A Geometric Algebra Approach to 3D-Reconstruction from Vanishing Points

C.B.U. Perwass[1], J. Lasenby[2]

CUED/F - INFENG/TR. 364
February 2000

[1] Institut für Informatik, Preusserstr. 1–9, 24105 Kiel, Germany
+49 (431) 560480
christian@perwass.de
www.perwass.de

[2] C. U. Engineering Department, Trumpington Street, Cambridge CB2 1PZ, UK
+44 (1223) 332639
jl@eng.cam.ac.uk
www-sigproc.eng.cam.ac.uk/~jl

Abstract

Here we present a 3D-reconstruction algorithm which reconstructs a scene from two static images if a set of image point matches and some parallel world lines are known. The algortihm exploits the fact that we can express the collineation of the plane at infinity in terms of a camera matrix. The algorithm we find is fast and robust and is investigated with synthetic and real data. Using synthetic data shows that the quality of the fundamental matrix and the epipoles is not of high importance. Unlike other reconstruction algorithms that use vanishing points, our algortihm also works with three pairs of parallel lines that do not point in mutually orthogonal directions.

Contents

1 Introduction

In the following we will consider a system of two pinhole cameras looking at a scene in the world. Our goal is to create a 3D-reconstruction of the world scene from the images taken by the pinhole cameras. We will show that such a 3D-reconstruction is possible if we know a number of point matches between the images and also some parallel world lines.

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 artifical with real environments (special effects).

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

1. *Reconstruction with calibrated cameras.* [1, 2, 3, 4, 5, 6, 7] 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 point matches between all the points are known.

2. *Reconstruction from sequences of images.* [8, 9, 10, 11, 12, 13, 14] 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 [15, 16].

3. *Reconstruction from static views.* [17, 18] 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*. 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. Furthermore, we take the 3D-coordinate frame of the first camera as the basis of the Euclidean space we reconstruct in, and find the rotation, translation and the internal parameters of the second camera *relative* to the first. Note that relative translation, rotation and internal parameters are not found explicitly. This is a disadvantage if these values have to be known, but an advantage if we are only interested in a 3D-reconstruction.

Other advantages of our algorithm are that it is fast[1] and robust[2]. Furthermore, our derivation hinges crucially on Geometric Algebra (GA), although probably not all readers will call this an advantage. With GA we are able to derive the relation between the collineation of the plane at infinity (Ψ^{∞}) and a camera matrix. This forms the basis of our reconstruction algorithm.

In the following discussion of our 3D-reconstruction algorithm we use the same notation as in [19]. 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 [19, 20]. Of course, all this assumes some familiarity with GA. For a complete introduction to GA see [21, 22] and for other brief summaries see [3, 23, 24]. A good introduction geared towards the use in the computer sciences can be found in [25].

2 Image Plane Bases

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 $\mathcal{E}^3$. These image points are then projected up into $\mathcal{P}^3$. All our calculations are then performed in $\mathcal{P}^3$ and the resultant reconstruction is projected back into $\mathcal{E}^3$. 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*

[1]On a PentiumII/233MHz under Windows 98 it took on average 160ms for a calibration (10000 trials). This time includes updating of dialog boxes and OpenGL windows. In an optimised program this time could probably be reduced to less than half.

[2]Robustness depends mostly on the set of vanishing points used. The more similar the directions the vanishing points describe, the less robust the algorithm is.

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 $\mathcal{E}^3$ is independent of the overall scale of the homogeneous vector. Calculations in $\mathcal{P}^3$ 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 projecting up from $\mathcal{E}^3$ to $\mathcal{P}^3$ 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 $\mathcal{P}^3$ 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[3] in the image coordinate frame. However, in $\mathcal{P}^3$ an image plane is defined by three vectors. Therefore, a point on a plane in $\mathcal{P}^3$ 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 $\mathcal{P}^3$ 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 project this point up into $\mathcal{P}^3$. An image point $\{x, y\}$ gives the horizontal and vertical displacements in the 2D-image plane coordinate frame. Let the basis corresponding to this 2D frame in $\mathcal{E}^3$ 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 $\mathcal{E}^3$, then an image

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

point with coordinates $\{x, y\}$ can be expressed as follows in $\mathcal{E}^3$.

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

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

$$\boldsymbol{x}_a \xrightarrow{\mathcal{P}^3} X_a = \boldsymbol{x}_a + e_4 = \hat{\alpha}^i A_i, \tag{2}$$

where we defined $A_1 \equiv \boldsymbol{a}_1$, $A_2 \equiv \boldsymbol{a}_2$ and $A_3 \equiv \boldsymbol{a}_3 + e_4$. That is, A_1 and A_2 are *direction vectors*, or points at infinity, because they have no e_4 component[4]. 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 that A_1 and A_2 do not project back to $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$, respectively. For example,

$$A_1 \xrightarrow{\mathcal{E}^3} \frac{A_1 \cdot e^i}{A_1 \cdot e^4} e_i = \frac{\boldsymbol{a}_1}{0} \longrightarrow \infty \tag{3}$$

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.

$$X_a \xrightarrow{\mathcal{E}^3} \boldsymbol{x}_a = \frac{X_a \cdot e^i}{X_a \cdot e^4} e_i = \frac{\hat{\alpha}^i\,\boldsymbol{a}_i}{\hat{\alpha}^3} = x\,\boldsymbol{a}_1 + y\,\boldsymbol{a}_2 + 1\,\boldsymbol{a}_3 \tag{4}$$

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 basis. We will call this type of frame a **normalised homogeneous camera frame**. The camera frames we will use in the following are all normalised homogeneous camera frames.

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 implication later on and is *essential* to understand our derivation.

In $\mathcal{P}^3$ 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 $\mathcal{E}^3$. 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 $\{\bar{\alpha}^i\} = \{\hat{\alpha}^i\}$.

[4]This shows very nicely that a Euclidean vector interpreted as a homogeneous vector is a direction.

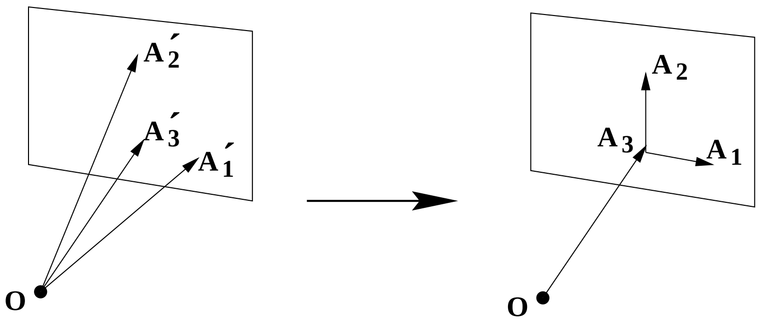


Figure 1: Transformation from general basis to a particular basis in which image points have coordinates of the type $\{x, y, 1\}$.

A general point in $\mathcal{P}^3$ 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 $\mathcal{E}^3$ we get[5]

$$\begin{aligned} X \xrightarrow{\mathcal{E}^3} \boldsymbol{x} &= \frac{X\cdot e^i}{X\cdot e^4}\, e_i = \frac{(\alpha^\mu A_\mu)\cdot e^i}{(\alpha^\mu A_\mu)\cdot e^4}\, e_i \\ &= \frac{\alpha^i}{\alpha^3+\alpha^4}\, \boldsymbol{a}_i = \frac{\bar{\alpha}^i}{1+\bar{\alpha}^4}\, \boldsymbol{a}_i \\ &= \hat{\alpha}^i\, \boldsymbol{a}_i\ ; \qquad\qquad \hat{\alpha}^i \equiv \frac{\bar{\alpha}^i}{1+\bar{\alpha}^4} \end{aligned} \tag{5}$$

Therefore, 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 $\mathcal{P}^3$.

In $\mathcal{P}^3$ a general plane is defined by three homogeneous vectors that give points on that plane. We will now show how we can transform such a general basis into a normalised homogeneous camera frame. Figure 1 shows this transformation.

