15. 3D-Reconstruction from Vanishing Points

Christian B.U. Perwass[1] and Joan Lasenby[2]

[1] Cavendish Laboratory, Cambridge
[2] C. U. Engineering Department, Cambridge

15.1 Introduction

3D-reconstruction is currently an active field in Computer Vision, not least because of its many applications. It is applicable wherever the "real world" has to be understood by a computer. This may be with regard to control movement (robots), to survey a scene for later interpretation (medicine), or to create and mix artificial with real environments (special effects).

Research on 3D-reconstruction can roughly be separated into three areas:

1. *Reconstruction with calibrated cameras*, [148, 146, 119, 79, 157]. In this case, a set of images is taken of a scene with one or more calibrated cameras. However, the camera positions are unknown. To perform a 3D-reconstruction we therefore first have to reconstruct the camera positions. To do this it is assumed that image point matches are known.
2. *Reconstruction from sequences of images*, [235, 237, 236, 246, 71, 62, 164]. Here a series of monocular, binocular or trinocular images is taken. To perform a reconstruction it is then assumed that point matches between the views in space and over time are known, and that the relative camera geometry and their internal parameters do not change. A popular method in this area is the use of the Kruppa equations [138, 78].
3. *Reconstruction from static views*, [38, 44]. A set of images of a scene taken with unknown cameras, from unknown positions is given. We still

assume that we have point matches over the images. However, note that we cannot assume anymore that the internal parameters of the cameras that took the images are the same.

The least information about a scene is given in point 3. In fact, there is so little information that a correct 3D-reconstruction is *impossible*, as we have shown in chapter 14. Therefore, some additional information is needed. Such information could be the knowledge of lengths, angles or parallel lines.

Our approach to 3D-reconstruction falls into the area of *Reconstruction from static views*. We have two images taken with unknown cameras from unknown positions and assume that apart from the point matches we also know the projections of a number of sets of parallel world lines. The latter are used to find vanishing points but also to constrain the reconstruction. This information allows us to perform an affine reconstruction of the scene. That is, we find the rotation, translation and the internal parameters of the second camera *relative* to the first. If we assume furthermore, that we have three mutually orthogonal sets of parallel lines, we can also find the internal calibration of the first camera and thus obtain a Euclidean 3D-reconstruction.

In the following discussion of our reconstruction algorithm we use the same notation as in chapter 14. We will also assume that the reader is familiar with our description of reciprocal frames, pinhole cameras, camera matrices and the basic form of the fundamental matrix. Of course, all this assumes some familiarity with Geometric Algebra (GA).

15.2 Image Plane Bases

We will be working in projective space ($\mathbb{P}^3$) with basis $\{e_1, e_2, e_3, e_4\}$ which has signature $\{---+\}$. We can project down to the corresponding Euclidean space ($\mathbb{E}^3$) via the projective split. Our general setup is that we have two pinhole cameras described by frames $\{A_\mu\}$ and $\{B_\mu\}$, respectively. The frame $\{A_\mu\}$ is also regarded as the world frame which we use for our reconstruction.

The basic form of our calculation is as follows. We start with the image points obtained from real cameras, i.e. in $\mathbb{E}^3$. These image points are embedded in $\mathbb{P}^3$. All our calculations are then performed in $\mathbb{P}^3$ and the resultant reconstruction is projected back into $\mathbb{E}^3$. This method forces us to take note of two important concepts.

1. **Correct Basis.** The power of GA in this field derives from the fact that we are not working purely with coordinates, but with the *underlying geometric basis*. Therefore, we have to make sure that the basis we are working with is actually appropriate for our problem.
2. **Scale Invariance.** The projection of homogeneous vectors into $\mathbb{E}^3$ is independent of the overall scale of the homogeneous vector. Calculations in $\mathbb{P}^3$ may depend on such an overall scale, though. We have to make sure that all our calculations are *invariant* under a scaling of the homogeneous

vectors, because such a scaling cannot and should not have any influence on our final result. Furthermore, since we are initially embedding vectors from $\mathbb{E}^3$ in $\mathbb{P}^3$ we are not given any particular scale. Any expression that is invariant under a scaling of its component homogeneous vectors will be called *scale invariant*.

As mentioned above, the frames $\{A_\mu\}$ and $\{B_\mu\}$ define two pinhole cameras. Since $\{A_\mu\}$ also serves as our world frame in $\mathbb{P}^3$ we can choose that A_4, the optical centre of camera A, sits at the origin. A_1, A_2 and A_3 define the image plane of camera A. If we want to be true to our previously stated concepts, we need to give some thought as to how we should choose the $\{A_i\}$.

Note here that we use latin indices to count from 1 to 3 and greek indices to count from 1 to 4. We also make use of the Einstein summation convention, i.e. if a superscript index is repeated as a subscript within a product, a summation over the range of the index is implied. Hence, $\alpha^i A_i \equiv \sum_{i=1}^{3} \alpha^i A_i$.

The images we obtain from real cameras are 2-dimensional. Therefore, the image point coordinates we get are of the form $\{x, y\}$, which give the displacement in a horizontal and vertical direction[1] in the image coordinate frame. However, in $\mathbb{P}^3$ an image plane is defined by three vectors. Therefore, a point on a plane in $\mathbb{P}^3$ is defined by *three* coordinates. A standard way given in the literature to extend the 2D image point coordinates obtained from a real camera to $\mathbb{P}^3$ is by writing the vector $\{x, y\}$ as $\{x, y, 1\}$. This is a well founded and very practical choice, and if we just worked with matrices and tensors we would not need to do anything else. However, since we want to tap into the power of GA, we need to understand what kind of basis is *implicitly assumed* when we write our image point coordinates in the form $\{x, y, 1\}$.

The best way to proceed is first to describe a 2D-image point in a 3D basis and then to embed this point in $\mathbb{P}^3$. An image point $\{x, y\}$ gives the horizontal and vertical displacement in the 2D-image plane coordinate frame. Let the basis corresponding to this 2D frame in $\mathbb{E}^3$ be $\{\boldsymbol{a}_1, \boldsymbol{a}_2\}$. If we define a third vecor $\boldsymbol{a}_3$ to point to the origin of the 2D frame in $\mathbb{E}^3$, then an image point with coordinates $\{x, y\}$ can be expressed as follows in $\mathbb{E}^3$.

$$\boldsymbol{x}_a = x\,\boldsymbol{a}_1 + y\,\boldsymbol{a}_2 + 1\,\boldsymbol{a}_3 = \hat{\alpha}^i\,\boldsymbol{a}_i, \tag{15.1}$$

with $\{\hat{\alpha}^i\} \equiv \{x, y, 1\}$. The $\{\hat{\alpha}^i\}$ are the image point coordinates corresponding to image point $\{x, y\}$ in $\mathbb{E}^3$. Now we embed the point $\boldsymbol{x}_a$ in $\mathbb{P}^3$.

$$X_a = (\boldsymbol{x}_a \cdot e_4) + e_4 = \hat{\alpha}^i A_i, \tag{15.2}$$

where we defined $A_1 \equiv \boldsymbol{a}_1 \cdot e_4$, $A_2 \equiv \boldsymbol{a}_2 \cdot e_4$ and $A_3 \equiv (\boldsymbol{a}_3 \cdot e_4) + e_4$. That is, A_1 and A_2 are *direction vectors*, or points at infinity, because they have no e_4 component. However, they still lie on image plane A. More precisely, they lie on the intersection line of image plane A with the plane at infinity. Note

[1] Note that although we call these directions horizontal and vertical, they may not be at a 90 degree angle to each other in general.

that A_1 and A_2 do not project to $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$, respectively, when projected back to Euclidean space. For example,

$$\frac{A_1 \wedge e_4}{A_1 \cdot e_4} = \frac{\boldsymbol{a}_1}{0} \longrightarrow \infty. \tag{15.3}$$

Nevertheless, $\{A_i\}$ is still the projective image plane basis we are looking for, as can be seen when we project X_a down to Euclidean space.

$$\boldsymbol{x}_a = \frac{X_a \wedge e_4}{X_a \cdot e_4} = \frac{\hat{\alpha}^i \, \boldsymbol{a}_i}{\hat{\alpha}^3} = x \, \boldsymbol{a}_1 + y \, \boldsymbol{a}_2 + 1 \, \boldsymbol{a}_3 \; ; \quad \hat{\alpha}^3 \equiv 1. \tag{15.4}$$

What is important here is that neither $\hat{\alpha}^1$ nor $\hat{\alpha}^2$ appear in the denominator. This shows that by writing our image point coordinates in the form $\{x, y, 1\}$ we have implicitly assumed this type of frame, which we will call a **normalised homogeneous camera frame**. The camera frames we will use in the following are all normalised homogeneous camera frames.

