3-D Reconstruction of Urban Scenes from Image Sequences

Olivier Faugeras, Luc Robert, Stéphane Laveau*
and
Gabriella Csurka, Cyril Zeller, Cyrille Gauclin and Imed Zoghlami
INRIA. 2004, route des Lucioles. B.P. 93. 06902 Sophia-Antipolis. FRANCE.

September 11, 1997

*Stéphane Laveau was supported by a grant under DRET contract No 91-815/DRET/EAR. This work was also partially funded by the EEC under Esprit project 6448, Viva and Esprit project 8878, Realise.

Abstract

In this paper, we address the problem of the recovery of a realistic textured model of a scene from a sequence of images, without any prior knowledge either about the parameters of the cameras, or about their motion. We do not require any knowledge of the absolute coordinates of some control points in the scene to achieve this goal. First, using various computer vision tools, we establish correspondences between the images and recover the epipolar geometry, from which we show how to compute the complete set of perspective projection matrices for all camera positions. Then, we proceed to reconstruct the geometry of the scene. We show how to rely on information of the scene such as parallel lines or known angles in order to reconstruct the geometry of the scene up to respectively an unknown affine transformation or an unknown similitude. Alternatively, if this information is not available, we can still recover the Euclidean structure of the scene through the techniques of self-calibration. The scene geometry is modeled as a set of polyhedra. Textures to be mapped on the scene polygons are extracted automatically from the images. We show how several images can be combined through mosaicing in order to automatically remove visual artifacts such as pedestrians or trees from the textures.

This vision system has been implemented as a vision server, which provides to a CAD-CAM modeler geometry or texture information extracted from the set of images. The whole system allows efficient and fast production of scene models of high quality for such applications as simulation, virtual or augmented reality.

1 Introduction

The problem which is tackled in this paper and for which we propose a number of partial solutions is the following: we want to reconstruct a textured three-dimensional model of a static environment viewed by one or several cameras whose motions or relative positions are unknown and whose intrinsic parameters are also unknown and may vary.

The sequence of images that is used can be either a video sequence, or a film, or a number of snapshots taken from usually fairly distinct viewpoints. In the first two cases, which we will denote as the M(ovie)-situation, it is, as explained in section 3.1 possible to use the continuity in time of the images to help simplify the problem. In the third case, which we will denote by the S(napshot)-situation, this is not possible, and we must work a little harder, as also explained in the same section.

Solving this problem at relatively low cost is extremely important for such applications as image synthesis, simulation, virtual and augmented reality. A number of techniques have been proposed so far for producing 3D information out of images in photogrammetry and computer vision. The photogrammetry approach mostly focuses on accuracy problems, and the derived techniques produce three-dimensional models of high accuracy [1]. However, they generally require heavy human interaction. Some commercial products, such as *Photomodeler*, already integrate these techniques. In computer vision, people produced a number of automatic techniques for computing structure from stereo or motion ([41] and [10, 26, 12] for reviews). With these techniques, the three-dimensional models are produced much more easily, but they are less accurate and potentially contain a small fraction of gross errors. Recent hybrid approaches aim at reducing the effort in the production of explicit 3D models of high quality by imposing constraints on the modeled scene [9].

Alternate representations have been proposed for realistic rendering from images. With image interpolation techniques [13, 39, 45], the scene is represented as a depth field, or equivalently, as a set of feature correspondences across two reference images. Techniques based on an explicit description of the 4D light intensity function have also been proposed [30]. Though suited to realistic rendering, these techniques correspond to specific conditions (viewpoints of rendered images are close to camera positions) which do not necessarily correspond to our hypotheses.

Our techniques build upon the knowledge which has been acquired in computer vision and photogrammetry in the last 20 years or so and can potentially reduce by a significant factor the amount of manual interaction which is currently necessary to get 3-D models of the world in the computer. As was mentioned before, no hypotheses are made upon the relative positions of the cameras, their intrinsic parameters, all of them are assumed unknown, or upon the presence in the environment to be modeled of control points with known coordinates in some fixed frame of reference. We nonetheless show in this article that the complete projective, affine, and Euclidean geometry (up to a global scale factor) of the scene can be accurately captured by a combination of techniques which encompass a wide range of traditionally distinct subjects such as feature detection (edges, corners, junctions) using non-parametric (image-based) and parametric (snake-like) models, tracking of image features in a sequence of images, geometric modeling of image correspondences at the projective, affine, and Euclidean levels with a clear distinction between those levels and

the amount of information they require, robust estimation of algebraic instantiation of this geometry (i.e. the perspective projection matrices).

In addition to a geometric description of the scene, realistic rendering applications rely on information such as textures or photometric properties of the scene. In this article we present the techniques which allow us to combine information from several images in order to extract obstacle-free textures from the images.

Far from being after a complete automation of the modeling of the scene, we aim at providing to a human agent working with a CAD system from a set of images of the environment he wants to model, the most advanced tools in computer vision to help him solve such problems as accurate (i.e. subpixel) detection of image features, matching of those features across views, estimation of the geometry of the set of views, computation of the 3-D coordinates of scene points, curves, and surfaces. This framework is depicted in figure 1.

Figure 1: approximately here.

In section 2, we give the geometric background which will be used in the remainder of the article. Section 3 shows the details of the estimation of the projective geometry of the cameras and the scene. In section 4 we show how we derive affine and Euclidean geometric models of the scene. In section 5 we show how textures are extracted automatically from the images. In section 6 we outline our computer-vision-assisted modeler and show reconstructed models. Section 7 concludes the paper.

2 Background

In this section, we briefly review the basic geometric material which is essential for the rest of the article.

2.1 The geometry of cameras

We assume the cameras to follow the stantard *pinhole model*: The image m of a point M is obtained by perspective projection through the *optical center* C onto the *retina* plane R. The line joining C, m and M is called an *optical ray*. With homogeneous coordinates, projection is represented by the simple equation: $\mathbf{m} = \mathbf{PM}$, where $\mathbf{P}$ is the 3×4 so-called perspective projection matrix of the camera, and all quantities are defined up to an unknown scale factor.

Two-camera systems: The fundamental property of a system with two cameras is the epipolar geometry: given a point in one image, we can draw a line in the second image on which its corresponding point necessarily lies. This *epipolar line* is the projection on the second image of the optical ray defined by the point in the first camera (figure 2). It only depends on the position of the point in the first image and on the geometric configuration of the cameras.

We will use the *fundamental* matrix representation of the epipolar geometry. In this representation, two points in correspondence in images 1 and 2 (expressed

in homogeneous coordinates) $\mathbf{m}_1$ and $\mathbf{m}_2$ satisfy the following projective relation: $\mathbf{m}_2^T \mathbf{F}_{12} \mathbf{m}_1 = 0$. The fundamental matrix is defined up to a scale factor and satisfies $\mathbf{F}_{21}\mathbf{e}_{21} = \mathbf{F}_{12}\mathbf{e}_{12} = 0$, with $\mathbf{e}_{ij}$ being the epipole in image i generated by image j (or equivalently, the image in i of the optical center of camera j). (figure 2).

Figure 2: approximately here.

Recently, it has been discovered that the full calibration of the cameras (intrinsic and extrinsic parameters) is not needed to obtain a useful reconstruction of a scene viewed by a stereo system [11, 24]: one only needs to know the epipolar geometry which can be retrieved from point correspondences in pairs of images. Since these first attempts at an uncalibrated stereovision, a lot of work has been done on the estimation of the epipolar geometry of two images [34, 35, 37, 36, 22, 21, 40, 4]. Robust programs which work automatically are now publicly available. We will consider this problem as solved for the rest of this article; the interested reader is referred to the bibliography.