Let the $\{A'_i\}$ be normalised homogeneous vectors, i.e. $A'_i \cdot e_4 = 1$. This can be assumed without loss of generality, because any homogeneous vector can be normalised without changing the point it corresponds to in $\mathcal{E}^3$. A point

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

X'_a on plane $P'_a \equiv A'_1 \wedge A'_2 \wedge A'_3$ may then be given by

$$\begin{aligned} X'_a &= \acute{\alpha}^1 A'_1 + \acute{\alpha}^2 A'_2 + \acute{\alpha}^3 A'_3 \\ &= \acute{\alpha}^1 (A'_1 - A'_3) + \acute{\alpha}^2 (A'_2 - A'_3) + (\acute{\alpha}^1 + \acute{\alpha}^2 + \acute{\alpha}^3) A'_3 \\ &\equiv \alpha^1 A_1 + \alpha^2 A_2 + \alpha^3 A_3 \\ &= X_a \end{aligned} \tag{6}$$

where we identified $A_1 \equiv A'_1 - A'_3$, $A_2 \equiv A'_2 - A'_3$ and $A_3 \equiv A'_3$. Also $\alpha^1 \equiv \acute{\alpha}^1$, $\alpha^2 \equiv \acute{\alpha}^2$ and $\alpha^3 \equiv (\acute{\alpha}^1 + \acute{\alpha}^2 + \acute{\alpha}^3)$. A_1 and A_2 are directions now, i.e. $A_1 \cdot e_4 = A_2 \cdot e_4 = 0$, but we still have $P'_a = A_1 \wedge A_2 \wedge A_3$. That is, the $\{A_i\}$, which are a normalised homogeneous camera frame, are also a valid basis for plane P'_a. As before we now have $\{\hat{\alpha}^i\} = \{\bar{\alpha}^i\}$.

3 Plane Collineation

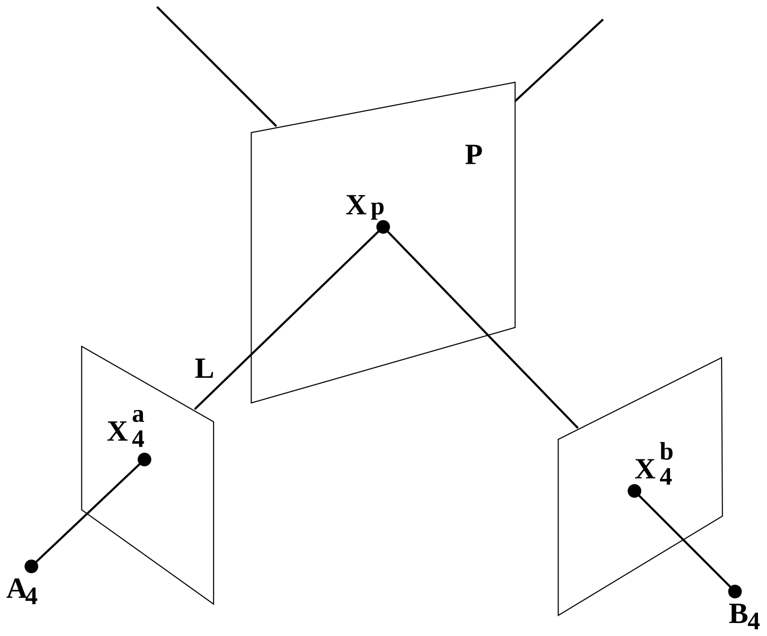


Figure 2: Schematic representation of a plane collineation. Image point X_4^a is projected to X_4^b under the P-collineation.

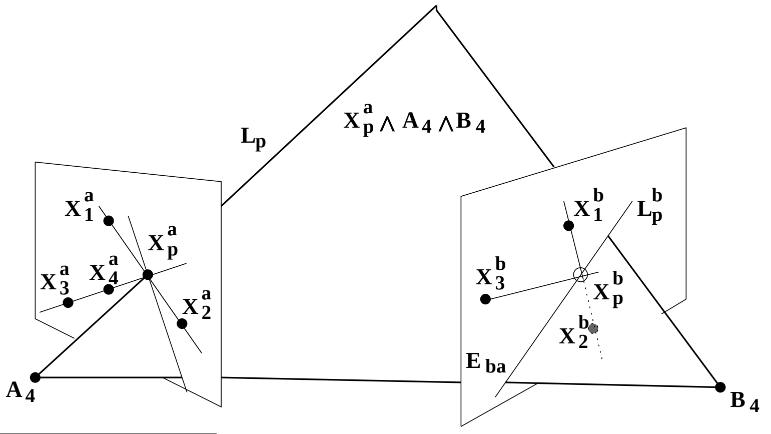
Before we can get started on the actual reconstruction algorithm, we need to derive some more mathematical objects which we will need as tools. The problem we want to solve is the following. Let us assume we have three image point matches in cameras A and B. That is, if three points in space, $\{X_i\}$, 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[6], and then we project this intersection point onto image plane B (see figure 2). This transformation can also be 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 just knowing the projections of the points $\{X_i\}$ onto image planes A and B. Faugeras presents a method in [16] for doing this[7]. We will follow this method to obtain a $3 \times 3 \times 3$ collineation tensor.

3.1 Calculating the Collineation Tensor M

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. We want to find the collineation induced by the plane $P = X_1 \wedge X_2 \wedge X_3$.

Unfortunately, it turns out that we cannot find the collineation directly as shown in figure 2, because we only know the normalised coordinates of the $\{X_i^a\}$ and $\{X_i^b\}$. Instead we have to use a two step construction.

Step 1:



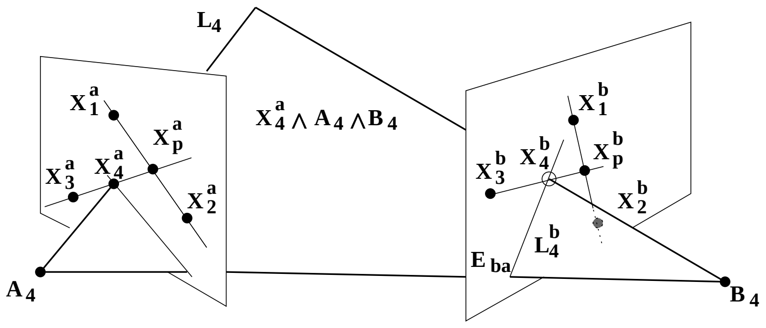
[6]Recall that A_4 is the optical centre of camera A.

[7]In [16] this method is called the **Point-Plane** procedure.

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 $X_3^a \wedge X_4^a$ 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 $\mathcal{P}^3$ 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

$$X_p^b = (X_p^a \wedge A_4 \wedge B_4) \vee (X_1^b \wedge X_2^b) \tag{7}$$

Step 2:



Now that we have calculated the point X_p^b, we can project X_4^a under the P-collineation in an analogous 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

$$X_4^b = (X_4^a \wedge A_4 \wedge B_4) \vee (X_3^b \wedge X_p^b) \tag{8}$$

Now we will perform this two step calculation, starting with X_p^a.

$$\begin{aligned} X_p^a &= (A_4 \wedge X_3^a \wedge X_4^a) \vee (X_1^a \wedge X_2^a) \\ &= [\![A_4 X_3^a X_1^a X_4^a]\!] X_2^a + [\![A_4 X_2^a X_3^a X_4^a]\!] X_1^a \\ &= \phi_{pa}^1 X_1^a + \phi_{pa}^2 X_2^a \end{aligned} \tag{9}$$

with $\phi^1_{pa} \equiv [\![A_4 X^a_2 X^a_3 X^a_4]\!]$ and $\phi^2_{pa} \equiv [\![A_4 X^a_3 X^a_1 X^a_4]\!]$. X^b_p is then found by

$$\begin{aligned}
X^b_p &= (X^a_p \wedge A_4 \wedge B_4) \vee (X^b_1 \wedge X^b_2) \\
&= -[\![X^a_p X^b_1 A_4 B_4]\!] X^b_2 + [\![X^a_p X^b_2 A_4 B_4]\!] X^b_1 \\
&= -\phi^2_{pa} [\![X^a_2 X^b_1 A_4 B_4]\!] X^b_2 + \phi^1_{pa} [\![X^a_1 X^b_2 A_4 B_4]\!] X^b_1 \\
&= \phi^1_{pb} X^b_1 + \phi^2_{pb} X^b_2
\end{aligned} \tag{10}$$

with $\phi^1_{pb} \equiv \phi^1_{pa} [\![X^a_1 X^b_2 A_4 B_4]\!]$ and $\phi^2_{pb} \equiv -\phi^2_{pa} [\![X^a_2 X^b_1 A_4 B_4]\!]$. The step from line 2 to line 3 in the previous calculation follows because $[\![X^a_i X^b_i A_4 B_4]\!] = 0$. (Recall that the pairs $\{X^a_i, X^b_i\}$ are projections of the same point in space.) We are now in a position to calculate X^b_4.

