \cdots \\ \alpha_{F1} & \alpha_{F2} & \alpha_{F3} \\ 0 & 0 & 1 \\ \beta_{21} & \beta_{22} & \beta_{23} \\ \cdots & \cdots & \cdots \\ \beta_{F1} & \beta_{F2} & \beta_{F3} \end{bmatrix} \times \begin{bmatrix} i_{11} & i_{12} & i_{13} \\ j_{11} & j_{12} & j_{13} \\ i_{21} & i_{22} & i_{23} \end{bmatrix} \times \begin{bmatrix} X_1 & \cdots & X_P \\ Y_1 & \cdots & Y_P \\ Z_1 & \cdots & Z_P \end{bmatrix} = \tag{40}$$

$$
= \begin{bmatrix} i_{11} & i_{12} & i_{13} \\ i_{21} & i_{22} & i_{23} \\ i_{31} & i_{32} & i_{33} \\ \cdots & \cdots & \cdots \\ i_{F1} & i_{F12} & i_{F3} \\ j_{11} & j_{12} & j_{13} \\ j_{21} & j_{22} & j_{23} \\ \cdots & \cdots & \cdots \\ j_{F1} & j_{F2} & j_{F3} \end{bmatrix} \times \begin{bmatrix} X_1 & \cdots & X_P \\ Y_1 & \cdots & Y_P \\ Z_1 & \cdots & Z_P \end{bmatrix} \tag{41}
$$

which is exactly the factorization in [29] derived in a different way using the linearity constraints between different images of an object.

5.2.3 Proposed Method for Motion Estimation with n > 2 Images

Using this theory we can suggest now a method for estimating motion when we are given a sequence of images of an object taken under orthographic projection by a camera undergoing any rigid motion. Organizing the image coordinates in matrix W as described above, do:

1. Compute the camera position (translation) by averaging the rows of matrix W.

2. Subtract the translation from the entries of W to get matrix W'.

3. Using any existing method estimate shape and motion for the first two images.

4. For every $f > 2$ find corresponding points between image f and images 1 and 2. Use these correspondences to estimate the coefficients α_f and β_f of the linear relations described in equation 29, using a least squares method to minimize the error.

5. Having found the linearity coefficients we can factor W' as a product $W = MS$ like in equation 30. However, because of noise this will not be an exact factorization, but it is a good approximation since we estimated α_f and β_f so that we minimize any error.

6. Using the known orientation of the first and second camera, we can write down the motion and shape matrices in the general case using equation 41.

So, in principle, one can get all subsequent motions by just knowing the first two motions of the camera.

5.3 Shape and Motion under Perspective Case

There are several ways to approximate the perspective case. One way is to use an approximation of the projection equations 1 in section 2. For example if we replace the ratios with their Taylor expansions we can use a similar method as above in the case that we use a first order approximation.

Instead of approximating perspective projection from the beginning by replacing the projection equations as mentioned above, we can use Taylor expansion to replace the trilinear equations. If we expand the trilinearities up to first order factors, we can use exactly the same approach described above, but this time the coefficients α_f and β_f will be different. In this case the relation between α_f, β_f and motion is not straightforward. We still know the relation between motion and the initial coefficients of the trilinearity equations, but after doing the Taylor expansion things get more complicated. We have not studied what happens in case we use higher order approximations, since the algebra gets complicated. However, it is possible that a combination of the aforementioned approach and this one will give good results in the general perspective case.

6 Conclusion and Suggestions for Further Research

6.1 Summary of the Paper

In the main part of this paper we designed and tested a new method for generating virtual views of an object undertaking any rigid motion. Given three images taken from known positions A, B and C we can first compute the internal parameters of the camera. Then we showed that one can get any virtual image of the object from any position near A, B and C using the trilinearity constraints that bind any new image with any two real ones. As a side result of the paper we proposed two new methods for camera calibration and we developed one of them which we also used to estimate our camera model. We tested both methods on real images of an objects and faces.

Moreover we showed that even when only two example images taken from unknown positions are given, the system still works satisfactorily given a good estimate of the internal parameters of the camera. Although further tests are needed to test the robustness of the system, the results we had were encouraging. Using few example images we were able to get a continuous range of virtual images in a satisfactory wide range of views relative to the examples. The new images were realistic and gave nice 3-D effects to the user, and the degree of extrapolation was satisfactory.

In the second part of the paper we studied the relation between the factorization method for shape and motion estimation presented in [29] and the algebraic relations between images. We presented the basic ideas for motion estimation using the linearity constraints that bind three views of the moving object in the special case of orthographic projection. We showed that shape and motion estimation from $F > 2$ images is equivalent to shape and motion estimation from 2 images and motion estimation for the remaining ones. As a result of this relation we developed a new simple method for motion estimation from more than two images in the special case of orthographic projection. We also suggested possible extensions for the general case of perspective projection. However we did not implement the proposed method and suggestions. Further research on this approach might lead to a new system for motion estimation.