Three-cameras systems: It has been shown that the relative geometry of three images can be captured by a tensor, which imposes a number of trilinear relations between three image points which represent the same point in space. These trilinearities [15, 49, 23, 46] yield a very convenient way of predicting the image of a point or a line in a view given its images in two other views. It has been shown that the tensor depends on 18 independent coefficients, which can be easily deduced from the projection matrices.

Finding a suitable representation: The fundamental matrix depends upon seven parameters [33]. Therefore, the set of all possible fundamental matrices between N cameras would depend on $7N(N-1)/2$ parameters if there were no constraints between them. However, in our simple perspective model, each camera depends on a fixed number of parameters (we use 6 for the pose and orientation and 5 for the intrinsic or internal parameters). This leads to $O(N)$ parameters for the cameras and since the fundamental matrices are represented by $O(N^2)$ parameters, there exist constraints between them. These constraints can be enumerated [15, 16], but they are rather complex and difficult to use. For analogous reasons, representing the geometry of more than three images through trifocal tensors is difficult, because of the complexity of the constraints between the various tensors. In turn, we prefer the more compact representation which consists of the projective camera matrices. From this representation, one can easily derive the fundamental matrices and the trifocal tensors.

2.2 Computing the projection matrices

It has been shown by [32] that given a set of fundamental matrices satisfying the constraints, one can find corresponding projection matrices. The solution is unique up to an unknown projective transformation in space if the optical centers are not aligned.

The relation of the projection matrices to the fundamental matrices is simple: If we write $\mathbf{P}_i$, the projection matrix of camera i, as $[\mathbf{M}_i|\mathbf{t}_i]$, the epipoles $\mathbf{e}_{ij}$ satisfy Eq. (1) by definition (as images of an optical center).

$$\mathbf{e}_{ij} = \mathbf{t}_i - \mathbf{M}_i\mathbf{M}_j^{-1}\mathbf{t}_j \tag{1}$$

For the fundamental matrices, an elimination scheme leads to

$$\mathbf{F}_{ij} = [\mathbf{e}_{ij}]_\times \mathbf{M}_j\mathbf{M}_i^{-1} \tag{2}$$

where $[\mathbf{e}_{ij}]_\times$ is the 3×3 matrix such that for any vector $\mathbf{x}$, $[\mathbf{e}_{ij}]_\times \mathbf{x} = \mathbf{e}_{ij} \times \mathbf{x}$.

It is understood that these equations are projective and therefore are defined only up to an unknown scale factor. We assume that $\mathbf{M}_i$ and $\mathbf{M}_j$ are invertible. If they are not, we can always transform them by a proper choice of a projective transformation to a new frame such that they are invertible and satisfy our assumptions.

2.3 Reconstruction

From the consistent epipolar geometry, we can recover the 3D scene up to an unknown projective transformation of space. Various methods have been compared for reconstructing points in the projective space. Based on a study given in [43], we use a SVD-based method.

A projective reconstruction is not as far from a Euclidean reconstruction as it seems. The set of projection matrices for N cameras depends upon $11N - 7$ parameters in the Euclidean case, whereas only $11N - 15$ in the projective case.

These 8 additional free parameters are the *low* price to pay for not knowing the internal parameters of the cameras and their relative positions in space. We will later see how these unknown parameters can be recovered using very little information, either on the cameras or on the scene.

3 Robust recovery of the geometry

In the M-situation, we select (presently manually but we plan to automate this process in the near future) a subset of the images in the sequence so that we end up in the S-situation. The important difference is that the intermediate images can be used, as explained below, to simplify the process of establishing correspondences between the views.

3.1 Obtaining correspondences between images

The algorithm used to compute the projection matrices needs correspondences and a few epipoles in order to work. We first obtain feature points using a simple corner detector [20] and we refine their position using a model-based approach [3].

In the S-situation, we then establish correspondences between the corners using grey-level correlation between neighboring regions of those feature points. For a

given point in one image several candidate matches are in general possible in another image. In order to reduce the number of hypotheses, we make use of a relaxation method [52]. Figure 3(bottom) shows a subset of the correspondences which have been automatically obtained between the images shown on the top row.

Figure 3: approximately here.

In the M-situation we track the feature points in the sequence whenever possible (small motion between frames). If a given point can be tracked all the way between two of the selected views, a correspondence is established. Tracking of a point of interest is performed by predicting its position from one image to the next and searching in a small neighborhood of the predicted point for an actual point. Like above, window-based correlation is used to discriminate between potential candidates. An example of such a tracking is shown in Figure 4.

Figure 4: approximately here.

3.2 Estimating the fundamental matrices between pairs of images

At this stage, we have obtained a number of correspondences between some images. Correspondences between pairs of images are input to a program called *Image-Matching* that reliably and robustly estimates the fundamental matrices between those pairs [52]. This program has the capability of rejecting some of the correspondences as outliers. The Image-Matching executable is available at
`ftp://krakatoa.inria.fr/pub/robotvis/BINARIES`.

3.3 Estimating the uncertainty of the fundamental matrices

The uncertainty associated with the points of interest (typically between 0.1 and 1 pixel) is propagated to the fundamental matrices. In order to compute an estimate of this uncertainty, we parameterize the fundamental matrix with the minimum number of parameters, namely 7, and compute the covariance matrix of the corresponding vector of size 7. There are several technical difficulties in doing this. First, the parameterization using 7 parameters is nonlinear, not unique, and has singularities. We therefore have to find the best one in the sense that it is the most remote from singularities. Second, the criterion which is minimized in order to estimate the fundamental matrix is also nonlinear and does not provide an analytical expression of the solution as a function of the point correspondences. We therefore have to use the implicit function theorem to actually compute the covariance matrix of the parameter vector of the fundamental matrix. The details of those computations can be found in [8].

3.4 Recovering the geometry of the N cameras

Up to this stage in the processing, we have estimated the fundamental matrices of consecutive pairs of images as well as obtained a number of point correspondences between the views. Nonetheless, we usually still have false matches to account for. The set of false matches for the epipolar geometry between pairs of images is a strict subset of the false matches for the projective geometry: this simply means that the images of a point can satisfy the epipolar geometry between pairs of images and still be incorrect when considering the complete set of images. The geometry estimation algorithm to be described next has been designed to deal with this problem.

Since the choice of a particular projective basis does not change the projective geometry of the scene, we are free to choose one for which the process of estimating the perspective projection matrices is the most stable numerically. We are going to use this property to compute our projection matrices. Using the theory developed in [11], from 5 points in correspondence in a pair of images and the epipoles, we can obtain the projection matrices, expressed in the projective basis defined by those points. This step is just a matter of writing equations of the type:

$$\mathbf{m}_{ij} = \mathbf{P}_j \mathbf{E}_i, i \in \{1, \ldots, 5\}, j \in \{1, 2\} \tag{3}$$

where the $\mathbf{E}_i$ represent the canonic 3D projective basis:$[0,0,0,1]^T$, $[0,0,1,0]^T$, $[0,1,0,0]^T$, $[1,0,0,0]^T$, $[1,1,1,1]^T$. The 20 scalar equations given by (3) need to be complemented by two equations exploiting the fact that the epipoles are known.