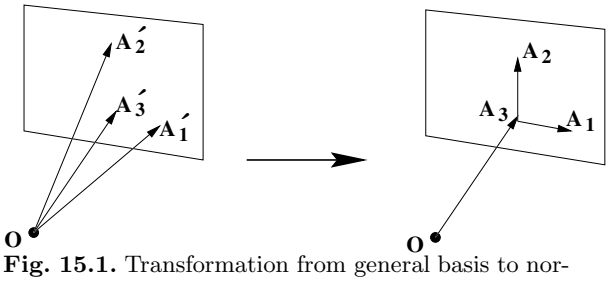


Fig. 15.1. Transformation from general basis to normalised homogeneous camera frame, in which image points have coordinates of the type $\{x, y, 1\}$

Figure 15.1 shows the difference between a general image plane basis in $\mathbb{P}^3$, denoted by $\{A'_i\}$, and a normalised homogeneous camera frame $\{A_i\}$. Note that homogeneous vectors A_1 and A_2 are drawn as lying in the image plane to indicate that they are direction vectors.

It might seem a bit odd that we have devoted so much space to the development of normalised homogeneous camera frames. However, this has far reaching implications later on and is *essential* to understand our derivation.

In $\mathbb{P}^3$ a point on the image plane of camera A can be written as $X_a = \alpha^i \, A_i$ in general. We can normalise the coordinates without changing the projection of X_a into $\mathbb{E}^3$. That is, $X_a \simeq \bar{\alpha}^i \, A_i$ with $\bar{\alpha}^i \equiv \alpha^i / \alpha^3$. The symbol $\simeq$ means equality up to a scalar factor. In this case we clearly have $\{\bar{\alpha}^i\} = \{\hat{\alpha}^i\}$.

A general point in $\mathbb{P}^3$ can be written as $X = \alpha^\mu \, A_\mu$ in the A-frame. We can normalise the coordinates of X_a in the same way as before to obtain $X \simeq \bar{\alpha}^\mu \, A_\mu$ with $\bar{\alpha}^\mu \equiv \alpha^\mu / \alpha^3$. If we project this point down to $\mathbb{E}^3$ we get[2]

$$\boldsymbol{x} = \frac{X \wedge e_4}{X \cdot e_4} = \frac{\bar{\alpha}^i}{1 + \bar{\alpha}^4}\, \boldsymbol{a}_i = \hat{\alpha}^i\, \boldsymbol{a}_i, \tag{15.5}$$

with $\hat{\alpha}^i \equiv \bar{\alpha}^i/(1 + \bar{\alpha}^4)$. That is, if $\bar{\alpha}^4 = 0$, then X is a point on the image plane of camera A. Also, if $\bar{\alpha}^4 = -1$ then X is a point at infinity. We will call $\bar{\alpha}^4$ the **projective depth** of a point in $\mathbb{P}^3$.

15.3 Plane Collineation

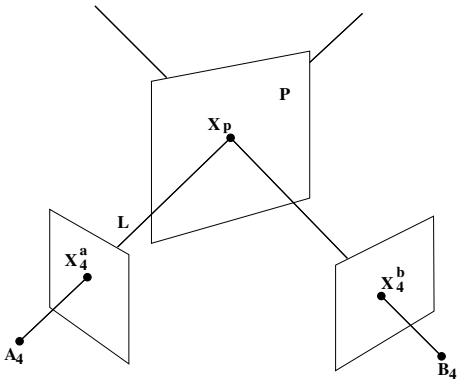


Fig. 15.2. Schematic representation of a plane collineation. Image point X_4^a is projected to X_4^b under the P-collineation

Before we can get started on the actual reconstruction algorithm, we need to derive some more mathematical objects which we will need as tools. The problem we want to solve first is the following. Let us assume we have three image point matches in cameras A and B. That is, if three points in space, $\{X_i\}$, are projected onto image planes A and B to give images $\{X_i^a\}$ and $\{X_i^b\}$ respectively, then we know that the pairs $\{X_i^a, X_i^b\}$ are images of the same point in space. If the three points in space do not lie along a line, they define a plane. This plane induces a collineation, which means that we can transfer image points from camera A to camera B through that plane. For example, let X_4^a be the image point on image plane A which we want to transfer to camera B through the plane. First we have to find the intersection point of line $A_4 \wedge X_4^a$ with the plane[3], and then we project this intersection point onto image plane B (see figure 15.2). This transformation can also be

[2] Recall that $A_4 = e_4$ (the origin of $\mathbb{P}^3$) and that the $\{A_i\}$ are a normalised homogeneous camera frame.

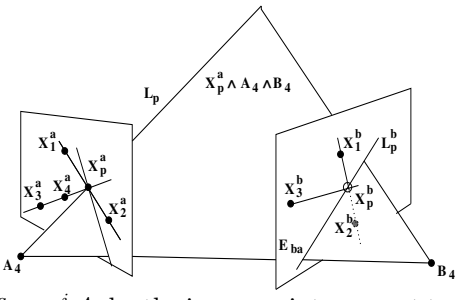
[3] Recall that A_4 is the optical centre of camera A.

represented by a 3×3 matrix, which is called a collineation matrix. Our goal is to find the collineation induced by the plane $P \equiv X_1 \wedge X_2 \wedge X_3$ by knowing the projections of the points $\{X_i\}$ onto image planes A and B, and the fundamental matrix for the two cameras. Since we know the fundamental matrix we can also calculate the epipoles. The epipoles on the two image planes are always projections of a single point in space and thus give us the projections of a fourth point on any plane in space. That is, we have in fact the projections of four points that lie on some plane P. Hence, we can find the collineation matrix directly through a matrix diagonalisation.

However, it is interesting to see what this means geometrically. Faugeras gives a geometrical interpretation[4] in [78]. We will follow his construction method to obtain a $3 \times 3 \times 3$ collineation tensor.

We start by defining three points $X_i = \alpha_i^\mu A_\mu$. The projections of these three points onto image planes A and B are $X_i^a = \bar{\alpha}_i^j A_j$ and $X_i^b = \bar{\beta}_i^j B_j$, respectively. We know the coordinates $\{\bar{\alpha}_i^j\}$ and $\{\bar{\beta}_i^j\}$, and we know that the pairs $\{\bar{\alpha}_i^j, \bar{\beta}_i^k\}$ are images of the same point in space. Furthermore, we have the fundamental matrix for the two cameras. We find the collineation induced by the plane $P = X_1 \wedge X_2 \wedge X_3$ geometrically through a two step construction.