$$\begin{aligned}
X^b_4 &= (X^a_4 \wedge A_4 \wedge B_4) \vee (X^b_3 \wedge X^b_p) \\
&= -[\![X^a_4 X^b_3 A_4 B_4]\!] X^b_p + [\![X^a_4 X^b_p A_4 B_4]\!] X^b_3 \\
&= -\phi^1_{pb} [\![X^a_4 X^b_3 A_4 B_4]\!] X^b_1 - \phi^2_{pb} [\![X^a_4 X^b_3 A_4 B_4]\!] X^b_2 \\
&\quad + \phi^1_{pb} [\![X^a_4 X^b_1 A_4 B_4]\!] X^b_3 + \phi^2_{pb} [\![X^a_4 X^b_2 A_4 B_4]\!] X^b_3
\end{aligned} \tag{11}$$

If we write $X^b_4 = \beta^k_4 B_k$ then the $\{\beta^k_4\}$ are given by

$$\begin{aligned}
\beta^k_4 &= -\phi^1_{pb} [\![X^a_4 X^b_3 A_4 B_4]\!] \beta^k_1 - \phi^2_{pb} [\![X^a_4 X^b_3 A_4 B_4]\!] \beta^k_2 \\
&\quad + \Big(\phi^1_{pb} [\![X^a_4 X^b_1 A_4 B_4]\!] + \phi^2_{pb} [\![X^a_4 X^b_2 A_4 B_4]\!] \Big) \beta^k_3
\end{aligned} \tag{12}$$

At this point we should think about whether we can use the normalised image point coordinates $\{\bar{\alpha}^i_j\}$, $\{\bar{\beta}^i_j\}$ and $\{\bar{\alpha}^i_4\}$, instead of their unnormalised counterparts. If this is not the case, we cannot use equation (12). Let the $\{\bar{\phi}^i_{pa}\}$ be the $\{\phi^i_{pa}\}$ calculated from normalised coordinates. They are related in the following way.

$$\phi^1_{pa} = \alpha^3_2 \alpha^3_3 \alpha^3_4 \bar{\phi}^1_{pa} \tag{13a}$$
$$\phi^2_{pa} = \alpha^3_3 \alpha^3_1 \alpha^3_4 \bar{\phi}^2_{pa} \tag{13b}$$

Therefore, the relation between the $\{\bar{\phi}^i_{pb}\}$ and $\{\phi^i_{pb}\}$ is

$$\phi^1_{pb} = \alpha^3_2 \alpha^3_3 \alpha^3_4 \alpha^3_1 \beta^3_2 \bar{\phi}^1_{pb} \tag{14a}$$
$$\phi^2_{pb} = \alpha^3_3 \alpha^3_1 \alpha^3_4 \alpha^3_2 \beta^3_1 \bar{\phi}^2_{pb} \tag{14b}$$

Hence, the $\{\phi^i_{pb}\}$ have the term $(\alpha^3_1\alpha^3_2\alpha^3_3\alpha^3_4)$ in common. Therefore, we can write equation (12) as

$$
\begin{aligned}
\beta^k_4 &\simeq -\beta^3_2\bar{\phi}^1_{pb}\,(\alpha^i_4\beta^j_3F_{ij})\,\beta^k_1 - \beta^3_1\bar{\phi}^2_{pb}\,(\alpha^i_4\beta^j_3F_{ij})\,\beta^k_2 \\
&\quad +\Big(\beta^3_2\bar{\phi}^1_{pb}\,(\alpha^i_4\beta^j_1F_{ij}) + \beta^3_1\bar{\phi}^2_{pb}\,(\alpha^i_4\beta^j_2F_{ij})\Big)\,\beta^k_3 \\
&\simeq \bar{\phi}^1_{pb}\,(\bar{\alpha}^i_4\bar{\beta}^j_3F_{ij})\,\bar{\beta}^k_1 + \bar{\phi}^2_{pb}\,(\bar{\alpha}^i_4\bar{\beta}^j_3F_{ij})\,\bar{\beta}^k_2 \\
&\quad -\Big(\bar{\phi}^1_{pb}\,(\bar{\alpha}^i_4\bar{\beta}^j_1F_{ij}) + \bar{\phi}^2_{pb}\,(\bar{\alpha}^i_4\bar{\beta}^j_2F_{ij})\Big)\,\bar{\beta}^k_3
\end{aligned} \tag{15}
$$

where $F_{ij} \equiv [\![A_iB_jA_4B_4]\!]$ is the fundamental matrix relating cameras A and B. That is, we can find the $\{\beta^k_4\}$ up to an overall constant from the $\{\bar{\alpha}^j_i\}$, $\{\bar{\beta}^j_i\}$ and $\{\bar{\alpha}^j_4\}$. To obtain our final equation we will expand the $\{\bar{\phi}^i_{pa}\}$ and $\{\bar{\phi}^i_{pb}\}$.

$$
\begin{aligned}
\bar{\phi}^{j_1}_{pa} &= [\![A_4\bar{X}^a_4\bar{X}^a_{j_2}\bar{X}^a_{j_3}]\!] \\
&= \bar{\alpha}^i_4\,(\bar{\alpha}^{k_2}_{j_2}\bar{\alpha}^{k_3}_{j_3} - \bar{\alpha}^{k_3}_{j_2}\bar{\alpha}^{k_2}_{j_3})\,[\![A_4A_iA_{k_2}A_{k_3}]\!] \\
&\simeq \bar{\alpha}^i_4\,\bar{\lambda}^{j_1}_{a\,i}
\end{aligned} \tag{16}
$$

with

$$
\bar{\lambda}^{j_1}_{a\,k_1} \equiv (\bar{\alpha}^{k_2}_{j_2}\bar{\alpha}^{k_3}_{j_3} - \bar{\alpha}^{k_3}_{j_2}\bar{\alpha}^{k_2}_{j_3}). \tag{17}
$$

To simplify the final equation we make the following definitions.

$$
F(r,s) \equiv \bar{\alpha}^i_r\bar{\beta}^j_s\,F_{ij} \tag{18a}
$$

$$
f^b_{ir} \equiv \bar{\beta}^j_r\,F_{ij} \tag{18b}
$$

Now we can write the $\{\bar{\phi}^i_{pb}\}$ as

$$
\bar{\phi}^1_{pb} = \bar{\alpha}^i_4\,\bar{\lambda}^1_{a\,i}\,F(1,2) \tag{19a}
$$

$$
\bar{\phi}^2_{pb} = -\bar{\alpha}^i_4\,\bar{\lambda}^2_{a\,i}\,F(2,1) \tag{19b}
$$

Therefore, equation (15) becomes

$$
\beta^k_4 \simeq \bar{\alpha}^i_4\bar{\alpha}^j_4\,M^k_{ij}, \tag{20}
$$

with

$$
\begin{aligned}
M^k_{ij} \equiv \Big[&\quad \Big(F(1,2)\,\bar{\lambda}^1_{a\,i}\,\bar{\beta}^k_1 - F(2,1)\,\bar{\lambda}^2_{a\,i}\,\bar{\beta}^k_2\Big)\,f^b_{j3} \\
&- \Big(F(1,2)\,\bar{\lambda}^1_{a\,i}\,f^b_{j1} - F(2,1)\,\bar{\lambda}^2_{a\,i}\,f^b_{j2}\Big)\,\bar{\beta}^k_3\Big]
\end{aligned} \tag{21}
$$

Note that since M_{ij}^k can be calculated from the normalised image point coordinates, it is scale invariant.