In order to obtain a set of projection matrices, 5 points in correspondence in the N images are selected automatically (We show in section 3.4.2 how to proceed when five correspondences cannot be found over the whole sequence). We then proceed pairwise. For each consecutive pair of images, we compute the projection matrices in the basis of the 5 chosen points. For this, we make use of the coordinates of the points in the images and of the coordinates of the epipoles that were determined by the estimation of the fundamental matrices. Of course, there can be conflicts: the projection matrix $\mathbf{P}_j$ computed from the pair $(j-1, j)$ can be different from the one computed with $(j, j+1)$. Note that this can only be due to the epipoles, because the coordinates of the 5 points remain unchanged. We do not consider this as a major problem because they are usually not very different and because this initial estimate is just a starting point for a refinement procedure. Only one of the possible projection matrices is kept.

Of course, running through this process only once has very little chance to succeed because of the possible outliers. If one of the matches is erroneous, then the projection matrix of the corresponding camera and its neighbors will be useless. Note that all other matrices will be correct. This quality of localness is desirable and cannot be achieved with iterative (image after image) techniques. To overcome this problem, we use robust methods.

3.4.1 Least Median of Squares

The Least Median of Squares (LMedS) is a classic method in outlier detection. A very good introduction can be found in [44]. We need a quality measure of the set of

projection matrices for each point. We define r_i as the sum taken over all cameras of the image distances between the ith measured point and the reprojection of the 3D reconstructed point with these projection matrices. In detail

$$r_i = \sum_{j=1}^{N} d(\mathbf{m}_{ij}, \mathbf{P}_j \mathbf{M}_i) \tag{4}$$

$\mathbf{M}_i$ is obtained with the reconstruction algorithms mentioned in section 4. The LMedS method estimates the parameters, i.e. the projection matrices, by solving the non-linear minimization problem:

$$\min(\text{med}_i(r_i^2)) \tag{5}$$

That is, the estimator must yield the smallest value for the median of squared residuals computed for the complete set of points. Of course, it is not reasonable to generate all the possible subsets of 5 point correspondences. Rather, we use a Monte-Carlo technique [44] to draw m random subsamples of $p = 5$ different point correspondences. For each subsample J, we estimate the set of projection matrices $\mathbf{P}_j^J$ by the methods previously described. For each set $\mathbf{P}_j^J, j = 1, .., N$, we can determine the median of the squared residuals denoted $M^J = \text{med}_i(r_i^2)$, with respect to the whole set of point correspondences. We retain the estimate of the $\mathbf{P}_j$ leading to the minimal M^J. Given this set of projection matrices, we characterize as outliers the point correspondences for which $r_i > \sigma$, where σ is an estimate of the variance derived from the data [44].

The question now is: *how do we determine* m*?* A subsample is considered good if it consists of p correct correspondences across the N images. Assuming that the probability of a point correspondence across 2 images being an outlier is ϵ, the probability of a point correspondence across the N images being an outlier is $1 - (1 - \epsilon)^{N-1}$. The probability that at least one of the m subsamples is good is given by

$$P = 1 - (1 - (1 - \epsilon)^{(N-1)p})^m \tag{6}$$

In our implementation, we assume that $\epsilon = 15\%$ (in practice a slightly over-estimated value), and require $P = 0.99$, thus $m = 6907$. Note that the algorithm can be sped up by means of parallel computation, because the processing for each subsample is done separately.

The five points of a subsample may be very close to each other. Such a situation should be avoided because the estimation of the 3D structure from such a projective basis is highly unstable and the result is useless. It is a waste of time to evaluate such a subsample. Bucketing techniques were developed to ensure that such configurations are avoided. The images are evenly divided into rectangular regions or *buckets* (in practice, we use 8×8 buckets), and we impose that the points be drawn from distinct buckets. The previous formula determining m still holds under the assumption that the outliers are uniformly distributed over the image.

3.4.2 Block estimation

Over a long sequence, it is very difficult or even impossible to find correspondences for the same five points across the whole sequence of images. We therefore split our estimation process over different consecutive blocks of images, with the precaution that the intersection of two consecutive blocks of images contains at least two images. Knowing the projective bases used in each such pair of blocks, we can compute the projective transformation from one to the other and apply it to the matrices of the second block. This process glues the blocks together. It allows processing of image sequences where groups of five point correspondences can be found for any consecutive three images (at least), instead of the whole sequence. This assumption is in practice very reasonable.

3.5 Refinement

Once a correct set of projection matrices has been computed, we can refine it using three different methods. Of course, the outliers found at the previous step are marked as invalid and are not taken into account any further.

Bundle adjustment: This classical method in photogrammetry [5, 6, 18, 19, 47] is very well suited to our problem. It is based on the observation that due to errors in the estimation of the projection matrices and on the positions of the 2D points, the optical rays issued from corresponding image points do not intersect in space, though in an ideal configuration they should all intersect at one single 3D point. Through non-linear optimization over the projection matrices and the reconstructed 3D points, the bundle adjustment tries to bring the system as close as possible to the ideal situation.

With our initial estimation, the optical rays used in the method approximately intersect because the reprojections of the 3D points are close to the initial points. This method has the advantage of being fast. The only modification that we have made is that instead of reconstructing the actual Euclidean 3D points, we reconstruct the points in a projective basis. Although our problem is over-parameterized (we allow the projective basis in space to change), the minimization converges because of the nice properties of the Levenberg-Marquardt algorithm. The average distance between one point and the reprojection of its reconstruction is initially around 1 pixel and typically goes down to 0.3 pixels.

Epipolar line adjustment: From the set of projection matrices, we can compute a consistent set of fundamental matrices. The points that we have matched must satisfy the epipolar constraints. We then minimize the sum of the distances between the points and the epipolar lines generated by their correspondences by varying the projection matrices. However, this method is slow because we have to recompute the epipolar lines at each step. In other words, there is no possible decoupling of the minimization because of its high non-linearity.

Trifocal adjustment: From the set of projection matrices, we can compute the trilinearities which relate point coordinates in any three images of the sequence (cf

section 2.1. Given three images and two points measured in the first two images, each trilinear relation defines one line in the third image. The set of all trilinearities defines nine lines in the third image [28], which would all intersect at the corresponding point if the point locations were noiseless. The criterion which we measure is the sum of the squared distances of the point measured in the third image to these lines, computed from the points in the first two images. The complete residual is the sum of this criterion over all point correspondences, for all image triples.

Comparison of the three methods: We compared the three methods on synthetic image points, perturbed with Gaussian noise. Figure 5 summarizes the results. We draw the following conclusions:

- Errors have the same order of magnitude in all cases.
- Epipolar adjustment is more robust to noise, then bundle adjustment, then trifocal adjustment.
- Convergence is slower for trifocal adjustment.
- The epipolar line adjustment is slower than the bundle adjustment but performs best in the bad cases when the set of projection matrices is not very well initialized, except when at least three optical centers are close to being aligned.

Figure 5: approximately here.

Figure 6 shows the epipolar geometry obtained through bundle adjustment on a set of aerial images. Computation took 6.3 minutes for 10000 projective bases tried for initialization on a *Sun Sparc 20* workstation. The average image error was less than 0.3 pixels on the 183 points used for the refinement.

Figure 6: approximately here.

4 Recovering Euclidean structure

As mentioned in section 3, the projection matrices that we have determined allow us to compute three-dimensional structure from image correspondences, *up to an unknown projective transformation*. This fact had been first stated in [27] in the case of two affine cameras (in this case, the unknown transformation in space is affine). The case of two projective cameras has been presented in [11, 24].