Step 1:

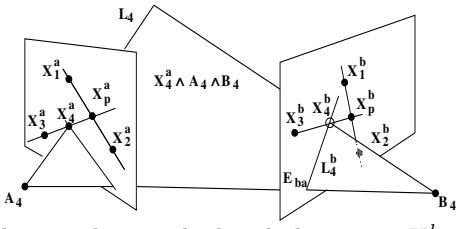


Let $X_4^a = \alpha_4^i A_i$ be the image point we want to project onto image plane B under the P-collineation. Now consider the intersection point X_p^a of lines $X_3^a \wedge X_4^p$ and $X_1^a \wedge X_2^a$. The intersection point of line $L_p \equiv A_4 \wedge X_p^a$ with an arbitrary plane in $\mathbb{P}^3$ obviously lies on L_p. Denote the projection of L_p onto image plane B by L_p^b. Obviously X_p^a can only be projected to some point on L_p^b, independent of the collineation. We also know that X_p^a has to project to some point on the line $X_1^b \wedge X_2^b$ under the specific P-collineation. Hence, X_p^b is the intersection point of lines L_p^b and $X_1^b \wedge X_2^b$. We can also write this as

$$X_p^b = (X_p^a \wedge A_4 \wedge B_4) \vee (X_1^b \wedge X_2^b) \tag{15.6}$$

[4] In [78] this method is called the **Point-Plane** procedure.

Step 2:



Now that we have calculated the point X_p^b, we can project X_4^a under the P-collineation in an analogue way. We form a line $L_4 = A_4 \wedge X_4^a$ which we project onto image plane B. X_4^b, the projection of X_4^a under the P-collineation, is then the intersection point of L_4^b and line $X_3^b \wedge X_p^b$. This can also be expressed as

$$X_4^b = (X_4^a \wedge A_4 \wedge B_4) \vee (X_3^b \wedge X_p^b) \tag{15.7}$$

By substituting equation (15.6) into equation (15.7) we can find a collineation tensor M_{ij}^k. Details of this calculation can be found in [191]. The resultant expression for M_{ij}^k is

$$\begin{aligned} M_{ij}^k \equiv \Big[\quad & \Big(F(1,2)\, \bar{\lambda}_{a\,i}^1\, \bar{\beta}_1^k - F(2,1)\, \bar{\lambda}_{a\,i}^2\, \bar{\beta}_2^k \Big)\, f_{j3}^b \\ - & \Big(F(1,2)\, \bar{\lambda}_{a\,i}^1\, f_{j1}^b - F(2,1)\, \bar{\lambda}_{a\,i}^2\, f_{j2}^b \Big)\, \bar{\beta}_3^k \Big], \end{aligned} \tag{15.8}$$

with

$$F(r,s) \equiv \bar{\alpha}_r^i \bar{\beta}_s^j\, F_{ij}\ ; \quad f_{ir}^b \equiv \bar{\beta}_r^j\, F_{ij}\ ; \quad \bar{\lambda}_{a\,k_1}^{j_1} \equiv (\bar{\alpha}_{j_2}^{k_2} \bar{\alpha}_{j_3}^{k_3} - \bar{\alpha}_{j_2}^{k_3} \bar{\alpha}_{j_3}^{k_2}), \tag{15.9}$$

where F_{ij} is the fundamental matrix for the two cameras. Here, and throughout the rest of this chapter, indices of the type $\{i_1, i_2, i_3\}$ are taken to be an even permutation of $\{1,2,3\}$. Also indices of the type $\{\mu_1, \mu_2, \mu_3, \mu_4\}$ are an even permutation of $\{1,2,3,4\}$.

To project a point $X_4^a = \bar{\alpha}_4^i\, A_i$ on image plane A, onto image plane B under the collineation described by points $\{X_1, X_2, X_3\}$, we can now simply write

$$\beta_4^k \simeq \bar{\alpha}_4^i \bar{\alpha}_4^j\, M_{ij}^k, \tag{15.10}$$

where the $\{\bar{\beta}_4^j\}$ are the coordinates of the projected point $X_4^b = \bar{\beta}^j\, B_j$ on image plane B. It can be shown that M_{ij}^k is scale invariant [191].

Equation (15.10) seems to indicate that a collineation is a quadratic relation. However, we know that $\beta_4^k = \alpha_4^i H_i^k$ where H_i^k is the collineation matrix. If we take a closer look at the components of equation (15.8) we find that $\bar{\lambda}_{a\,3}^r$ is linearly dependent on $\bar{\lambda}_{a\,1}^r$ and $\bar{\lambda}_{a\,2}^r$. Therefore, the three matrices in indices i, j of M_{ij}^k are of rank 2. We can write equation (15.10) as

$$\begin{aligned}\beta_4^k \simeq\ & \bar{\alpha}^1\bar{\alpha}^1\ M_{11}^k + \bar{\alpha}^2\bar{\alpha}^2\ M_{22}^k + \bar{\alpha}^1\bar{\alpha}^2\ (M_{12}^k + M_{21}^k) \\ & + \bar{\alpha}^1\ (M_{13}^k + M_{31}^k) + \bar{\alpha}^2\ (M_{23}^k + M_{32}^k) + \bar{\alpha}^3\ M_{33}^k\end{aligned} \tag{15.11}$$

since $\bar{\alpha}^3 = 1$ by definition. Thus, if we perform a set of similarity transforms on M_{ij}^k such that the components $M_{11}^k, M_{22}^k, M_{12}^k, M_{21}^k$ are zero, we can read off the components of the collineation matrix from the transformed M_{ij}^k. Such a similarity transformation on M_{ij}^k is possible because the matrices in indices i, j of M_{ij}^k are of rank 2.

15.4 The Plane at Infinity and Its Collineation

It will be very useful for us to see what the collineation of the plane at infinity looks like. Recall that $A_4 = e_4$ and that the $\{A_i\}$ form a normalised homogeneous camera frame. That is, A_1 and A_2 are direction vectors. Therefore, the plane at infinity P_∞ may be given by

$$P_\infty = A_1 \wedge A_2 \wedge (A_3 - A_4) \tag{15.12}$$

Now that we have the plane at infinity we can also find an expression for the collineation matrix associated with it. More details of the following calculation can be found in [191].

We want to project a point $X^a = \alpha^i A_i$ on image plane A to image plane B under the P_∞-collineation. First we have to find the intersection point X_p of line $L = A_4 \wedge X^a$ with P_∞.

$$X_p = (A_4 \wedge X^a) \vee P_\infty \simeq \alpha^i A_i - \alpha^3 A_4 \tag{15.13}$$

Now we need to find the projection X_p^b of X_p onto image plane B.

$$X_p^b = X_p \cdot B^j\ B_j = \left(\alpha^i K_{j_i}^b - \alpha^3 \varepsilon_{ba}^j\right) B_j \tag{15.14}$$

where $K_{j_i}^b \equiv A_i \cdot B^j$ is the 3×3 camera matrix minor of camera B, and $\varepsilon_{ba}^j \equiv A_4 \cdot B^j$ is the epipole of camera B and also the fourth column of the full camera matrix[5]. Note that we use here a notation of relative super- and subscripts to keep the absolute superscript position free for other uses. From equation (15.14) it follows that we can write the collineation matrix of P_∞ as

$$\Psi_{j_i}^\infty \equiv [K_{j_1}^b, K_{j_2}^b, K_{j_3}^b - \varepsilon_{ba}^j] \tag{15.15}$$

[5] The full camera matrix is given by $K_{j_\mu}^b = A_\mu \cdot B^j$. See chapter 14for details on camera matrices and epipoles.

where i counts the columns. Therefore, if we want to project a point $X_a = \alpha^i A_i$ on image plane A, onto image plane B under the P_∞-collineation we can write

$$\beta^j_\infty \simeq \alpha^i \Psi^\infty_{j_i}. \tag{15.16}$$

What does the P_∞-collineation describe geometrically? If X_a is an image point in camera A and X^∞_b is its projection under the P_∞-collineation, then from the construction of the collineation it follows that the lines $L_a = A_4 \wedge X_a$ and $L_b = B_4 \wedge X^\infty_b$ meet in a point on P_∞. If two lines meet in a point on the plane at infinity, they are parallel. Therefore, the P_∞-collineation tells us which two image points X_a and X^∞_b on image planes A and B, repectively, correspond such that the lines $A_4 \wedge X_a$ and $B_4 \wedge X^\infty_b$ are parallel. Obviously, this tells us something about the relative orientation of the two cameras.