To our knowledge a collineation tensor which can be used instead of the collineation matrix has not been derived before. Faugeras describes the method which we followed to find M_{ij}^k [16], but he does not obtain a simple tensor to perform the collineation projection.

3.2 Rank of M

The tensor M_{ij}^k is not of full rank. This may be seen quite easily, if we reorder the terms in equation (21).

$$\begin{aligned} M_{ij}^k &= \Big(F(1,2)\, f_{j3}^b\, \bar{\beta}_1^k - F(1,2)\, f_{j1}^b\, \bar{\beta}_3^k\Big)\, \bar{\lambda}_{a\,i}^1 \\ &\quad -\Big(F(2,1)\, f_{j3}^b\, \bar{\beta}_2^k - F(2,1)\, f_{j2}^b\, \bar{\beta}_3^k\Big)\, \bar{\lambda}_{a\,i}^2 \end{aligned} \tag{22}$$

Now, $\bar{\lambda}_{a\,3}^r$ is a linear combination of $\bar{\lambda}_{a\,1}^r$ and $\bar{\lambda}_{a\,2}^r$. The relation is

$$\bar{\lambda}_{a\,3}^{r_1} = -\frac{\bar{\alpha}_{r_2}^1}{\bar{\alpha}_{r_2}^3}\, \bar{\lambda}_{a\,1}^{r_1} - \frac{\bar{\alpha}_{r_2}^2}{\bar{\alpha}_{r_2}^3}\, \bar{\lambda}_{a\,2}^{r_1}. \tag{23}$$

Therefore, M_{3j}^k has to be linearly dependent on M_{1j}^k and M_{2j}^k. Hence, the matrices M_{ij}^1, M_{ij}^2 and M_{ij}^3 are of rank 2. Note that this rank condition is true, independent of the points used to calculate M_{ij}^k.

We can use this fact to our advantage, if we wanted to reduce the storage space needed for M_{ij}^k or increase the calculation speed of equation (20). This may be achieved by applying a set of similarity transforms to M_{ij}^k until $M_{3j}^k = 0$.

4 The Plane at Infinity

It will be very useful to see what the collineation of the plane at infinity looks like. Without loss of generality we can set $A_4 = e_4$. Also recall that A_1 and A_2 are direction vectors, i.e. have no e_4 component, by definition. Therefore, the plane at infinity P_∞ may be given by

$$P_\infty = A_1 \wedge A_2 \wedge (A_3 - A_4) \tag{24}$$

Now that we have the plane at infinity we can also find an expression for the collineation matrix associated with it. 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_∞.

$$\begin{aligned}
X_p &= (A_4 \wedge X^a) \vee P_\infty \\
&= [\![A_4 X^a]\!] \cdot \Big((A_1 \wedge A_2 \wedge A_3) - (A_1 \wedge A_2 \wedge A_4) \Big) \\
&= \sum_{i_1} \Big([\![A_4 X^a A_{i_2} A_{i_3}]\!] A_{i_1} \Big) - [\![A_4 X^a A_1 A_2]\!] A_4 \\
&= \alpha^j \sum_{i_1} [\![A_4 A_j A_{i_2} A_{i_3}]\!] A_{i_1} - \alpha^j [\![A_4 A_1 A_2 A_j]\!] A_4 \\
&\simeq \alpha^i A_i - \alpha^3 A_4
\end{aligned} \tag{25}$$

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

$$\begin{aligned}
X_p^b &= X_p \cdot B^j \; B_j \\
&\simeq \Big(\alpha^i \; A_i \cdot B^j - \alpha^3 \; A_4 \cdot B^j \Big) \; B_j \\
&= \Big(\alpha^i K^b_{j_i} - \alpha^3 \varepsilon^j_{ba} \Big) \; B_j
\end{aligned} \tag{26}$$

where $K^b_{j_i} \equiv A_i \cdot B^j$ is the 3×3 camera matrix minor of camera B, and $\varepsilon^j_{ba} \equiv A_4 \cdot B^j$ is the epipole of camera B and also the fourth column of the full camera matrix[8]. That is,

$$K^b_{j_\mu} = [K^b_{j_i}, \varepsilon^j_{ba}] \tag{27}$$

From equation (26) it follows that we can write the collineation of P_∞ as

$$\Psi^\infty_{j_i} \equiv [K^b_{j_1}, K^b_{j_2}, K^b_{j_3} - \varepsilon^j_{ba}] \tag{28}$$

where i counts the columns. Then we can write, as before

$$\beta^j_\infty \simeq \alpha^i \Psi^\infty_{j_i} \tag{29}$$

for the projection of image point X^a under the P_∞-collineation.

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 $L_a = A_4 \wedge X^a$

[8]The full camera matrix is given by $K^b_{j_\mu} = A_\mu \cdot B^j$. See [20] for details on camera matrices and epipoles.

and $L_b = B_4 \wedge X^b$ 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 $B_4 \wedge X^b$ are parallel. Obviously, this tells us something about the relative orientation of the two cameras.

Unfortunately, we cannot calculate Ψ^∞ directly from a number of $\{\alpha^i, \beta^j_\infty\}$ point pairs, because of the unknown scale in the point coordinates. However, 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. We will show later how the $\{\bar{\beta}^j_\infty\}$ may be calculated. From the definition of the camera matrix we know that

$$\beta^j = \alpha^i K^b_{j_i} + \alpha^4 \varepsilon^j_{ba}. \tag{30}$$

Furthermore, equation (29) may be rewritten as

$$\beta^j_\infty \simeq \alpha^i K^b_{j_i} - \alpha^3 \varepsilon^j_{ba} \tag{31}$$

First of all note the $\simeq$ symbol, which means that we cannot use this equation directly. Secondly, we do not actually know the $\{\alpha^i\}$, $\{\beta^i\}$ and $\{\varepsilon^i_{ba}\}$ but only the normalised coordinates[9] $\{\bar{\alpha}^i\}$, $\{\bar{\beta}^i\}$ and $\{\bar{\varepsilon}^i_{ba}\}$, because in practical applications we find all our image point coordinates in the form $\{x, y, 1\}$. Therefore, we have to rewrite equations 30 and 31 so that they are independent of the unknown scales. Equation 30 becomes

$$\begin{aligned} \bar{\beta}^j = \frac{\beta^j}{\beta^3} &= \frac{\bar{\alpha}^i \bar{K}^b_{j_i} + \bar{\alpha}^4 \bar{\varepsilon}^j_{ba}}{\bar{\alpha}^i \bar{K}^b_{3_i} + \bar{\alpha}^4} \\ \iff \qquad \bar{\alpha}^i \bar{\beta}^j \bar{K}^b_{3_i} + \bar{\alpha}^4 \bar{\beta}^j &= \bar{\alpha}^i \bar{K}^b_{j_i} + \bar{\alpha}^4 \varepsilon^j_{ba}, \end{aligned} \tag{32}$$

and equation (31) becomes

$$\begin{aligned} \bar{\beta}^j_\infty = \frac{\beta^j_\infty}{\beta^3_\infty} &= \frac{\bar{\alpha}^i \bar{K}^b_{j_i} - \bar{\varepsilon}^j_{ba}}{\bar{\alpha}^i \bar{K}^b_{3_i} - 1} \\ \iff \qquad \bar{\alpha}^i \bar{\beta}^j_\infty \bar{K}^b_{3_i} - \bar{\beta}^j_\infty &= \bar{\alpha}^i \bar{K}^b_{j_i} - \bar{\varepsilon}^j_{ba}. \end{aligned} \tag{33}$$

[9]Recall that $\bar{\alpha}^i \equiv \bar{\alpha}^i / \bar{\alpha}^3$ and similarly for the other coordinates.