From a practical standpoint, this means that from image correspondences, we can compute three-dimensional points represented by their homogeneous coordinates (4-vectors). Knowing the coordinates of a point, we can back-project it onto any of the cameras for which a projection matrix has been computed. We can even project it onto an arbitrary virtual camera: this way we can produce new views of the scene. This process, usually called view transfer [2, 50], has been used for image synthesis[13, 29]. However, for simple reasons, the projective reconstruction

may not be sufficient: for instance, to perform realistic rendering and virtual walk-through on today's fast-rendering hardware, one needs a Euclidean description of the scene.

Without any additional information, recovering Euclidean structure is impossible: all the geometric relations induced by point correspondences have been already used. We need to use additional information, either on the viewing system, or on the scene.

4.1 Self-calibration of a moving camera

Approaches have been developed which deal with the case when the intrinsic parameters of the camera do not vary over a sequence of three images or more [38, 33, 14]. In this case, these parameters can be computed from a number of point correspondences in the three views by solving the so-called Kruppa equations. This approach tends to be sensitive to noise. Only recently have we been able to produce an implementation which is robust and can deal with an arbitrary (greater than three) number of images [51]. With input images of good quality (small non-linear distorsion, high resolution, typically, we use 1536×1024 *Photo-CD* images) it is now completely practical and accurate if uncertainty is correctly taken into account as shown in [8].

4.2 Self-calibration using information on the environment

If the intrinsic parameters of the camera are not constant throughout the sequence (e.g. pictures are taken with several cameras), we can still recover Euclidean structure based on some information about the scene:

- If we know the coordinates of at least five reconstructed points in general configuration (i.e., a projective basis) with respect to a Euclidean frame, we can compute the projective transformation which changes projective coordinates into Euclidean ones. This principle of using a few "anchor points" to derive Euclidean coordinates is commonly used in photogrammetry [19]. It supposes that one has performed manual measurements on the real scene, which is rather constraining.
- A Euclidean frame can be characterized as a frame where parallel lines intersect at infinity, and where orthogonal lines are indeed orthogonal (the dot-product of their direction is zero). The first property characterizes affine structure, whereas the second one characterizes Euclidean structure up to an unknown scale. As shown below, using images of parallel lines we can recover affine structure; Using pairs of orthogonal direction, we then reach scaled Euclidean structure. Less restrictive than the anchor point approach (here, no manual measurement is performed), this approach is perfectly suited to the requirements of *Realise*, because there are in general many images of parallel or orthogonal lines in views of buildings.

Let us now examine in more detail this last approach. In the first stage, we recover affine structure, in which parallelism is preserved. In the second one, we recover Euclidean structure in which orthogonality is preserved as well.

After the first stage described in the previous section, the world is modeled as a three-dimensional projective space $\mathcal{P}^3$. We use the standard embedding of a three-dimensional affine space $\mathcal{A}^3$ into $\mathcal{P}^3$ obtained by identifying $\mathcal{A}^3$ with $\mathcal{P}^3 \setminus \Pi_\infty$, where Π_∞ is a plane, called the plane at infinity, which can be thought as the set of directions of lines in $\mathcal{A}^3$. In particular, two parallel lines of $\mathcal{A}^3$, seen as lines of $\mathcal{P}^3$ intersect at a point of Π_∞ (called their point at infinity). Such a point is not as mysterious as it sounds since when viewed by a camera, the images of the two lines usually intersect at a point (called their vanishing point) which can be thought of as (in fact in some sense *is*) the image of the point at infinity of the two lines. Hence, in order to determine the plane at infinity, it is in principle sufficient to have in the scene three pairs of non coplanar parallel lines. Once the plane at infinity has been determined, an affine coordinate system can be chosen and affine coordinates of the points in the scene computed.

In the remainder of this article, we use the standard embedding of $\mathcal{A}^3$ into $\mathcal{P}^3$ which maps a point of affine coordinates $[x, y, z]^T$ onto its corresponding point of $\mathcal{P}^3 \setminus \Pi_\infty$ of homogeneous coordinates $[x, y, z, 1]^T$. This embedding simply means that the plane at infinity is the plane of equation $T = 0$ in the projective space with homogeneous coordinates X, Y, Z, T.

4.2.1 Parallel lines: affine structure

Two lines in space are parallel if and only if they intersect each other in the plane at infinity. A projective transformation preserves affine structure if and only if preserves parallelism, which means that it leaves the plane at infinity (the set of all points at infinity) globally invariant.

Thus, the problem of recovering affine structure is equivalent to finding a projective transformation H_a which maps the plane at infinity onto the plane represented by $[0, 0, 0, 1]^T$. This is a very simple operation provided that we can compute the coordinates of the plane at infinity in the initial projective frame. For this purpose, we first need to determine at least three non-aligned points on this plane, i.e. three non coplanar directions of lines. Since we observe images of lines, each of these points is computed as the vanishing point of a set of parallel lines observed in the images. This is shown in Figure 7.

Figure 7: approximately here.

In this figure the three pairs of lines (D_1, D_2), (D'_1, D'_2), and (D''_1, D''_2) are respectively parallel and the images of their points at infinity V, V', V'' are the points of intersection v, v', v'' of the pairs of image lines (d_1, d_2), (d'_1, d'_2), (d''_1, d''_2), respectively.

From a practical standpoint, parallel directions are identified in the images by the user in a semi-automatic process. First, for each image, we compute a polygonal approximation of the edge chains extracted with a sub-pixel feature detector. The user then selects a small number of line segments (at least 3, chosen in at least two distinct images) representing parallel lines in space. Based on these, the program computes a first estimate of the corresponding point at infinity in space and back-projects it onto all the images. All the image line segments whose supporting lines are close enough

to the vanishing point (a simple threshold on the angular criterion described below is used) are proposed to the user. After potentially editing the program's selection, the user runs the complete computation of the vanishing point. Figure 8 shows for one image the line segments with which points at infinity have been computed.

Figure 8: approximately here.

Computing points at infinity: We compared several methods for estimating the points at infinity given images of parallel lines in space. Let us assume that we measure in the images the projections of parallel space lines $\langle D_i \rangle$. $\langle d_{ij} \rangle$, represented by the three-dimensional homogeneous vector $\mathbf{d}_{ij}$, is the image of $\langle D_i \rangle$ in view j. We want to compute their point of intersection $\mathbf{V} = \bigcap_i \langle D_i \rangle$ which we know to be in Π_∞. The image $\mathbf{v}_j$ of $\mathbf{V}$ in view j is the vanishing point of the image lines $\mathbf{d}_{ij}$. The problem that we need to solve is the following: given $\mathbf{d}_{ij}$, compute $\mathbf{V}$.

A first method consists of first estimating the vanishing points $\mathbf{v}_j$, then reconstructing V from the obtained vanishing points. $\mathbf{v}_j$ is obtained as the weighted-least-squares solution [17] of the homogeneous system :

$$\forall i \qquad \mathbf{d}_{ij}^T \mathbf{v}_j = 0$$

This process turns out to be very sensitive to noise, due to the fact that vanishing points are estimated independently in the different images without enforcing the epipolar constraints on them. For this reason, we prefer the following direct linear method: since the image of $\mathbf{V}$ in camera j lies on line $\mathbf{d}_{ij}$, we have:

$$\forall i, j \qquad \mathbf{d}_{ij}^T \mathbf{P_j} \mathbf{V} = 0.$$

This system of linear, homogeneous equations in the four homogeneous coordinates of $\mathbf{V}$ is solved using SVD.