We can use our knowledge of the relation between Ψ^∞ and the camera matrix to find the depths of a set of world points whose projections are known in both cameras, if we also know the projections of at least three pairs of parallel lines. We will assume for the moment that for each point pair $\{\bar\alpha^i, \bar\beta^j\}$ we also know $\bar\beta^j_\infty$, which is the projection of $\bar\alpha^i$ under the P_∞-collineation. From the definition of the camera matrix we know that

$$\beta^j = \alpha^i K^b_{j_i} + \alpha^4 \varepsilon^j_{ba}. \tag{15.17}$$

Furthermore, equation (15.16) may be rewritten as

$$\beta^j_\infty \simeq \alpha^i K^b_{j_i} - \alpha^3 \varepsilon^j_{ba} \tag{15.18}$$

We can now combine equations (15.17) and (15.18) to obtain the following expression (see [191] for details).

$$\bar\alpha^4 = \bar\alpha^i \bar K^b_{3_i}\, \zeta^j_1 - \zeta^j_2\ ; \quad j \in \{1,2\}. \tag{15.19}$$

with

$$\zeta^j_1 \equiv \frac{\bar\beta^j_\infty - \bar\beta^j}{\bar\beta^j - \bar\varepsilon^j_{ba}}\ ; \quad \zeta^j_2 \equiv \frac{\bar\beta^j_\infty - \bar\varepsilon^j_{ba}}{\bar\beta^j - \bar\varepsilon^j_{ba}} \tag{15.20}$$

Since equation (15.19) has to give the same result for both $j = 1$ and $j = 2$ independent of $\bar K^b_{3_i}$, it follows that $\zeta^1_1 = \zeta^2_1$ and $\zeta^1_2 = \zeta^2_2$. Therefore, we will discard the superscript of the ζs in the following.

Equation 15.19 by itself is still not useful, since we neither know $\bar\alpha^4$ nor $\bar K^b_{3_i}$. However, if we had some constraints on the projective depths ($\bar\alpha^4$) for a number of points we could find $\bar K^b_{3_i}$. Once $\bar K^b_{3_i}$ is known for a particular camera setup, we can use it to calculate the depths for any point matches. Before we show how $\bar K^b_{3_i}$ can be evaluated, we will take a closer look at how to find the $\{\bar\beta^j_\infty\}$.

15.5 Vanishing Points and P_∞

We mentioned earlier that the $\{\beta^j_\infty\}$ are the projections of the $\{\alpha^i\}$ onto image plane B under the P_∞-collineation. We can find the P_∞-collineation Ψ^∞ from the projection pairs of three points on P_∞ and the fundamental matrix.

If two parallel world lines are projected onto an image plane, their projections are only parallel if the image plane is parallel to the world lines. The intersection point of the projections of two parallel world lines is called a *vanishing point*.

Two parallel world lines meet at infinity. In projective space $\mathbb{P}^3$ this may be expressed by saying that the intersection point of two parallel world lines lies on P_∞. Points on P_∞ may also be interpreted as directions. Therefore, intersecting a line with P_∞ gives its direction. In this light, a vanishing point is the projection of the intersection point of two parallel lines. Or, in other words, it is the projection of a direction.

If we knew three vanishing points which are projections of three mutually orthogonal directions, we would know how a basis for the underlying Euclidean space $\mathbb{E}^3$ projects onto the camera used. This information can be used to find the internal camera calibration [44]. Here our initial goal is to find the *relative* camera calibration of the two cameras. We can then find an affine reconstruction. To achieve this, we do not require the vanishing points to relate to orthogonal directions. However, the more mutually orthogonal the directions related to the vanishing points are, the better the reconstruction will work.

15.5.1 Calculating Vanishing Points

Before we go any further with the actual reconstruction algorithm, let us take a look at how to calculate the vanishing points. Suppose we have two image point pairs $\{\bar{\alpha}^i_{u1}, \bar{\alpha}^i_{u2}\}$ and $\{\bar{\alpha}^i_{v1}, \bar{\alpha}^i_{v2}\}$, defining two lines on image plane A, which are projections of two parallel world lines. The vanishing point is the intersection of lines L_u and L_v where

$$L_u = \lambda^u_i L^i_a \quad ; \quad L_v = \lambda^v_i L^i_a, \tag{15.21}$$

and

$$\lambda^u_{i_1} \equiv \bar{\alpha}^{i_2}_{u1}\bar{\alpha}^{i_3}_{u2} - \bar{\alpha}^{i_3}_{u1}\bar{\alpha}^{i_2}_{u2} \quad ; \quad \lambda^v_{i_1} \equiv \bar{\alpha}^{i_2}_{v1}\bar{\alpha}^{i_3}_{v2} - \bar{\alpha}^{i_3}_{v1}\bar{\alpha}^{i_2}_{v2}, \tag{15.22}$$

are the homogeneous line coordinates. Also note that $L^{i_1}_a \equiv A_{i_2} \wedge A_{i_3}$ (see chapter 14). The intersection point X^a_{uv} of lines L_u and L_v is then given by

$$X^a_{uv} = L_u \vee L_v = \alpha^i_{uv} A_i, \tag{15.23}$$

where

$$\alpha_{uv}^{i_1} \equiv (\lambda_{i_2}^{v}\, \lambda_{i_3}^{u} - \lambda_{i_3}^{v}\, \lambda_{i_2}^{u}). \tag{15.24}$$

First of all note that the $\{\alpha_{uv}^{i}\}$ define a point in $\mathbb{P}^2$. Since we defined A_1 and A_2 to be directions, the image point coordinates $\{x, y\}$ in $\mathbb{E}^2$ corresponding to the $\{\alpha_{uv}^{i}\}$, are found to be $\{\bar{\alpha}_{uv}^{1}, \bar{\alpha}_{uv}^{2}\}$ through the projective split, where $\bar{\alpha}_{uv}^{i} \equiv \alpha_{uv}^{i}/\alpha_{uv}^{3}$. Note that points which lie at infinity in $\mathbb{E}^2$ can be expressed in $\mathbb{P}^2$ by points which have a zero third component. Such points will also be called directions.

The fact that points at infinity in $\mathbb{E}^2$ are nothing special in $\mathbb{P}^2$ shows an immediate advantage of using homogeneous coordinates for the intersection points over using 2D-coordinates. Since we are looking for the intersection point of the projections of two parallel world lines, it may so happen, that the projections are also parallel, or nearly parallel. In that case, the 2D image point coordinates of the vanishing point would be very large or tend to infinity. This, however, makes them badly suited for numerical calculations. When using homogeneous coordinates, on the other hand, we do not run into any such problems.

15.5.2 Vanishing Points from Multiple Parallel Lines

Above we described how to find a vanishing point from the projections of two parallel world lines. In practical applications the lines will only be known with a finite precision and will also be subject to a measurment error. Therefore, we could improve on the quality of a vanishing point if sets of more than two parallel lines are known. In particular, the vanishing point quality is improved if these parallel lines are taken from varying depths within in world scene. In [191] we discuss a standard method, which consists of finding the null space of a matrix of the homogeneous line coordinates. This method gives us the best fitting vanishing point in homogeneous coordinates, in the least squares sense.