6.2 Suggestions for Future Research

One of the most important questions to be solved is the visibility problem. Although in many cases, for practical purpose, the heuristic we used is satisfying, the problem still remains for most cases. For example we still cannot detect situations where a point is visible in only one of the example images. We believe that such cases are often the reason that holes appear in the virtual image, since wrong correspondences cause the trilinear relations to break. Very little research has been done on this issue. Although the problem might be unsolvable in general, further research for better heuristics is necessary.

At a different but important direction, the system developed in this paper can be combined with one such as [8] with the long term goal of developing a complete tool that can be used for generation of images of objects undergoing any transformation - rigid as well as flexible. For example we can formally describe the process of generating a new image I^{new} for a given point $\mathbf{r}$ in the space of transformations as a mapping

$$I^{new} = g(\mathbf{r}; I_1, \cdots, I_n)$$

where the other arguments of the function are the example images. The mapping g can be formally decomposed into $g = \psi\phi$. ψ is the transformation due to a change in viewpoint while ϕ is the transformation due to a non rigid motion of the object in the scene. When the desired image results from a camera transformation, that is a rigid object transformation, $g = \psi$ has the analytic form studied in this paper. On the other hand, for the non-rigid transformation ϕ we can use a system such as [8].

Research on image-based rendering is still at its first stages. Future research in this area can potentially lead to significant results. We believe that the most important issues that need to be solved are the visibility problem as well as speed and memory efficiency issues. Notice that having many example images of an object means having every point of the object in multiple images. Avoiding such a memory inefficient representation is an important issue. We believe that the question of representation of the objects - ie image-based or 3-D model or something in between - is a central issue in machine vision and computer graphics. Moreover many questions will unavoidably arise from the need to combine image-based rendering techniques for objects with ones for panoramas (such as [18]) as well as with 3-D models used either for some of the objects or for parts of some objects - depending on their representation. Such hybrid systems can potentially lead on the one hand to realistic virtual reality tools and on the other hand to better understanding of important issues in machine vision as well as our vision system, such as how we represent an object, how we deal with occlusions, or how we find correspondences between different images of an object.

A Trilinear Constraints Between Three Images

Like in section 3, suppose we are given three images I_1, I_2, I_3 of an object point $O = (X, Y, Z)$ taken with three cameras from different positions. Let 1, 2, 3 be the rows of the first camera matrix, 4, 5, 6 be the rows of the second and 7, 8, 9 the rows of the third. Let (x, y), (x', y'), (x'', y'') be the image coordinates of the projection of O in the three images respectively. We assume O is visible in all three images, so its three projections are Euclidean points (their third projective coordinate can be set to 1). Then the 9 trilinear equation that hold between the image coordinates are of the form:

$$\begin{aligned} THS_{i,j} = x(&u'_i u''_j T_{133} - u''_j T_{1i3} - u'_i T_{13j} + T_{1ij} + \\ +y(&u'_i u''_j T_{133} - u''_j T_{1i3} - u'_i T_{13j} + T_{1ij} + \\ +(&u'_i u''_j T_{133} - u''_j T_{1i3} - u'_i T_{13j} + T_{1ij} + \end{aligned} \tag{42}$$

for i, j in $1, 2, 3$, where $u'_1 = x', u'_2 = y', u'_3 = 1, u''_1 = x'', u''_2 = y'', u'_3 = 1$, and T_{ijk} is the determinant of the matrix obtained by choosing the ith element of the first column, the jth element of the second, and the kth of the third of the following matrix:

$$\begin{bmatrix} 2,3 & 4 & 7 \\ 3,1 & 5 & 8 \\ 1,2 & 6 & 9 \end{bmatrix} \tag{43}$$

So, for example, the equations that we use in section 3 are derived for $i = 1$, $j = 1$ and for $i = 2$, $j = 2$. Indeed, substituting i and j we get:

$$\begin{aligned} &[2369]x''xx' - [2349]x''x + [3169]x''yx' - [3149]x''y + [1269]x''x' - [1249]x'' - \\ &-[2367]xx' + [2347]x - [3167]yx' + [3147]y - [1267]x' + [1247] = 0 \end{aligned} \tag{44}$$

$$\begin{aligned} &[2369]y''xy' - [2359]y''x + [3169]y''yy' - [3159]y''y + [1269]y''y' - [1259]y'' - \\ &-[2368]xy' + [2358]x - [3168]yy' + [3158]y - [1268]y' + [1258] = 0 \end{aligned} \tag{45}$$

These are the equations as shown in [11].

A.1 The Case of Orthographic Projection

Using the formulation above, we can now see why in the special case of orthographic projection the trilinearity constraints become linearities. In the case of orthographic projection the camera matrices, as we show in section 2, have a special form: the third row is always [0001]. Taking this into account we have that, for example for equation 44, the coefficients become:

$$[2369] = [2349] = [3169] = [3149] = [1269] = [2367] = [3167] = 0 \tag{46}$$

since all these coefficients are determinants of matrices that have two equal rows (since rows 3, 6, 9 are all [0001]). The remaining coefficients can be non-zero. Notice that the only non-zero coefficients are the ones of the first or zero order terms, which makes the trilinearity into a linear equation. Similarly all the other trilinearity equations become linear.