This estimate is then used as initial estimate of a Nonlinear Least Squares minimization (Levenberg Marquardt). The criterion measured for each measured line segment is the minimum angle between its supporting line and one of the two lines joining its extreme points to the projected point at infinity. The sum of squared angles over all line segments is the minimized value.

Computing the plane at infinity: The previous process can be applied to all the directions for which parallel lines are observed, yielding points at infinity $\mathbf{V}_k$. Provided that there are at least three non-aligned points at infinity, we can compute the plane at infinity Π_∞, represented by the four-dimensional homogeneous vector $\mathbf{\Pi}_\infty$, as the non-zero solution of the linear homogeneous system:

$$\forall k \qquad \mathbf{V}_k^T \mathbf{\Pi}_\infty = 0$$

Once we know $\mathbf{\Pi}_\infty$, we can compute the point at infinity of any line as long as this line can be reconstructed in space, by computing the intersection of this line with the plane at infinity.

Deriving an affine reconstruction: To define the transformation which maps the plane at infinity onto $[0,0,0,1]^T$, we proceed as follows:

First, we compute a projective reconstruction of the scene, using standard multi-camera reconstruction based on SVD (cf. section 2.3).

One reconstructed point of the scene, denoted by $\mathbf{C}$, is chosen as the origin: in the new frame, it has coordinates $[0,0,0,1]^T$. Then, three arbitrary independent directions are selected as coordinate axes, and their points at infinity $\mathbf{V}_X, \mathbf{V}_Y, \mathbf{V}_Z$ are computed using the method described above. They are respectively mapped onto $[1,0,0,0]^T$,
$[0,1,0,0]^T$, $[0,0,1,0]^T$. To define a projective transformation in space, we need a fifth point mapping. We select another reconstructed point $\mathbf{S}$, which does not lie on any of the three planes defined by the origin and two of the three axes. In the new frame, this point is assigned arbitrary non-zero coordinates $[\alpha,\beta,\gamma,1]$. The choice of the two points, the three directions and the coefficients $\alpha,\beta,\gamma]$ is completely automatic. By default, all the coefficients are set to 1. However, the user may override this and enter his own selection of points, lines and/or coefficient values.

$\mathbf{H}_a$ is then computed as the projective transformation which maps the initial projective basis $\mathbf{V}_X, \mathbf{V}_Y, \mathbf{V}_Z, \mathbf{C}, \mathbf{S}$ onto the final one. Since $\mathbf{V}_X, \mathbf{V}_Y, \mathbf{V}_Z$ are all mapped onto points whose fourth component is zero, any point of the plane at infinity will also be mapped onto a point whose fourth component is zero, which is precisely what is needed for the reconstruction to be affine. Figure 9 shows an example of affine reconstruction. Two of the three directions chosen as coordinate axes form a very small angle (left). This implies a strong affine skew effect on the reconstructed scene (right).

Parameters α,β,γ have a simple meaning: they represent scale factors along the three coordinate axes. For instance, if α is multiplied by a non-zero factor, then the reconstructed scene will be stretched by the same scale factor along the $\mathbf{V}_X$ axis. This is visible in Figure 11, which displays two affine reconstructions which differ by only one scale factor (along the direction of the top edge of the roof).

Figure 9: approximately here.

4.2.2 Euclidean structure up to three scale factors

As we have seen in Figure 9, the choice of non-orthogonal reference directions may cause severe affine distortion of the reconstructed scene. A first step toward the recovery of Euclidean structure is to use three pairwise orthogonal directions.

In this case, illustrated by figure 11, the directions of edges parallel to the reference directions are preserved. In fact, the recovered structure is equivalent to Euclidean structure scaled with three scale factors along the three coordinate axes. As a consequence, two edges aligned with two orthogonal coordinate axes remain orthogonal in the final affine reconstruction, for any value of the scale parameters α,β,γ (e.g. previous section). This is for instance the case of the roof on which the two horizontal directions have been defined (left). The relative values of the scale factors used for the two displayed affine reconstructions (middle,right) are very different.

This modifies drastically the aspect of the reconstructed roof (at the bottom-right in each view), but the principal directions remain orthogonal in both reconstructions.

Of course, angles between lines which are not aligned with the coordinate axes are not preserved. In particular, orthogonality is not preserved for such directions (see the roof on the bottom-left). We will now see how this property can be used for recovering Euclidean structure up to one global scale factor.

4.2.3 Euclidean structure up to one scale factor

We now assume that some pairs of orthogonal lines are known a priori. The points at infinity of these lines, $\mathbf{V}_i$, are computed as described above. In a Euclidean frame, lines i, j are orthogonal if and only if $\mathbf{V}_i^T \mathbf{V}_j = 0$.

If we consider the orthogonal frame defined in the previous paragraph, finding Euclidean structure is equivalent to finding relative values of the scale parameters α, β, γ for which the dot-products $\mathbf{V}_i^T \mathbf{V}_j$ are zero for all pairs (i, j) of orthogonal lines.

If the three reference axes used for affine reconstruction are not orthogonal, three additional parameters ("skew" parameters) are introduced which account for the non-orthogonality of the reference affine frame. More precisely, instead of using the mapping defined in 4.2.1, we respectively map $\mathbf{V}_X, \mathbf{V}_Y, \mathbf{V}_Z$ onto $[1, 0, 0, 0]^T$ (this has not changed), $[\lambda, 1, 0, 0]^T$, $[\mu, \nu, 1, 0]^T$.

We end up with the following criterion to be minimized over the scale parameters and the skew parameters: :

$$E(\alpha, \beta, \gamma, \lambda, \mu, \nu) = \sum_{i,j \text{ orthogonal}} (\mathbf{V}_i^T \mathbf{V}_j)^2$$

The global scale of the scene cannot be recovered. So, we search for the particular solution for which $\alpha = 1$. Minimizing $E(1, \beta, \gamma, \lambda, \mu, \nu)$ with the standard Levenberg-Marquardt iterative technique (the initial values of the five parameters are the ones used for affine reconstruction), we end up with a Euclidean reconstruction up to a global scale factor (see Figure 12).

Thus, we now have a way of computing a Euclidean reconstruction of the scene without any knowledge of the camera parameters, nor of the scene coordinates. This is a major difference with the method presented in [9], where knowledge of the intrinsic parameters of the camera is required. Only informations about point and line matches, parallelism and angular relations have been used. Moreover we can even in some cases obtain a Euclidean reconstruction without using this information, by self-calibration [8]. In practice we have found it useful to combine *all* the available information to obtain robust results. Once the mappings which bring points from projective to affine and Euclidean space have been computed, the projection matrices are updated so that image point correspondences are directly reconstructed in the Euclidean space.

The structure of the process that recovers the Euclidean structure of the scene is shown if figure 10 which clearly shows that this can be done in two ways, either through the use of a priori geometric information about the scene, e.g. parallel lines, angles and ratios of lengths, or through the use of a priori information about the

internal parameters of the cameras, e.g. in the most general case that they are constant but unknown.

Figure 10: approximately here.

Figure 11: approximately here.

Figure 12: approximately here.

4.3 Building a polyhedral description

The problem of constructing a polyhedral representation of the scene is by no means trivial and we have in fact not attempted to solve it directly from scratch. What we have done is to extend the capabilities of a number of modellers to allow them to use the 3D information provided by a set of images. The main advantage of this approach is that we benefit from all the functionalities which are available in currently available 3D modellers, e.g. levels of representations, interaction, display, bookkeeping and simply enhance them with the ability to use the state of the art computer vision procedures, some of them described in this paper.