Note that in [38] vanishing points are found as 2D-image point coordinates, which means that only parallel world lines can be used that are not parallel in the image. In [44] the projections of at least three parallel world lines have to be known to calculate a vanishing point. The implementation of our algorithm switches automatically between finding a vanishing point from two parallel lines, and calculating it from multiple parallel lines, depending on how much information is available.

15.5.3 Ψ^{∞} from Vanishing Points

Now we return to our reconstruction algorithm. We discussed vanishing points since they are projections of points on P_∞. If we know three vanishing point matches over cameras A and B and the epipoles, we can calculate the P_∞-collineation matrix Ψ^{∞}. Once we have Ψ^{∞} we can find the projections of

some image points $\{\bar{\alpha}_n^i\}$ on image plane A, onto image plane B under the P_∞-collineation. That is,

$$\bar{\beta}^k_{n\,\infty} \simeq \bar{\alpha}^i_n \, \Psi^\infty_i \tag{15.25}$$

We can now use the $\{\bar{\beta}^j_{n\,\infty}\}$ to find the $\{\zeta^j_n\}$ for equation (15.19).

15.6 3D-Reconstruction of Image Points

Now that we have found Ψ^∞ and thus can calculate the $\{\zeta_n\}$ from equation (15.20), we can think about how to find the correct depth values for the image point matches $\{\bar{\alpha}^i_n, \bar{\beta}^j_n\}$.

We will perform an affine reconstruction. That is, we reconstruct in the frame of camera A. When we plot our final reconstructed points we will assume that the A-frame forms an orthonormal frame of $\mathbb{E}^3$, though. However, we do not need to assume anything about the frame of camera B, since we will find the translation, rotation and internal calibration of camera B *relative* to camera A. To find the internal calibration of camera A relative to an orthonormal frame of $\mathbb{E}^3$, we would need to know the projection of this orthonormal set of directions onto camera A [44].

We have already found sets of parallel lines to calculate vanishing points. We can reuse these sets of lines to constrain the depth values found with equation (15.19). In particular, we will regard the $\{\bar{K}^b_{3_i}\}$ as free parameters. If we now take the image point matches that define the projections of two parallel world lines, we can use this extra information to constrain the $\{\bar{K}^b_{3_i}\}$. That is, we vary the free parameters until the reconstructed points define a pair of parallel world lines again.

15.6.1 The Geometry

Before we start developing an algorithm to find the best $\{\bar{K}^b_{3_i}\}$ we will take a quick look at the relevant geometry. In figure 15.3 we have drawn the geometry underlying our reconstruction algorithm.

A_4 and B_4 are the optical centres of cameras A and B, respectively. We have also chosen A_4 to lie at the origin of $\mathbb{E}^3$. Recall that A_1, A_2 and B_1, B_2 are direction vectors in $\mathbb{P}^3$. We have drawn these vectors here as lying on the image planes to indicate this.

A world point X is projected onto image planes A and B giving projections X_a and X_b, respectively. X_b^∞ is the projection of X_a onto image plane B under the P_∞-collineation. Also, E_{ba} is the epipole of camera B.

Now we can see what the $\{\zeta_{1n}, \zeta_{2n}\}$ components from equation (15.19) express.

$$\zeta_{1n} \equiv \frac{\bar{\beta}^j_{n\,\infty} - \bar{\beta}^j_n}{\bar{\beta}^j_n - \bar{\varepsilon}^j_{ba}}$$

gives the ratio of the distance (in x or y direction) between X_b^∞ and X_b, and X_b and E_{ba}.

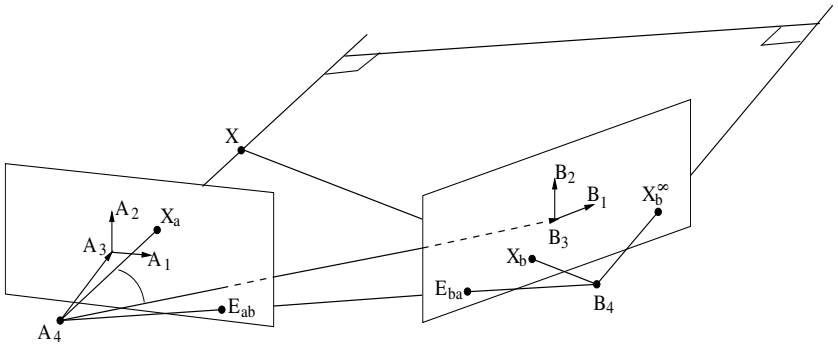


Fig. 15.3. This figure shows the geometry behind equation (15.19). A point X is projected onto cameras A and B, giving images X_a and X_b, respectively. Projecting X_a onto image plane B under the P_∞-collineation gives X_b^∞. We choose A_4 to be the origin of $\mathbb{E}^3$. $K_{3_i}^b$ gives the components of A_1, A_2 and A_3 along B_3

$$\zeta_{2n} \equiv \frac{\bar{\beta}_{n\,\infty}^{j} - \bar{\varepsilon}_{ba}^{j}}{\bar{\beta}_{n}^{j} - \bar{\varepsilon}_{ba}^{j}}$$

gives the ratio of the distance (in x or y direction) between X_b^∞ and E_{ba}, and X_b and E_{ba}.

Recall that $K_{3_i}^b = A_i \cdot B^3$, that is, it gives the components of the $\{A_i\}$ along B_3. Therefore, varying the $\{K_{3_i}^b\}$ means that we are moving B_3, which is the principal point on image plane B. Since X_b^∞ cannot change when we vary $K_{3_i}^b$ the relation between B_3 and B_4 is fixed. Thus, changing B_3 means changing B_4. In this respect, finding the correct $\{K_{3_i}^b\}$ means finding the correct translation of camera B relative to camera A. The relative rotation has already been fixed through finding P_∞.

However, it is only the relative sizes of the $\{\bar{K}_{3_i}^b\}$ that are really important. An overall scale factor will only change the depths of all reconstructed points simultaneously. Therefore, we can fix the depth of one image point, to fix the scale of $\bar{K}_{3_i}^b$.

15.6.2 The Minimization Function

We mentioned before that we will use our knowledge of parallel lines once again to constrain the $\{\bar{K}_{3_i}^b\}$ from equation (15.19). Let $L_u^a = X_{u1}^a \wedge X_{u2}^a$ and $L_v^a = X_{v1}^a \wedge X_{v2}^a$ be the projections of two parallel world lines onto image plane A. In general we define world points and image points as

$$\left.\begin{array}{ll} X_{ur} \equiv \bar{\alpha}_{ur}^{\mu}\, A_\mu\ ; & X_{ur}^a \equiv \bar{\alpha}_{ur}^{i}\, A_i \\ X_{vr} \equiv \bar{\alpha}_{vr}^{\mu}\, A_\mu\ ; & X_{vr}^a \equiv \bar{\alpha}_{vr}^{i}\, A_i \end{array}\right\} r \in \{1, \ldots, n\}. \tag{15.26}$$

Furthermore, if we know the image points on image plane B corresponding to X^a_{u1}, X^a_{u2}, X^a_{v1} and X^a_{v2}, and we have found Ψ^∞, then we can calculate the corresponding ζs from equation (15.20). Equation (15.19) will now allow us to find the projective depths for X^a_{u1}, X^a_{u2}, X^a_{v1} and X^a_{v2}. Therefore, we can calculate the world lines $L_u = X_{u1} \wedge X_{u2}$ and $L_v = X_{v1} \wedge X_{v2}$.