Note that $\bar{K}^b_{j_i} \equiv K^b_{j_i}/\varepsilon^3_{ba}$. Subtracting equation (33) from equation (32) gives

$$\begin{aligned}
& \bar{\alpha}^i \bar{K}^b_{3_i}(\bar{\beta}^j - \bar{\beta}^j_\infty) + \bar{\alpha}^4\bar{\beta}^j + \bar{\beta}^j_\infty = \bar{\alpha}^4\varepsilon^j_{ba} + \bar{\varepsilon}^j_{ba} \\
\iff \quad & \bar{\alpha}^i \bar{K}^b_{3_i}(\bar{\beta}^j - \bar{\beta}^j_\infty) + \bar{\beta}^j_\infty - \bar{\varepsilon}^j_{ba} = \bar{\alpha}^4(\bar{\varepsilon}^j_{ba} - \bar{\beta}^j) \\
\iff \quad & \bar{\alpha}^4 = \bar{\alpha}^i \bar{K}^b_{3_i}\, \frac{\bar{\beta}^j_\infty - \bar{\beta}^j}{\bar{\beta}^j - \bar{\varepsilon}^j_{ba}} - \frac{\bar{\beta}^j_\infty - \bar{\varepsilon}^j_{ba}}{\bar{\beta}^j - \bar{\varepsilon}^j_{ba}}\ ; \quad j \in \{1,2\}.
\end{aligned} \tag{34}$$

The last equation may be written more succinctly as

$$\bar{\alpha}^4 = \bar{\alpha}^i \bar{K}^b_{3_i}\, \zeta^j_1 - \zeta^j_2\ ; \quad j \in \{1,2\}. \tag{35}$$

with

$$\zeta^j_1 \equiv \frac{\bar{\beta}^j_\infty - \bar{\beta}^j}{\bar{\beta}^j - \bar{\varepsilon}^j_{ba}} \quad ; \quad \zeta^j_2 \equiv \frac{\bar{\beta}^j_\infty - \bar{\varepsilon}^j_{ba}}{\bar{\beta}^j - \bar{\varepsilon}^j_{ba}} \tag{36}$$

Since equation (35) has to give the same result for both $j = 1$ and $j = 2$ independent of $\bar{K}^b_{3_i}$, it follows[10] 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 35 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 depth components $\bar{\alpha}^4$ 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 $\{\bar{\beta}^j_\infty\}$.

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 have also shown that the P_∞-collineation Ψ^∞ cannot be found from point pairs $\{\bar{\alpha}^i, \bar{\beta}^j_\infty\}$. However, we can find a P_∞-collineation *tensor*, M^∞ as described in equation (20), if we know the projection pairs of three points on P_∞ and the fundamental matrix (F).

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

[10]The ζs are only equal for different j if the image points they are calculated from are perfect. For real data they are not.

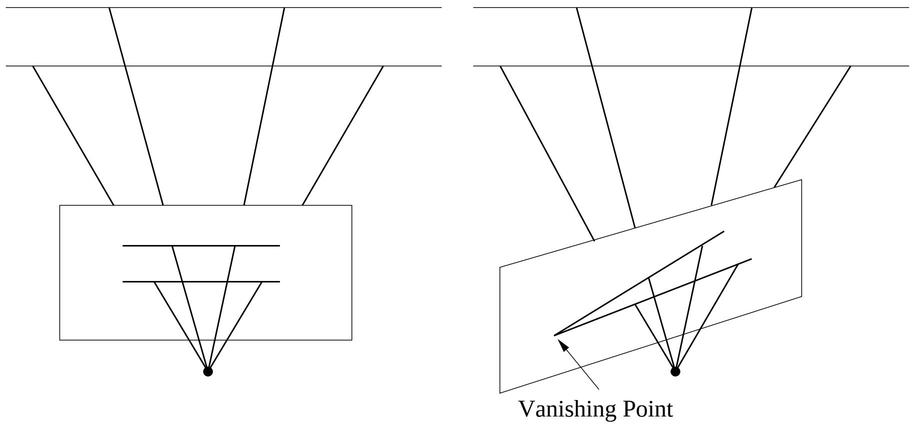


Figure 3: The figure demonstrates that the projections of two parallel world lines onto an image plane, are only parallel if the image plane is parallel to the world lines. The intersection point of the projections of two parallel world lines is called a *vanishing point*.

intersection point of the projections of two parallel world lines is called a *vanishing point* (see figure 3).

Two parallel world lines meet at infinity. In projective space $\mathcal{P}^3$ 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 $\mathcal{E}^3$ projects onto the camera used. This information can be used to find the camera calibration [18]. Here our initial goal is *not* to find the camera calibration, but to reconstruct a scene in $\mathcal{E}^3$ directly. We will show that 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.

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.

5.1.1 Vanishing Points from Parallel Line Pairs

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

$$L_u = \lambda^u_i L^i_a \quad ; \quad L_v = \lambda^v_i L^i_a \tag{37}$$

and

$$\lambda^u_{i_1} \equiv \bar{\alpha}^{i_2}_{u1}\bar{\alpha}^{i_3}_{u2} - \bar{\alpha}^{i_3}_{u1}\bar{\alpha}^{i_2}_{u2} \quad ; \quad \lambda^v_{i_1} \equiv \bar{\alpha}^{i_2}_{v1}\bar{\alpha}^{i_3}_{v2} - \bar{\alpha}^{i_3}_{v1}\bar{\alpha}^{i_2}_{v2} \tag{38}$$

are the homogeneous line coordinates. Also note that $L^{i_1}_a \equiv A_{i_2} \wedge A_{i_3}$ (see chapter [20]). Since lines L_u and L_v lie on image plane A we know that their join is simply $J = A_1 \wedge A_2 \wedge A_3$. Therefore, $J^{-1} = A^3 \wedge A^2 \wedge A^1$. The intersection point X^a_{uv} of lines L_u and L_v is then given by

$$\begin{aligned}
X^a_{uv} &= L_u \vee L_v \\
&= \Big(L_u \cdot (A^3 \wedge A^2 \wedge A^1)\Big) \cdot L_v \\
&= \sum_{i_1, j_1} \lambda^u_{i_1}\, \lambda^v_{j_1} \Big((A_{i_2} \wedge A_{i_3}) \cdot (A^3 \wedge A^2 \wedge A^1)\Big) \cdot (A_{j_2} \wedge A_{j_3}) \\
&= \sum_{i_1, j_1} \lambda^u_{i_1}\, \lambda^v_{j_1}\, A^{i_1} \cdot (A_{j_2} \wedge A_{j_3}) \\
&= \sum_{i_1} (\lambda^v_{i_2}\, \lambda^u_{i_3} - \lambda^v_{i_3}\, \lambda^u_{i_2}) A_{i_1} \\
&= \alpha^i_{uv} A_i \; ; \quad \alpha^{i_1}_{uv} \equiv (\lambda^v_{i_2}\, \lambda^u_{i_3} - \lambda^v_{i_3}\, \lambda^u_{i_2})
\end{aligned} \tag{39}$$

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

The fact that points at infinity in $\mathcal{E}^2$ are nothing special in $\mathcal{P}^2$ 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.