In slightly more detail, the scene is reconstructed as a set of polygons in space. The user defines the topology of the model, and is helped by the system in defining its geometry. This is done at several levels, which differ by their degree of interaction.

At the lowest level, the user defines a 3D vertex by specifying its position in two or more images. In this process, he gets visual help from the system which can display epipolar lines, perform trifocal transfer, etc...

At a slightly higher level, the user can restrict his or her attention to only one image: typically he or she will select or outline roughly a visually prominent detail such as a vertex, a line, or a closed contour and the vision server will use the currently available calibration information to obtain correspondences for these features in other images, perhaps after some refinement of their position in the original image, thereby providing 3D information to the modeller and usually also an update of the calibration. At this stage several vision algorithms may be selected and put in competition by the system which can evaluate their results according to a variety of criteria ranging from a simple consideration of the uncertainty and completeness of the results to a more elaborate criticism of how they fit with the current model.

Because this part of the system is presently not completely stabilized, we prefer to postpone its full discussion to a future paper.

5 Texture extraction

Once a polyhedral description of the scene has been derived, our system automatically extracts from the images the textures attached to the polygons. For each planar

polygon, we compute a rectified *texture image*, i.e., a bitmap related to a system of coordinates attached to the supporting plane of the polygon, and the corresponding two-dimensional *texture coordinates* of the polygon vertices.

In the favorable cases, an obstacle-free, complete texture can be extracted from one single image. However, it often happens that the texture is not completely visible in any of the images: Some part of the polygon may project outside of the image, or there might be visual obstacles in front of the polygon. To address these problems, we developed a technique inspired from mosaicing [48, 25]. It is based on the fact that since the polygon is planar, it induces homographic (linear, projective) correspondences between the images. The approach consists of first computing the homographies – in practice, homogeneous 3×3 matrices – from the texture image to each view, then generating the texture image. At each pixel, the computed texture value is a combination of the values read in the different images at the corresponding pixels.

Let us now describe in more detail all the stages involved in the texture computation of one polygon.

5.1 Texture coordinates

First, a reference image is selected. By default, the system chooses the image in which the polygon is observed from the angle closest to its normal. Let us denote by $\mathbf{P_r}$ its corresponding projection matrix.

Second, a normalized coordinate system is chosen for the texture image. For this, we introduce a new orthonormal system of coordinates (O', X', Y', Z') such that the Z' axis is normal to the plane of the polygon, the X' axis is aligned with the longer edge of the polygon, and the origin O' lies in the plane of the polygon.

In this new coordinate system, the equation of the plane of the polygon is $Z' = 0$. In other words, if the change of coordinates corresponds to the rigid transformation $(\mathbf{R}, \mathbf{t})$, for any point (X, Y, Z) belonging to the plane of the polygon we have

$$[X', Y', 0, 1]^T = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_3^{\mathrm{T}} & 1 \end{bmatrix} [X, Y, Z, 1]^T$$

Coordinates X', Y' describe the position of the point within the polygon plane. They are rescaled (scale factors k_u, k_v) and shifted (translation δ_u, δ_v) in order for the transformed polygon vertices to have coordinates between 0 and 1. The resulting *normalized texture coordinates*, denoted by u_n, v_n, satisfy the following relation:

$$[u_n, v_n, 1]^T = \begin{bmatrix} \kappa_u & 0 & \delta_u \\ 0 & \kappa_v & \delta_v \\ 0 & 0 & 1 \end{bmatrix} [X', Y', 1]$$

Thus, the space coordinates of a point lying on the plane of the polygon can be computed from its normalized texture coordinates:

$$[X, Y, Z, 1]^T = \mathbf{K}[u_n, v_n, 1]^T \quad \text{with } \mathbf{K} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_3^{\mathrm{T}} & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \kappa_u & 0 & \delta_u \\ 0 & \kappa_v & \delta_v \\ 0 & 0 & 1 \end{bmatrix}^{-1}$$

The homography which transforms normalized texture coordinates into coordinates in the reference image is then:

$$\mathbf{H}_r = \mathbf{P}_r \mathbf{K}.$$

The resolution of the texture image can be chosen arbitrarily. We constrain the polygon in the texture image to have the same area as the polygon in the reference image, which implies that both polygon textures will approximately have the same level of detail. We finally derive the homography $\mathbf{H}_r$ which maps a point in the texture image onto its corresponding point in the reference image.

5.1.1 Accurate image-image registration

From the projection matrices and the three-dimensional coordinates of the polygon vertices, we can easily derive the homography, denoted by $\mathbf{H}^0_{ri}$, from the reference image to any image i. However, there might be small errors in the image-to-image registration, due to slight imprecision in the calibration and reconstruction processes. These errors are typically of the order of one or two pixels. To address this problem, we use the following refinement technique: We extract Harris corners [20], which we denote by $\mathbf{c}_k$. The homography gives us a pixel-wise correspondence between a square window centered at $\mathbf{c}_k$ in the reference image, and a skewed window around $\mathbf{H}^0_{ri}\mathbf{c}_k$ in image i. Translating the skewed window within a small neighborhood of this point, we find the point $\mathbf{c'}_{ik}$ for which cross-correlation of the intensity distributions within the windows is maximal. Finally, we compute $\mathbf{H}_{ri}$ which minimizes

$$\mathrm{med}\left(d(\mathbf{c'}_{ik}, \mathbf{H}_{ri}\mathbf{c}_k)^2\right)$$

where $d(.,.)$ is the image distance (in pixels). The least-median-of-squares estimator allows us to get rid of the wrong point correspondences caused by visual obstacles.

5.1.2 Compensation for intensity variations

In a second stage, we compensate for global intensity and contrast differences across the images. For this purpose, we extract a number of points $\mathbf{e}_k$ in the reference image, and we estimate the affine function which best maps intensities $I_r(\mathbf{e}_k)$ in image r onto intensities $I_i(\mathbf{H}_{ri}\mathbf{e}_k)$ in image i. Once again, we use a least-median-of-squares estimator to get rid of outliers. Here, it is better to choose points $\mathbf{e}_k$ such that intensity varies slowly around them. In practice, we compute the points which locally minimize the module of the Harris criterion.

5.1.3 Texture image generation

Finally, we generate the texture image. For each pixel in the texture image, we compute the intensities at the corresponding pixels in the images, using the estimated homographies and the affine functions for intensity adjustment. We obtain a vector of intensity values $I_1, ..., I_n$. In an ideal case, e.g. lambertian surfaces, constant lighting conditions and constant sensitivity of the imaging device, all these should

be equal, except for values corresponding to visual obstacles. So, we compute the subset of p values among the n which have minimum variance (typically, we use $p = 2$ for $n \in \{3, 4\}$, $p = 3$ for $n > 4$ among n), and set the texture pixel value to the median of these p values.

The same computation can be extended to color images in a straightforward manner, by considering three-component values for the intensities. Figure 13 shows an example of texture generated from 4 images.

Figure 13: approximately here.

6 Results

The whole system was developed according to the server/client architecture mentioned in section 1. At Inria, we developed the vision procedures, which were integrated within the vision server by our partners from Thomson-Syseca. Our partners from the Fraunhofer Institut (Darmstadt) developed a client, i.e., an interactive image-oriented 3D modeler capable of requesting information from the vision server. A brief description of the system and its applications to virtual reality is given in [31]. In order to test the vision procedures, we developed another client based upon the *TargetJr* image understanding environment developed by GE. In the current system, the vision procedures which are used in order to help the end-used are the following:

- Features such as corners, edges, line segments are accurately localized using a model-based approach [3].
- When building primitives such as polygons or polyhedra, we can rely upon edge matches automatically produced by a multi-image curve matching algorithm based on [42]. This speeds up the modeling process by allowing us to define accurate 3D objects while interacting with only one image.
- In cases when the previous method fails, the user can still build the model manually. He is assisted in this task by the fact that he can observe simultaneously in all images the effects of his actions (display of the epipolar lines corresponding to the cursor position, or for a fixed optical ray, display in all the images of the corresponding points along the epipolar lines). This relies on epipolar and trifocal geometry.
- Textures are automatically extracted from the images.