Now, we know that L_u and L_v are supposed to be parallel, which means that they have to intersect P_∞ in the same point. This will be the constraint which we will use to find the correct $\{\bar{K}^b_{3_i}\}$. Let X^∞_u and X^∞_v be defined as

$$X^\infty_u \equiv L_u \vee P_\infty \; ; \quad X^\infty_v \equiv L_v \vee P_\infty . \tag{15.27}$$

Lines L_u and L_v are parallel iff

$$X^\infty_u \wedge X^\infty_v = 0 \tag{15.28}$$

Instead of using this condition we could also project L_u and L_v into $\mathbb{E}^3$, and then check that they are parallel. However, projecting into $\mathbb{E}^3$ means dividing through the projective depth, which means that our free parameters are now in the denominator of a minimisation function. Apart from creating a minimisation surface with singularities, the derivatives of such a minimisation function will be more complicated and thus cost more computing time.

Finding the Minimisation Parameters. The following expression for X^∞_u is derived in more detail in [191].

$$X^\infty_u = L_u \vee P_\infty = \chi^i_u \, A^\infty_i \tag{15.29}$$

where

$$\begin{aligned} &\chi^i_u \equiv (\bar{\lambda}^u_{i3} + \bar{\lambda}^u_{i4}) \; ; \quad \bar{\lambda}^u_{\mu_1\mu_2} \equiv \bar{\alpha}^{\mu_1}_{u1} \, \bar{\alpha}^{\mu_2}_{u2} - \bar{\alpha}^{\mu_2}_{u1} \, \bar{\alpha}^{\mu_1}_{u2} \\ &A^\infty_1 \equiv A_1 \; ; \quad A^\infty_2 \equiv A_2 \; ; \quad A^\infty_3 \equiv A_3 - A_4 \end{aligned} \tag{15.30}$$

The free parameters we have are the $\{\bar{K}^b_{3_i}\}$. To make future equations somewhat clearer we will define $\varphi_i \equiv \bar{K}^b_{3_i}$. Hence, equation (15.19) will be written as

$$\bar{\alpha}^4_n = \bar{\alpha}^i_n \, \zeta_{1n} \, \varphi_i - \zeta_{2n} . \tag{15.31}$$

Recall that lines L_u and L_v are parallel iff $X^\infty_u \wedge X^\infty_v = 0$. We can now write this expression in terms of the $\{\chi^i\}$.

$$X^\infty_u \wedge X^\infty_v = \Lambda^{uv}_i \, L^i_\infty \; ; \quad \Lambda^{uv}_{i_1} \equiv \chi^{i_2}_u \, \chi^{i_3}_v - \chi^{i_3}_u \, \chi^{i_2}_v \tag{15.32}$$

with $L^{i_1}_\infty \equiv A^\infty_{i_2} \wedge A^\infty_{i_3}$. Each of the $\{\Lambda^{uv}_i\}$ has to be zero if $X^\infty_u \wedge X^\infty_v = 0$. Therefore, from an analytical point of view, the expression we should try to minimise for each parallel line pair $\{L_u, L_v\}$ is

$$\Delta^{uv} : \; \varphi_j \longrightarrow \sum_{i=1}^{3} (\Lambda^{uv}_i)^2 . \tag{15.33}$$

Improving Computational Accuracy. However, for a computer with finite floating point precision, this equation poses a problem. The culprits in this case are the $\{\chi^i\}$. Recall that they give the direction of a line in homogeneous coordinates. Before they are used in equation (15.32) they should be normalised to improve the precision of the equation on a computer.

$$\hat{\chi}_u^i \equiv \frac{\chi_u^i}{\sqrt{\sum_i (\chi_u^i)^2}} \tag{15.34}$$

Therefore, the minimisation function we will use is

$$\Delta^{uv} : \varphi_j \longrightarrow \sum_{i=1}^{3} (\hat{\Lambda}_i^{uv})^2 ; \quad \hat{\Lambda}_{i_1}^{uv} \equiv \hat{\chi}_u^{i_2} \hat{\chi}_v^{i_3} - \hat{\chi}_u^{i_3} \hat{\chi}_v^{i_2} \tag{15.35}$$

The Derivatives. The derivative of Δ^{uv} is computationally not a particularly expensive expression. Therefore, we can use a minimisation routine that also uses the derivatives of the minimisation function. This will make the minimisation process more efficient and robust. Details about the derivatives can be found in [191].

Implementing the Depth Constraint. At the moment the minimisation function Δ^{uv} depends on three parameters: the $\{\varphi_j\}$. However, we mentioned earlier that we can fix, the depth of one point. This will reduce the number of free parameters to two. How this is done best is described in [191]. It turns out that constraining the depth of one point is necessary. Otherwise the minimisation routine tries to push the whole scene to infinity.

The Minimisation Routine. We use a modified version of the *conjugate gradient* method to perform the minimisation. This modified version is called *MacOpt* and was developed by David MacKay [165]. It makes a number of improvements over the conjugate gradient method as given in [195]. MacOpt assumes that the minimisation surface is fundamentally convex with no local minima. However, our surface is only of that shape near the absolute minimum[6]. It turns out that the success rate of finding the absolute minimum can be improved if we first use the unnormalised χs to step towards the minimum, and then use the normalised χs to find the minimum with high accuracy. This is because the minimisation surface for the unnormalised χs is of a convex shape, whereas the minimisation surface for the normalised χs has a number of local minima.

[6] A number of examples of minimisation surfaces and their corresponding reconstructions are demonstrated by the program **MVT**, which can be downloaded from C.Perwass' home page. This program runs under Windows 95/98 and NT4/5.

Image Point Normalisation. Before we can calculate the collineation tensor for the P_∞-collineation we have to find the fundamental matrix (F) for the two views (see equation (15.8)). For the calculation of the fundamental matrix we cannot use the pixel coordinates directly, because they are typically too large to obtain good accuracy in our numerical calculations. This is also true for all other calculations performed here. Therefore, we need to scale the image point coordinates so that they are of order 1.

In [106] Hartley suggests that the scales and skews applied to the image point coordinates are found in the following way. The skew is given by the coordinates of the centroid of all image points. Then the average distance of the skewed image points from the origin is calculated. The inverse of that distance gives the scale.

This is a good method if we just wanted to calculate F. However, it turns out that for our purposes such a scaling is not suitable. In fact, we found that it is important to conserve the aspect ratio of the images (separately), and to ensure that the origin of the image plane is chosen in the same way in both images.

We choose the image plane origin to be in the centre of each image plane and then scale the image points by dividing their x and y coordinate by the image resolution in the x-direction. This preserves the aspect ratio.

15.7 Experimental Results

We can now outline the structure of our reconstruction algorithm.

Step 1: We find point matches and sets of projections of parallel lines over the two images.

Step 2: We calculate three vanishing points and the fundamental matrix. This allows us to find the P_∞-collineation matrix Ψ^∞.

Step 3: We select a set of parallel lines that we want to use to constrain our minimisation. Note that one pair of parallel lines may be enough. More pairs do not necessarily improve the result, since they may not be consistent due to errors.

Step 4: The image points on image plane A which define the chosen parallel lines are projected onto image plane B under the P_∞-collineation with Ψ^∞.

Step 5: We can now find the $\{K^b_{3_i}\}$ by minimising equation (15.33) or equation (15.35).

Step 6: Once we have found $K^b_{3_i}$ we can use it in conjunction with Ψ^∞ in equation (15.31) to reconstruct any other image point matches for this camera setup.

15.7.1 Synthetic Data

To test the quality of the reconstructions we created synthetic data. The advantage of using synthetic data is that we can get a *geometric* quality measure of the reconstruction. Also if an algorithm fails with synthetic data it is clearly unlikely to work with real data.