5.1.2 A Closer Look at Vanishing Points

It will be instructive to see what the homogeneous intersection point coordinates look like for certain sets of lines. We rewrite the $\{\alpha^i_{uv}\}$ from above as

$$\alpha^1_{uv} = \lambda^v_2\lambda^u_3 - \lambda^v_3\lambda^v_2 = \bar{\alpha}^1_{v1}\bar{\alpha}^1_{v2}\, d^3_v d^1_u - \bar{\alpha}^1_{u1}\bar{\alpha}^1_{u2}\, d^3_u d^1_v \tag{40a}$$

$$\alpha^2_{uv} = \lambda^v_3\lambda^u_1 - \lambda^v_1\lambda^v_3 = \bar{\alpha}^1_{v1}\bar{\alpha}^1_{v2}\, d^3_v d^2_u - \bar{\alpha}^1_{u1}\bar{\alpha}^1_{u2}\, d^3_u d^2_v \tag{40b}$$

$$\alpha^3_{uv} = \lambda^v_1\lambda^u_2 - \lambda^v_2\lambda^v_1 = d^1_u d^1_v \left(\frac{d^2_u}{d^1_u} - \frac{d^2_v}{d^1_v}\right) \tag{40c}$$

with

$$d^1_u \equiv \bar{\alpha}^1_{u1} - \bar{\alpha}^1_{u2}\ ;\quad d^2_u \equiv \bar{\alpha}^2_{u1} - \bar{\alpha}^2_{u2}\ ;\quad d^3_u \equiv \frac{\bar{\alpha}^2_{u2}}{\bar{\alpha}^1_{u2}} - \frac{\bar{\alpha}^2_{u1}}{\bar{\alpha}^1_{u1}} \tag{41a}$$

$$d^1_v \equiv \bar{\alpha}^1_{v1} - \bar{\alpha}^1_{v2}\ ;\quad d^2_v \equiv \bar{\alpha}^2_{v1} - \bar{\alpha}^2_{v2}\ ;\quad d^3_v \equiv \frac{\bar{\alpha}^2_{v2}}{\bar{\alpha}^1_{v2}} - \frac{\bar{\alpha}^2_{v1}}{\bar{\alpha}^1_{v1}} \tag{41b}$$

d^1_u and d^2_u define the direction of the line L_u in $\mathcal{E}^2$. Hence, $d^1_u = 0$ if L_u is parallel to A_2, and $d^2_u = 0$ if L_u is parallel to A_1.

$\bar{\alpha}^2_{u1}/\bar{\alpha}^1_{u1}$ gives the gradient of the line from the origin of $\mathcal{E}^2$ to the point $\{\bar{\alpha}^1_{u1}, \bar{\alpha}^2_{u2}\}$. Therefore, d^3_u gives the difference in gradients between the lines passing through the origin of $\mathcal{E}^2$ and $\{\bar{\alpha}^1_{u1}, \bar{\alpha}^2_{u1}\}$ or $\{\bar{\alpha}^1_{u2}, \bar{\alpha}^2_{u2}\}$, respectively. Hence, $d^3_u = 0$ if the line L_u passes through the origin of $\mathcal{E}^2$.

Now we can see the effect of some special cases of line sets $\{L_u, L_v\}$ on the homogeneous coordinates of their intersection points.

1. If L_u and L_v are parallel then $\alpha^3_{uv} = 0$. That is, the homogeneous coordinates of their intersection point are of the form $\{x, y, 0\}$.
2. If L_u and L_v are both parallel to the A_1 direction, i.e. $\bar{\alpha}^2_{u1} = \bar{\alpha}^2_{u2}$ and $\bar{\alpha}^2_{v1} = \bar{\alpha}^2_{v2}$, then $d^2_u = d^2_v = 0$. Thus, the intersection point of L_u and L_v in homogeneous coordinates is of the form $\{x, 0, 0\}$.
3. If L_u and L_v are both parallel to the A_2 direction, i.e. $\bar{\alpha}^1_{u1} = \bar{\alpha}^1_{u2}$ and $\bar{\alpha}^1_{v1} = \bar{\alpha}^1_{v2}$, then $d^1_u = d^1_v = 0$. Then the intersection point of L_u and L_v in homogeneous coordinates is of the form $\{0, y, 0\}$.
4. If L_u and L_v both pass through the origin of $\mathcal{E}^2$, i.e. the principal point of the image plane, then the homogeneous coordinates of their intersection point are of the form $\{0, 0, w\}$.

From these special cases it becomes clear that if we knew three vanishing points on image plane A of the types 2, 3 and 4 from above, and also knew how these vanishing points project onto image plane B, we would know the 3×3 camera matrix minor $K^b_{j_i}$ of camera B. This may be seen as follows.

5.1.3 A Special Set of Vanishing Points

Let $\{\alpha_n^i, \beta_n^j\}$ be three sets of matching vanishing points in cameras A and B. That is, the direction that projects to $\alpha_n^i A_i$ on camera A, projects to $\beta_n^j B_j$ on camera B. Now suppose that $\alpha_1^i = \{\alpha_1^1, 0, 0\}$, $\alpha_2^i = \{0, \alpha_2^2, 0\}$ and $\alpha_3^i = \{0, 0, \alpha_3^3\}$. Then we can write

$$\begin{aligned}
& \beta_n^j = \alpha_n^i \; A_i \cdot B^j \\
\Longleftrightarrow \quad & \beta_1^j = \alpha_1^1 \; A_1 \cdot B^j \; ; \quad \beta_2^j = \alpha_2^2 \; A_2 \cdot B^j \; ; \quad \beta_3^j = \alpha_3^3 \; A_3 \cdot B^j \\
\Longleftrightarrow \quad & \frac{\beta_n^j}{\alpha_n^n} = A_n \cdot B^j
\end{aligned} \tag{42}$$

Note that this calculation does not give the internal calibration of the cameras. It only tells us how the two camera frames are related. However, we can always take the $\{A_\mu\}$ frame as the world frame. If we furthermore knew the correct scale of the epipole on camera B, we could use it in conjunction with $K^b_{j_i}$ to find the depths of image point matches in the A-frame. The problem with this approach, apart from the unknown scale of the epipole, is that vanishing points of the form needed are hard to obtain in real life situations.

5.1.4 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.

How can we find the vanishing point of more than two projections of parallel lines? Let the $\{L_n\}$ be a set of N projections of parallel world lines onto image plane A. We are looking for the point X_v that is closest to the intersection points of all lines. X_v is an intersection point of all lines $\{L_n\}$ if $L_n \wedge X_v = 0$ for all n. That is, finding the best vanishing point means minimising a measure ε which is given by

$$\varepsilon^2 = \sum_n \Big((L_n \wedge X_v) \cdot P_a^{-1} \Big)^2 \tag{43}$$

where $P_a^{-1} \equiv (A_1 \wedge A_2 \wedge A_3)^{-1}$. Note that if $L_n \wedge X_v$ is not zero, it gives a scalar multiple of $P_a = A_1 \wedge A_2 \wedge A_3$. That is, in general we have $L_n \wedge X_v = \tau_n \, P_a$, where τ_n is a scalar. The closer X_v is to line L_n, the smaller is τ_n. Since we are interested in minimising τ_n, we take the inner product of $L_n \wedge X_v$ with P_a^{-1}, which cancels the P_a and leaves us with τ_n. We then take the sum of the squares of τ_n for all n to obtain an overall quality measure.

Let $L_n = \lambda_{ni} L_a^i$ and $X_v = \alpha_v^i A_i$, then

$$\begin{aligned}(L_n \wedge X_v) \cdot P_a^{-1} &= \sum_{i_1,j} \lambda_{ni_1} \alpha_v^j \Big((A_{i_2} \wedge A_{i_3}) \wedge A_j \Big) \cdot P_a^{-1} \\ &= \lambda_{ni}\, \alpha_v^i \Big((A_1 \wedge A_2 \wedge A_3) \cdot (A^3 \wedge A^2 \wedge A^1) \Big) \\ &= \lambda_{ni}\, \alpha_v^i\end{aligned} \tag{44}$$

The above expression is equivalent to taking the inner product of vectors[11] $\vec{l}_n \equiv [\lambda_{n1}, \lambda_{n2}, \lambda_{n3}]$ and $\vec{v} \equiv [\alpha_v^1, \alpha_v^2, \alpha_v^3]$. If we define a matrix $\Lambda \equiv [\vec{l}_1, \vec{l}_2, \ldots, \vec{l}_N]$, then $\vec{v}$ has to minimise the length of the vector $\vec{\varepsilon}$ given by

$$\vec{\varepsilon} = \Lambda\, \vec{v}^{\,T} \tag{45}$$

$\vec{\varepsilon}$ is related to the measure ε^2, which we try to minimise, via

$$\varepsilon^2 = \vec{\varepsilon}^{\,2} = \vec{v}\ (\Lambda^T \Lambda)\ \vec{v}^{\,T} \tag{46}$$

To find the best $\vec{v}$ we can now simply perform a *Singular Value Decomposition* (SVD) on the matrix $\Lambda^2 = \Lambda^T \Lambda$. This will give us the $\vec{v}$ that minimises $\Lambda^2\, \vec{v}^{\,T}$ and thus minimises ε^2. That is, we found the best fitting vanishing point in homogeneous coordinates, in the least squares sense.