6.1 Qualitative assessment of the results

The system has now been run on many sets of images to reconstruct tens of models. We only show here two examples of three-dimensional wireframe and textured models that we have produced. In figures 14 (respectively 15), the top row shows two images of the 10-image (respectively 15-image) sequence, with the reconstructed wireframe model superimposed; the bottom row shows two views of a reconstructed textured model. Of course, the models have been obtained from all the views, not

only the two shown (in the case of the wireframe example), which explains why some of the features in the wireframe model may be hidden in the chosen views.

Figure 14: approximately here.

Figure 15: approximately here.

6.2 Quantitative assessment of the results

Beside the qualitative assessment of the quality of the produced models which was presented in the previous section, we have also tested our results against actual measurement. For example, we show in figure 16 a number of line segments in one of the images of the "Arcades" sequence. These segments correspond to actual physical features in the scene whose lengths as well as some of their angles we could measure. Table 1 shows a comparison of the real, i.e. measured, lengths and angles with the estimated ones. In the case which is presented here, the Euclidean structure of the scene was obtained from self-calibration, the internal parameters of the camera being constant but unknown (see section 4.1). Similar results are shown for the

Figure 16: approximately here.

"Church" sequence in figure 17 and table 2. In both cases the results are satisfactory, considering the fact that no a priori information about the scenes has been used.

7 Conclusion

We have described in this article the skeleton of a system based on computer vision that is going to be used to partially automate the 3-D CAD modeling of urban scenes. The system can use any number of cameras and images of the scenes to be modeled and proceeds to estimate automatically the perspective projection matrices corresponding to all the images by matching image features such as corners, junctions, lines. The resulting matrices do not in general allow recovery of a metric model of the scene since no metric information has been used so far, only a projective one which can be used for some applications. In order to go further, the system can do two things: 1) make some assumptions about the internal parameters of the cameras, for example that they do not significantly vary over at least three frames and use the self-calibration methods introduced in [38, 33, 14] and recently made more robust to image noise [8] and 2) use information provided by the user about the actual affine or Euclidean structure of the scene, such as parallel lines, ratios of lengths, and angles. This information allows the system to specialize its representation of the environment from projective to affine and finally Euclidean. The whole system uses sophisticated computer vision tools and has been developed as a flexible

Table 1: approximately here.

Figure 17: approximately here.

server, a vision server, that can be queried by a human user who is using a CAD system to develop a 3-D textured model of the scene.

One of the advantages of this system is that it does not require any prior knowledge about the cameras, which is handy in applications like video-based modeling for example. The user is then allowed to use his camera the way he likes, without any special set-up, or to use images of unknown source. This is in sharp contrast with systems that have been proposed in the past by other groups. We have already discussed in section 4.2.3 why we thought that our system was more flexible than the one described in [9]. We should also mention the fact that it is much more flexible and robust than the ones that could possibly be built on top of the results described in [41] and [7]. There, the authors can only cope with one calibrated affine camera and use structure from motion as the basic ingredient for recovering 3D. As shown in this paper, we can deal with uncalibrated cameras performing full perspective projection and use large baseline stereo for recovering 3D. We have found in particular that for the kinds of views shown in figures 14 and 15, the perspective effects were quite large and hard to account for with an affine camera model. This is of course at the cost of adding some limited human interaction in the system but we think that it buys us a lot more accuracy and robustness.

We believe that neither the theory, nor the technology are ready for fully automatic 3D modellers but we are convinced that the time has come to build and sell systems that offer unsophisticated users the possibility to use interactively highly sophisticated computer vision tools.

Acknowledgement
The authors would like to acknowledge the fact that the work presented here would not have been possible without the contributions of Thierry Blaszka, Rachid Deriche, Charlie Rothwell, and Zhengyou Zhang.

Table 2: approximately here.

References

[1] K.B. Atkinson, editor. *Close Range Photogrammetry and Machine Vision*. Whittles Publishing, 1996.

[2] Eamon B. Barrett, Michael H. Brill, Nils N. Haag, and Paul M. Payton. Invariant linear methods in photogrammetry and model-matching. In Joseph L. Mundy and Andrew Zimmerman, editors, *Geometric Invariance in Computer Vision*, chapter 14. MIT Press, 1992.

[3] Thierry Blaszka and Rachid Deriche. Recovering and characterizing image features using an efficient model based approach. Technical Report 2422, INRIA, November 1994.

[4] Boubakeur Boufama and Roger Mohr. Epipole and fundamental matrix estimation using the virtual parallax property. In *Proceedings of the 5th International Conference on Computer Vision*, pages 1030–1036, Boston, MA, June 1995. IEEE Computer Society Press.

[5] Duane C. Brown. A solution to the general problem of multiple station analytical stereotriangulation. Technical Report 43, RCA Data Reduction Technical Report, Patrick Air Force base, Florida, 1958.

[6] Duane C. Brown. Close-range camera calibration. *Photogrammetric Engineering*, 37(8):855–866, 1971.

[7] J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. In *Proceedings of the 5th International Conference on Computer Vision*, pages 1071–1076, Boston, MA, June 1995. IEEE Computer Society Press.

[8] Gabriella Csurka, Cyril Zeller, Zhengyou Zhang, and Olivier Faugeras. Characterizing the uncertainty of the fundamental matrix. *CVGIP: Image Understanding*, 1997. To appear.

[9] P.E. Debevec, C.J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *SIGGRAPH*, pages 11–20, New Orleans, August 1996.

[10] Umesh R. Dhond and J.K. Aggarwal. Structure from stereo - a review. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1489–1510, 1989.

[11] Olivier Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig. In G. Sandini, editor, *Proceedings of the 2nd European Conference on Computer Vision*, volume 588 of *Lecture Notes in Computer Science*, pages 563–578, Santa Margherita Ligure, Italy, May 1992. Springer-Verlag.

[12] Olivier Faugeras. *Three-Dimensional Computer Vision: a Geometric Viewpoint*. MIT Press, 1993.

[13] Olivier Faugeras and Stéphane Laveau. Representing three-dimensional data as a collection of images and fundamental matrices for image synthesis. In *Proceedings of the International Conference on Pattern Recognition*, pages 689–691, Jerusalem, Israel, October 1994. Computer Society Press.

[14] Olivier Faugeras, Tuan Luong, and Steven Maybank. Camera self-calibration: theory and experiments. In G. Sandini, editor, *Proc 2nd ECCV*, volume 588 of

Lecture Notes in Computer Science, pages 321–334, Santa Margherita Ligure, Italy, May 1992. Springer-Verlag.

[15] Olivier Faugeras and Bernard Mourrain. On the geometry and algebra of the point and line correspondences between n images. In *Proceedings of the 5th International Conference on Computer Vision*, pages 951–956, Boston, MA, June 1995. IEEE Computer Society Press.

[16] Olivier Faugeras and Bernard Mourrain. On the geometry and algebra of the point and line correspondences between n images. Technical Report 2665, INRIA, October 1995.