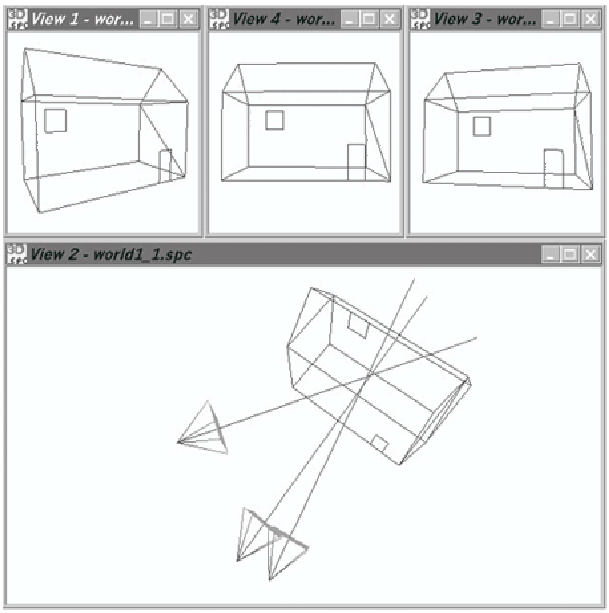


Fig. 15.4. The synthetic data was created from projecitons of the house onto the cameras

The lower picture in figure 15.4 shows a house with three cameras. The three smaller pictures on top show the projections of the house onto the three image planes. The house consists of 18 vertices, which were all used in our calculations. We performed two trials: trial 1 uses an orthogonal set of vanishing points. Trial 2 uses two orthogonal vanishing points but the third vanishing point is found from the two lines on the roof which are vertically sloping and closest to the camera. In each trial we also tested two camera configurations: the camera to the very left and the very right, and the two cameras which are close together. The former will be called the *far cameras* and the latter the *close cameras* configuration.

Recall that we can and, in fact, have to fix the depth of one point. Since we know the true points we can set this depth to its true value. Also remember that we perform our reconstruction in the frame of one of the cameras. But

we also know this frame and can therefore transform our reconstructed points to lie in the appropriate frame. The reconstruction obtained in this way can then be compared *directly* with the true object.

In our experiments we added a Gaussian error with a mean deviation between 0 and 12 pixels to the image points. The camera resolutions were 600×600 pixels. For each setting of the mean deviation of the induced error we calculated the $\{K_{3_i}^b\}$ 100 times, each time with different errors, to obtain a statistically meaningful result. Each calculation of the $\{K_{3_i}^b\}$ can be used to reconstruct any image point matches in the two images. Therefore, we projected the house again onto the two image planes, again introducing an error of the same mean deviation. These image points are then reconstructed and compared with the true points. This was done 20 times for each calculation of the $\{K_{3_i}^b\}$. This way we obtained a separation of the calibration and the reconstruction.

The quality measure of a reconstruction is given by the root mean squared error between the locations of the reconstructed points and the true points. That is, we take the root of the mean of the sum of the distances squared between the true and the reconstructed points. We evaluated the RMS error over the 20 reconstructions for each calibration (i.e. calculation of the $\{K_{3_i}^b\}$), and also over all calculations of the $\{K_{3_i}^b\}$ for each mean deviation of the induced error. The former will be called the "RMS/Trial" and the latter the "Total RMS".

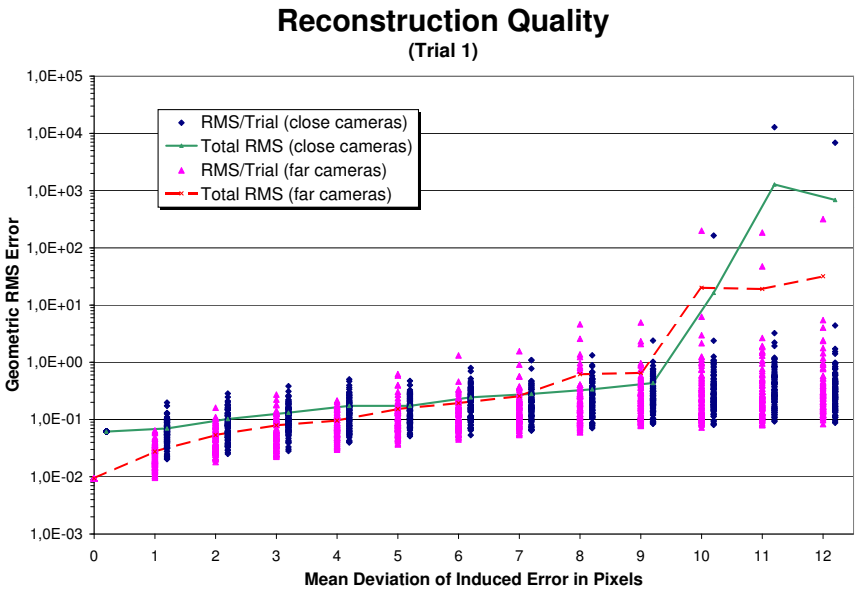


Fig. 15.5. Comparison of reconstruction quality for first trial

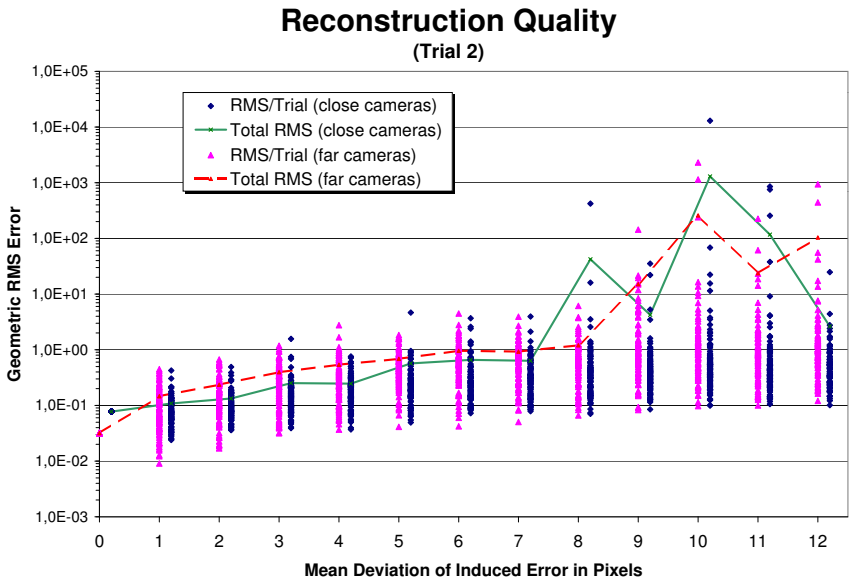


Fig. 15.6. Comparison of reconstruction quality for second trial

Figure 15.5 shows the results when using an orthogonal set of vanishing points and figure 15.6 when using a non-orthogonal set, as described above. Note that the y-axis has a $\log_{10}$ scale. The length of the house is 2 units, its total height 1.5 units and its depth 1 unit. The results for the close camera configuration are slighty displaced to the right, so that they can be distinguished from the far cameras setup.

The first thing we can see from the graphs is that as the induced error increases over 6 pixels we start to get error configurations where the algorithm breaks down. This can be either due to the minimisation getting stuck in local minima or because the absolute minimum is at a wrong position. The latter is possible since the minimisation surface depends on Ψ^∞ and F.

Furthermore, it can be seen that the far cameras configuration is more immune to induced errors than the close cameras configuration. Also the non-orthogonal set of vanishing points fares worse than the orthogonal one. Curiously, in trial 2 the far cameras configuration is worse than the close cameras configuration.