Note that in [17] 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 [18] 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.

Also in both [17] and [18] vanishing points that define an orthogonal basis of the world have to be known. Our algorithm does not have this restriction. However, we also do not find the internal calibrations of the cameras.

[11]We use here the notation with the arrow above a letter to describe a vector in some orthonormal frame. This notation is used to distinguish these vectors from vectors in $\mathcal{E}^3$.

5.2 M^{∞} 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, then we can use equation (21) to find $M^{\infty}_{k_{ij}}$, the P_{∞}-collineation tensor, from the homogeneous image point coordinates of the vanishing points. Once we have $M^{\infty}_{k_{ij}}$ we can find the projections of some image points $\{\bar{\alpha}^i_n\}$ on image plane A, onto image plane B under the P_{∞}-collineation. That is,

$$\bar{\beta}^k_{n\,\infty} \simeq \bar{\alpha}^i_n \, \bar{\alpha}^j_n \; M^{\infty}_{k_{ij}} \tag{47}$$

We can now use the $\{\bar{\beta}^j_{n\,\infty}\}$ to find the $\{\zeta_n\}$ for equation (35). It is worth repeating this equation.

$$\bar{\alpha}^4_n = \bar{\alpha}^i_n \, \bar{K}^b_{3_i} \, \zeta_{1n} - \zeta_{2n} \tag{48}$$

with

$$\zeta_{1n} \equiv \frac{\bar{\beta}^j_{n\,\infty} - \bar{\beta}^j_n}{\bar{\beta}^j_n - \bar{\varepsilon}^j_{ba}} \quad ; \quad \zeta_{2n} \equiv \frac{\bar{\beta}^j_{n\,\infty} - \bar{\varepsilon}^j_{ba}}{\bar{\beta}^j_n - \bar{\varepsilon}^j_{ba}} \tag{49}$$

where j is either 1 or 2.

6 The Reconstruction Algorithm

Now that we have found M^{∞} and thus can calculate the $\{\zeta_n\}$ from equation (48), 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 our reconstruction in the frame of camera A. We are free to choose this frame. Of course, it is important to understand what effect this choice has on our final reconstruction. When we plot our final reconstructed points we will assume that the A-frame forms an orthonormal frame of $\mathcal{E}^3$. 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.

The assumption that the camera frame is an orthonormal frame of $\mathcal{E}^3$, is quite good for most modern cameras. If we still needed to find the internal camera calibration of camera A relative to an orthonormal frame of $\mathcal{E}^3$, we would need to know the projection of this orthonormal set of directions onto camera A [18]. Note that we would not need to find the internal calibration of camera B separately, since it would be found by our algorithm relative to the internal calibration of camera A.

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 (48). 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.

6.1 The Geometry

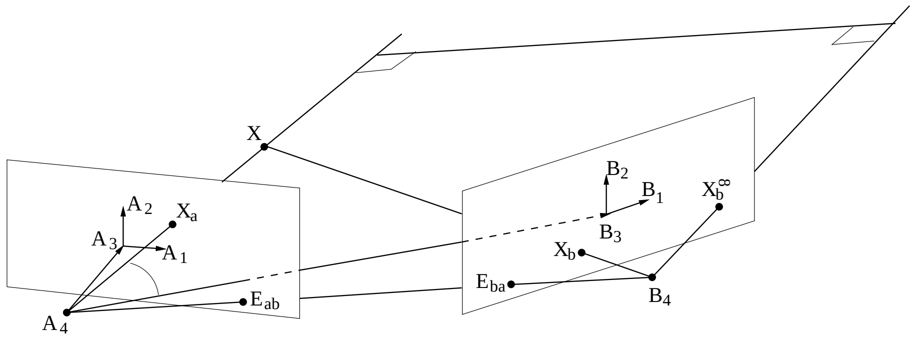


Figure 4: This figure shows the geometry behind equation (48). A point X is projected onto cameras A and B, giving images X_a and X_b, respectively. Projecting X_a onto image plane B under the P_∞-collineation gives X_b^∞. We choose A_4 to be the origin of $\mathcal{E}^3$. $K^b_{3_i}$ gives the components of A_1, A_2 and A_3 along B_3.

Before we start developing an algorithm to find the best $\{\bar{K}^b_{3_i}\}$ we should understand what varying the free parameters means geometrically. In figure 4 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 $\mathcal{E}^3$. Recall that A_1, A_2 and B_1, B_2 are direction vectors in $\mathcal{P}^3$. 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 (48) express.

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

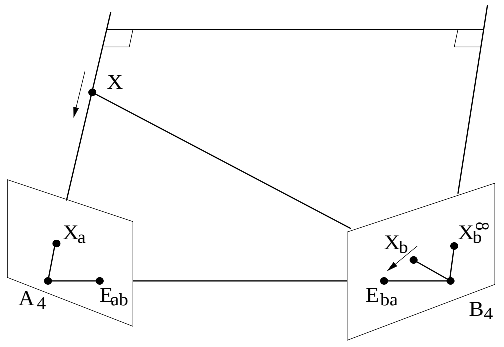
gives the ratio of the distance (in x or y direction) between X_b^∞ and X_b, and X_b and E_{ba}.

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

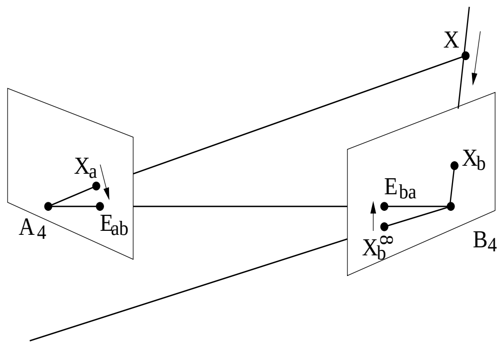
gives the ratio of the distance (in x or y direction) between X_b^∞ and E_{ba}, and X_b and E_{ba}.

6.1.1 Some Special Cases

1. If $X_b = X_b^\infty$ then X is a point on P_∞, for example a vanishing point. In this case $\zeta^j_{1n} = 0$ and $\zeta^j_{2n} = 1$. Hence, equation (48) gives $\bar{\alpha}^4_n = -1$. From equation (5) it follows that this does indeed give a point on P_∞.

2. 

Suppose we move a world point X along a line towards A_4 (see figure). In that case, X_a and X_b^∞ will stay constant, but X_b will move towards E_{ba}. Therefore, $\zeta^j_{1n} \to \infty$ and $\zeta^j_{2n} \to \infty$, and thus $\bar{\alpha}^4 \to \infty$ (from equation (48)) and $\hat{\alpha}^i \to 0$. That is, in the limit that $X = A_4$, X will be reconstructed to lie at the origin, which is A_4.

3. 

Now suppose we move a world point X along a line towards B_4 (see figure). Now X_b will stay constant, whereas $X_a \to E_{ab}$ and $X_b^\infty \to E_{ba}$. Therefore, $\zeta^j_{1n} \to -1$ and $\zeta^j_{2n} \to 0$. In the limit that $X = B_4$ we have $X_a = E_{ab}$ and thus equation (48) gives $\bar{\varepsilon}^4_{ab} = -\bar{\varepsilon}^i_{ab}\,\bar{K}^b_{3_i}$. That is, we have found the position of B_4 in the A-frame.