[17] G.H. Golub and C.F. van Loan. *Matrix Computations*. The John Hopkins University Press, 1989.

[18] Armin Gruen. Accuracy, reliability and statistics in close-range photogrammetry. In *Proceedings of the Symposium of the ISP Commission V*, Stockholm, 1978.

[19] Armin Gruen and Horst A. Beyer. System calibration through self-calibration. In *Proceedings of the Workshop on Calibration and Orientation of Cameras in Computer Vision*, Washington D.C., August 1992.

[20] C. Harris and M. Stephens. A combined corner and edge detector. In *Proc. 4th Alvey Vision Conf.*, pages 189–192, 1988.

[21] R.I. Hartley. Euclidean reconstruction from uncalibrated views. In Joseph Mundy and Andrew Zisserman, editors, *Applications of Invariance in Computer Vision*, volume 825 of *Lecture Notes in Computer Science*, pages 237–256, Berlin, Germany, 1993. Springer-Verlag.

[22] Richard Hartley. Calibration of cameras using the essential matrix. In *Proceedings of the ARPA Image Understanding Workshop*, pages 911–915. Defense Advanced Research Projects Agency, Morgan Kaufmann Publishers, Inc., 1992.

[23] Richard Hartley. Lines and points in three views-an integrated approach. In *Proceedings of the ARPA Image Understanding Workshop*. Defense Advanced Research Projects Agency, Morgan Kaufmann Publishers, Inc., 1994.

[24] Richard Hartley, Rajiv Gupta, and Tom Chang. Stereo from uncalibrated cameras. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pages 761–764, Urbana Champaign, IL, June 1992. IEEE.

[25] M. Irani, P. Anandan, and S. Hsu. Mosaic based representations of video sequences and their applications. In *Proceedings of the 5th International Conference on Computer Vision*, pages 605–611, Boston, MA, June 1995. IEEE Computer Society Press.

[26] C.P. Jerian and R. Jain. Structure from motion. a critical analysis of methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3):572–587, 1991.

[27] Jan J. Koenderink and Andrea J. van Doorn. Affine structure from motion. *Journal of the Optical Society of America*, A8:377–385, 1991.

[28] Stéphane Laveau. *Géométrie d'un système de N caméras. Théorie, estimation et applications*. PhD thesis, École Polytechnique, May 96.

[29] Stéphane Laveau and Olivier Faugeras. 3-D scene representation as a collection of images and fundamental matrices. Technical Report 2205, INRIA, February 1994.

[30] M. Levoy and P. Hanrahan. Light field rendering. In *SIGGRAPH*, pages 31–42, New Orleans, August 1996.

[31] F. Leymarie, A. de la Fortelle, J. Koenderink, A. Kappers, M. Stavridi, B. van Ginneken, S. Muller, S. Krake, O. Faugeras, L. Robert, C. Gauclin, S. Laveau, and C. Zeller. Realise: Reconstruction of reality from image sequences. In P.Delogne, editor, *Proceedings of the International Conference on Image Processing*, volume 3, pages 651–654, Vienna, September 1996.

[32] Q.-T. Luong and T. Viéville. Canonic representations for the geometries of multiple projective views. In J-O. Eklundh, editor, *Proceedings of the 3rd European Conference on Computer Vision*, volume 1 of *Lecture Notes in Computer Science*, pages 589–599, Stockholm, Sweden, May 1994. Springer-Verlag.

[33] Quang-Tuan Luong. *Matrice Fondamentale et Calibration Visuelle sur l'Environnement-Vers une plus grande autonomie des systèmes robotiques*. PhD thesis, Université de Paris-Sud, Centre d'Orsay, December 1992.

[34] Quang-Tuan Luong, Rachid Deriche, Olivier Faugeras, and Théodore Papadopoulo. On determining the fundamental matrix: analysis of different methods and experimental results. In *Israelian Conf. on Artificial Intelligence and Computer Vision*, Tel-Aviv, Israel, 1993. A longer version is INRIA Tech Report RR-1894.

[35] Quang-Tuan Luong, Rachid Deriche, Olivier Faugeras, and Théodore Papadopoulo. On determining the fundamental matrix: Analysis of different methods and experimental results. Technical Report 1894, INRIA, 1993.

[36] Quang-Tuan Luong and Olivier Faugeras. An optimization framework for efficient self-calibration and motion determination. In *Proceedings of the International Conference on Pattern Recognition*, volume I, pages 248–252, Jerusalem, Israel, October 1994. Computer Society Press.

[37] Quang-Tuan Luong and Olivier Faugeras. The fundamental matrix: theory, algorithms, and stability analysis. *The International Journal of Computer Vision*, 17(1):43–76, January 1995.

[38] S. J. Maybank and O. D. Faugeras. A theory of self-calibration of a moving camera. *The International Journal of Computer Vision*, 8(2):123–152, August 1992.

[39] L. McMillan. Acquiring immersive visual environments with an uncalibrated camera. Technical Report TR95-006, University of North Carolina, 1995.

[40] S.I. Olsen. Epipolar line estimation. In *Proceedings of the 2nd European Conference on Computer Vision*, pages 307–311, Santa Margherita Ligure, Italy, May 1992.

[41] Conrad J. Poelman and Takeo Kanade. A paraperspective factorization for shape and motion recovery. In J-O. Eklundh, editor, *Proceedings of the 3rd European Conference on Computer Vision*, volume B of *Lecture Notes in Computer Science*, pages 97–108, Stockholm, Sweden, May 1994. Springer-Verlag.

[42] Luc Robert and Olivier Faugeras. Curve-based stereo: Figural continuity and curvature. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pages 57–62, Lahaina, Hawai, June 1991. IEEE.

[43] C. Rothwell, G. Csurka, and O. Faugeras. A comparison of projective reconstruction methods for pairs of views. In *Proceedings of the 5th International Conference on Computer Vision*, pages 932–937, Boston, MA, June 1995. IEEE Computer Society Press.

[44] P.J. Rousseeuw and A.M. Leroy. *Robust Regression and Outlier Detection*. John Wiley & Sons, New York, 1987.

[45] S.M. Seitz and C.R. Dyer. Physically-valid view synthesis by image interpolation. In *Proc. IEEE Workshop on Representation of Visual Scenes*, pages 18–25, Cambridge, Massachusetts, USA, June 1995.

[46] Amnon Shashua. Trilinearity in visual recognition by alignment. In J-O. Eklundh, editor, *Proceedings of the 3rd European Conference on Computer Vision*, volume 800-801 of *Lecture Notes in Computer Science*, pages 479–484, Stockholm, Sweden, May 1994. Springer-Verlag.

[47] C. C. Slama, editor. *Manual of Photogrammetry*. American Society of Photogrammetry, fourth edition, 1980.

[48] R. Szeliski. Image mosaicing for tele-reality applications. Technical Report CRL94/2, DEC-CRL, May 1994.

[49] Bill Triggs. Matching constraints and the joint image. In *Proceedings of the 5th International Conference on Computer Vision*, pages 338–343, Boston, MA, June 1995. IEEE Computer Society Press.

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

[51] Cyril Zeller and Olivier Faugeras. Camera self-calibration from video sequences: the Kruppa equations revisited. Research Report 2793, INRIA, February 1996.

[52] Zhengyou Zhang, Rachid Deriche, Olivier Faugeras, and Quang-Tuan Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence Journal*, 78(1-2):87–119, 1994. Appeared in October 1995, also INRIA Research Report No.2273, May 1994.