In general it can be seen, though, that an error with a mean deviation of up to 5 pixels still gives acceptable reconstructions. It might seem odd, though, that if some error is introduced into the image points, the reconstruction can actually be better than with no noise at all. This is because even if no additional error is applied, there is still an error due to the digitisation

in the cameras. Particular configurations of induced error can compensate for that by chance. However, the figures also show that the probability of the added error improving the reconstruction is about as high as making the reconstruction worse (relative to the total RMS). Nevertheless, this fact supplies us with an interesting idea: we might be able to improve our reconstructions from real data by *adding noise* to the image points. To be more precise, we could vary the image point coordinates slightly until we obtain an improved reconstruction. Since our calibration algorithm is quite fast it seems feasible to employ maximum entropy methods. We will discuss this in future work.

Note that we have calculated F with a simple method which does not enforce the rank 2 constraint on F. Nevertheless, the reconstruction quality is quite good, which seems to indicate that a highly accurate F is not very important for our algorithm. Therefore, it appears that in certain cases fully constraint evaluations of F are not necessary to obtain good results. Of course, using a fully contraint F might improve the results. Research on calculating F or the trifocal tensor (which is a related problem) optimally can be found in [106, 108, 147, 189, 80, 84, 120].

15.7.2 Real Data

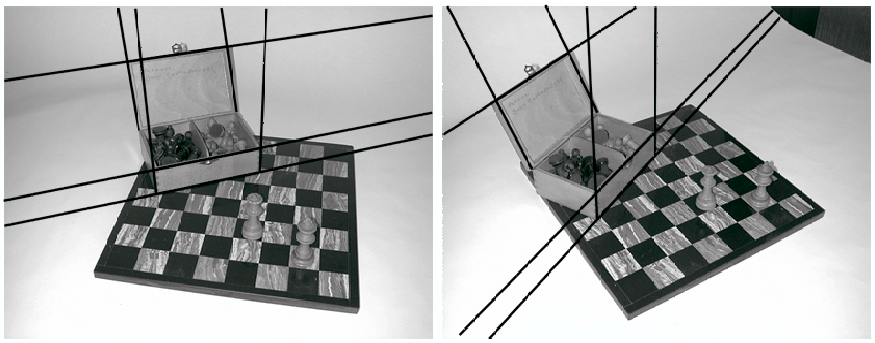


Fig. 15.7. Initial images with parallel lines used for the calculation of the vanishing points and minimisation function indicated

The real test for any reconstruction algorithm is the reconstruction of a real world scene, though. Figure 15.7 shows two views of a chessboard which we used for reconstruction[7]. The original images had a resolution of

[7] These pictures were actually taken by C.Perwass' father, in a different country, with equipment unknown to the authors. They were then sent via email to the authors. That is, the only thing known about the pictures to the authors, are the pictures themselves.

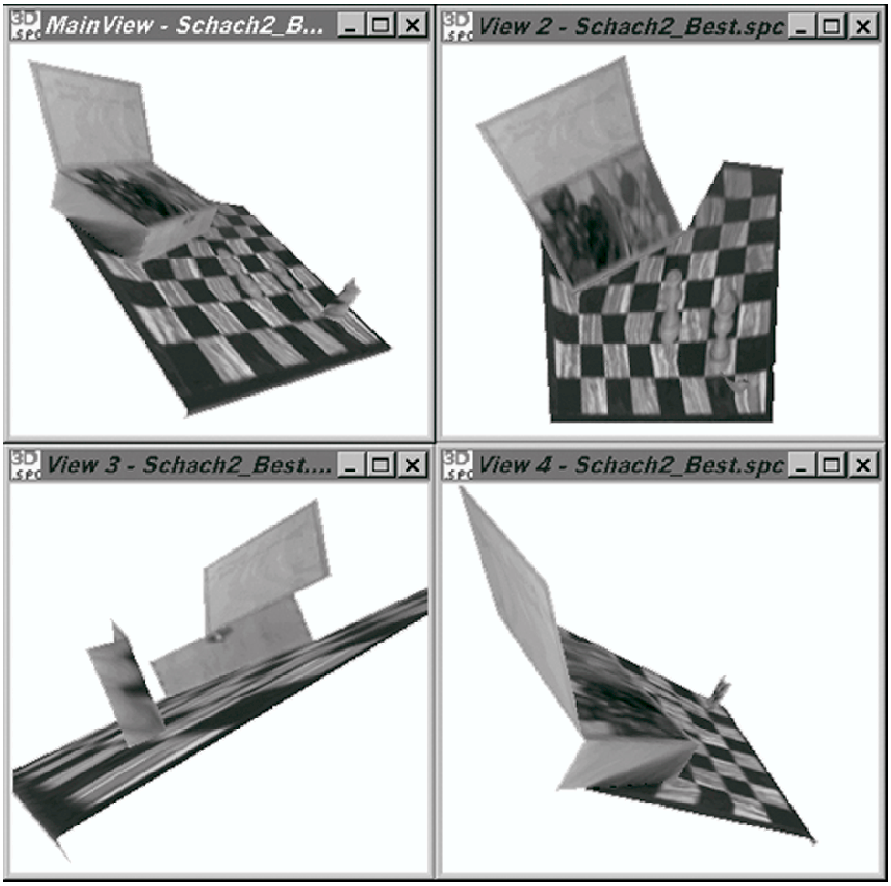


Fig. 15.8. Reconstruction of the chessboard (Schachbrett)

1280×960 pixels. The lines indicate the parallel lines used to calculate the vanishing points. The two sets of parallel lines on the front of the chessbox were used in the minimisation routine. The fundamental matrix used was calculated from 13 point matches. The resultant reconstruction[8] can be seen in figure 15.8.

The different views of the reconstruction show that the chessbox was reconstructed quite well. However, the chessboard is not really square. Remember, though, that this is only an affine reconstruction drawn in an orthonormal frame. That is, we assume that the camera frame is orthonormal. Furthermore, we have only used two line pairs and one line triplet to find

[8] This and other reconstructions, as well as some more analysis of the reconstruction algorithm are demonstrated by the program **MVT**, which can be downloaded from C.Perwass' home page.

three vanishing points, of which only two relate to orthogonal directions in $\mathbb{E}^3$. The reconstruction might be improved by exploiting all the parallel lines available, of which there are many on a chessboard.

Also note that the front side of the chessboard is reconstructed very nicely, at a proper right angle to its top side. The chess figure, which can be seen best in the bottom left hand view of figure 15.8, is not reconstructed particularly well, though. This is because it is very difficult to find matching point sets for round objects.

15.8 Conclusions

We have presented here an algorithm for the affine reconstruction of 3D scenes from two static images. The information we need is firstly point matches over the two images, and secondly at least three sets of parallel lines. From this information alone we implicitly[9] find the internal calibration, rotation and translation of the second camera relative to the first one. This allows us to perform an affine reconstruction of the scene. Assuming that the three sets of parallel lines are mutually orthogonal we could also find the internal calibration of the first camera.

Our algorithm is clearly not automatic. This is because apart from the point matches, combinations of vanishing points and parallel lines can be chosen freely. Also the information that certain lines in an image are actually parallel in the world, is a knowledge-based decision that humans are easily capable of, but not computers.

Advantages of our algorithm are that it is fast and that the reconstruction is robust for a particular calibration. On a PentiumII/233MHz under Windows 98 it took on average 160ms for a calibration (10000 trials). This time includes updating of dialog boxes. In an optimised program this time could probably be reduced to less than half. Robustness of the calibration depends mostly on the set of vanishing points used. The more similar the directions the vanishing points describe are, the less robust the calibration is.

We believe that apart from presenting an interesting affine reconstruction algorithm we have also shown that GA is a useful tool which allows us to gain geometric insight into a problem.

[9] Future work will look at how these entities can be found explicitly.