References

[1] S. Avidan and A. Shashua. Novel view synthesis in tensor space. CIS Technical Report 9602, Technion University, Israel, January 1996.

[2] J.R. Bergen and R. Hingorani. Hierarchical motion-based frame rate conversion. Technical report, David Sarnoff Research Center, Princeton, New Jersey, April 1990.

[3] D. Beymer, A. Shashua, and T. Poggio. Example based image analysis and synthesis. A.I. Memo No. 1431, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1993.

[4] Shenchang Eric Chen. Quicktime vr - an image-based approach to virtual environment navigation. In *SIGGRAPH '95 Proceedings*, pages 29–37, Los Angeles, CA, August 1995.

[5] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *SIGGRAPH '93 Proceedings*, pages 279–288, Anaheim, CA, August 1993.

[6] J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. *Proceedings of the International Conference on Computer Vision*, pages 1071-1076, Cambridge, MA, June 1995.

[7] R. Deriche and Z. Zhang and Q.-T. Luong and O. Faugeras. Robust recovery of the epipolar geometry for an uncalibrated stereo rig. *Lecture Notes in Computer Science, Vol.800, Computer Vision - ECCV '94*

[8] T. Ezzat. Example-Based Analysis and Synthesis for Images of Human Faces. Master Thesis, Massachusetts Institute of Technology, February 1996.

[9] O. Faugeras. Three-dimensional computer vision: a geometric viewpoint. MIT Press, 1993.

[10] O. Faugeras, Q.-T. Luong and S.J. Maynank. Camera self-calibration: theory and experiments. *Proceedings European Conference on Computer Vision*, pages 321-334, Santa-Margerita, Italy, 1992.

[11] O. Faugeras and B. Mourrain. On the geometry and algebra of the point and line correspondences between N images. *Proceedings of the International Conference on Computer Vision*, pages 951-956, Cambridge, MA, June 1995.

[12] O. Faugeras and L. Robert. What can two images tell us about a third one? *Proceedings of the European Conference on Computer Vision*, pages 485-492, Stockholm, Sweden, May 1994.

[13] R. Hartley. Lines and points in three views - a unified approach. *Proc. Image Understanding Workshop*, Monterey, California, November 1994.

[14] R. Hartley. In defence of the 8-point algorithm. *Proceedings of the International Conference on Computer Vision*, pages 1064-1070., Cambridge, MA, June 1995.

[15] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[16] K. Kanatani. Computational Projective Geometry. *Computer Vision, Graphics and Image Processing. Image Understanding*, 54(3), 1991.

[17] S. Laveau and O. Faugeras. 3-d scene representation as a collection of images and fundamental matrices. Technical Report Technical Report No. 2205, INRIA, 1994.

[18] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *SIGGRAPH '95 Proceedings*, Los Angeles, CA, 1995.

[19] Tomaso Poggio and Roberto Brunelli. A novel approach to graphics. A.I. Memo No. 1354, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1992.

[20] W.H. Press, B.P. Flannery, S.A. Teukolski and W.T. Vetterling. Numerical recipes in C. Cambridge University Press, Cambridge, Massachusetts, pp.309-317, 1988.

[21] Steven M. Seitz and Charles R. Dyers. Physically-valid view synthesis by image interpolation. *IEEE Workshop on Representation of Visual Scenes*, Boston, USA, June 1995.

[22] A. Shashua. Algebraic functions for recognition. *MIT Artificial Intelligence Laboratory, Memo 1452*, Cambridge, MA 1994.

[23] A. Shashua. Trilinearity in visual recognition by alignment. *Lecture Notes in Computer Science, Vol.800, Computer Vision - ECCV '94*

[24] A. Shashua. Algebraic functions for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):779-789, 1995.

[25] G. Stein. Internal camera calibration using rotation and geometric shapes. *MIT Artificial Intelligence Laboratory, Technical Report 1426*, Cambridge, MA 1993.

[26] Demetri Terzopoulos and Keith Waters. Analysis of facial images using physical and anatomical models. In *Proceedings of the International Conference on Computer Vision*, pages 727–732, Osaka, Japan, December 1990.

[27] C. Poelman and T. Kanade. A paraperspective factorization method for shape and motion recovery. *Proceedings European Conference on Computer Vision*, 1994.

[28] T. Poggio and T. Vetter. Recognition and structure from one 2D model view: observations on prototypes, object classes and symmetries. *Artificial Intelligence Memo 1347*, Massachusetts Institute of Techology, February 1992.

[29] C. Tomasi and T. Kanade. Shape and motion from image streams: a factorization method. *Technical Report CMU-CS-91-172*, Carnegie Mellon University, Pittsburgh, PA, September 1991.

[30] Shimon Ullman and Ronen Basri. Recognition by linear combinations of models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):992–1006, 1991.

[31] Tomáš Werner, Roger David Hersch, and Václav Hlaváč. Rendering real-world objects using view interpolation. In *Proceedings of the International Conference on Computer Vision*, Cambridge, MA, June 1995.