The equation for the projective depth of the $\{\bar{\varepsilon}^i_{ab}\}$ given above can also be derived as follows.

$$\begin{aligned} & B_4 = B_4 \cdot A^\mu \, A_\mu = \varepsilon^\mu_{ab} \, A_\mu \\ \Rightarrow \quad & B_4 \cdot B^j = \varepsilon^\mu_{ab} \, A_\mu \cdot B^j \\ \Rightarrow \quad & 0 = \varepsilon^\mu_{ab} \, K^b_{j_\mu} \; ; \quad K^b_{j_\mu} \equiv A_\mu \cdot B^j \\ \Rightarrow \quad & \varepsilon^4_{ab} = -\varepsilon^i_{ab} \frac{K^b_{j_i}}{K^b_{j_4}} \end{aligned} \tag{50}$$

What we have done here is first to express B_4 in the A-frame. Then we project B_4 onto image plane B. In the third line we use the fact that $B_4 \cdot B^j = 0$ by definition. The resultant equation has to be valid for each $j \in \{1, 2, 3\}$. If we choose $j = 3$ we can write

$$\bar{\varepsilon}^4_{ab} = -\bar{\varepsilon}^i_{ab} \, \bar{K}^b_{3_i}. \tag{51}$$

6.1.2 The Meaning of the Free Parameters

Let us return to figure 4. Recall that $K^b_{3_i} = A_i \cdot B^3$, that is, it gives the components of the $\{A_i\}$ along B_3. Therefore, varying the $\{K^b_{3_i}\}$ means that we are moving B_3, which is the principal point on image plane B. Since X^∞_b cannot change when we vary $K^b_{3_i}$ the relation between B_3 and B_4 is fixed. Thus, changing B_3 means changing B_4. In this respect, finding the correct $\{K^b_{3_i}\}$ 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 the relative sizes of the $\{\bar{K}^b_{3_i}\}$ 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 $\bar{K}^b_{3_i}$.

6.2 The Minimisation Function

We mentioned before that we will use our knowledge of parallel lines once again to constrain the $\{\bar{K}^b_{3_i}\}$ from equation (48). Let $L^a_u = X^a_{u1} \wedge X^a_{u2}$ and $L^a_v = X^a_{v1} \wedge X^a_{v2}$ be the projections of two parallel world lines onto image plane A. In general we define world points and their corresponding images on image plane A as

$$\left.\begin{aligned} X_{ur} \equiv \bar{\alpha}^\mu_{ur} A_\mu \; ; \quad X^a_{ur} \equiv \bar{\alpha}^i_{ur} A_i \\ X_{vr} \equiv \bar{\alpha}^\mu_{vr} A_\mu \; ; \quad X^a_{vr} \equiv \bar{\alpha}^i_{vr} A_i \end{aligned}\right\} r \in \{1, \ldots, n\}. \tag{52}$$

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 M^∞, then we can calculate the corresponding ζs from equation (49). Equation (48) 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

$$X^\infty_u \equiv L_u \vee P_\infty \; ; \quad X^\infty_v \equiv L_v \vee P_\infty . \tag{53}$$

Lines L_u and L_v are parallel iff

$$X^\infty_u \wedge X^\infty_v = 0 \tag{54}$$

Instead of using this condition we could also project L_u and L_v into $\mathcal{E}^3$, and then check that they are parallel. However, projecting into $\mathcal{E}^3$ means dividing through by 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.

6.2.1 Finding the minimisation parameters

We will now derive an expression for X^∞_u in terms of the $\{\bar{\alpha}^i_{u1}, \bar{\alpha}^i_{u2}\}$.

$$\begin{aligned}
X^\infty_u &= L_u \vee P_\infty = (\bar{\alpha}^\mu_{u1}\, \bar{\alpha}^\nu_{u2}\, A_\mu \wedge A_\nu) \vee P_\infty \\
&= \Big(\bar{\lambda}^u_{\mu_1\mu_2}\, A_{\mu_1}\wedge A_{\mu_2}\Big) \vee \Big(A_1\wedge A_2 \wedge (A_3 - A_4)\Big) \\
&= \quad \bar{\lambda}^u_{\mu_1\mu_2}\Big(\, [\![A_{\mu_1}A_{\mu_2}A_1A_2]\!]\, A_3 + [\![A_{\mu_1}A_{\mu_2}A_3A_1]\!]\, A_2 \\
&\qquad\qquad + [\![A_{\mu_1}A_{\mu_2}A_2A_3]\!]\, A_1\Big) \\
&\quad - \bar{\lambda}^u_{\mu_1\mu_2}\Big(\, [\![A_{\mu_1}A_{\mu_2}A_1A_2]\!]\, A_4 + [\![A_{\mu_1}A_{\mu_2}A_4A_1]\!]\, A_2 \\
&\qquad\qquad + [\![A_{\mu_1}A_{\mu_2}A_2A_4]\!]\, A_1\Big) \\
&\simeq (\bar{\lambda}^u_{13} + \bar{\lambda}^u_{14})\, A_1 + (\bar{\lambda}^u_{23} + \bar{\lambda}^u_{24})\, A_2 + \bar{\lambda}^u_{34}\, (A_3 - A_4) \\
&= \chi^i_u\, A^\infty_i
\end{aligned} \tag{55}$$

where

$$\chi^i_u \equiv (\bar{\lambda}^u_{i3} + \bar{\lambda}^u_{i4}) \; ; \quad \bar{\lambda}^u_{\mu_1\mu_2} \equiv \bar{\alpha}^{\mu_1}_{u1}\, \bar{\alpha}^{\mu_2}_{u2} - \bar{\alpha}^{\mu_2}_{u1}\, \bar{\alpha}^{\mu_1}_{u2}, \tag{56}$$

and

$$A_1^\infty \equiv A_1 \; ; \quad A_2^\infty \equiv A_2 \; ; \quad A_3^\infty \equiv A_3 - A_4 \tag{57}$$

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 (48) will be written as

$$\bar{\alpha}_n^4 = \bar{\alpha}_n^i \, \zeta_{1n} \, \varphi_i - \zeta_{2n}. \tag{58}$$

Substituting equation (58) into equation (56) gives

$$\begin{aligned} \chi_u^i &= \bar{\alpha}_{u1}^i \, \bar{\alpha}_{u2}^4 - \bar{\alpha}_{u2}^i \, \bar{\alpha}_{u1}^4 + \bar{\alpha}_{u1}^i - \bar{\alpha}_{u2}^i \\ &= \quad \bar{\alpha}_{u1}^i \, (\bar{\alpha}_{u2}^j \, \zeta_{1\,u2} \, \varphi_j - \zeta_{2\,u2}) \\ &\quad - \; \bar{\alpha}_{u2}^i \, (\bar{\alpha}_{u1}^j \, \zeta_{1\,u1} \, \varphi_j - \zeta_{2\,u1}) \; + \; \bar{\alpha}_{u1}^i - \bar{\alpha}_{u2}^i \\ &= \Big(\zeta_{1\,u2} \, \bar{\alpha}_{u1}^i \, \bar{\alpha}_{u2}^j - \zeta_{1\,u1} \, \bar{\alpha}_{u2}^i \, \bar{\alpha}_{u1}^j\Big) \, \varphi_j \\ &\quad + \bar{\alpha}_{u1}^i \, (1 - \zeta_{2\,u2}) - \bar{\alpha}_{u2}^i \, (1 - \zeta_{2\,u1}) \end{aligned} \tag{59}$$

This equation may be written more succinctly as

$$\chi_u^i = D_u^{ij} \, \varphi_j + p_u^i \tag{60}$$

with

$$\begin{aligned} D_u^{ij} &\equiv \zeta_{1\,u2} \, \bar{\alpha}_{u1}^i \, \bar{\alpha}_{u2}^j - \zeta_{1\,u1} \, \bar{\alpha}_{u2}^i \, \bar{\alpha}_{u1}^j \\ p_u^i &\equiv \bar{\alpha}_{u1}^i \, (1 - \zeta_{2\,u2}) - \bar{\alpha}_{u2}^i \, (1 - \zeta_{2\,u1}) \end{aligned} \tag{61}$$

The reason for defining the $\{D_u^{ij}\}$ and $\{p_u^i\}$ is, that they can be calculated for every parallel line pair available *before* we minimise over the $\{\varphi_j\}$. In this way we reduce the calculation time at each minimisation step.

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

$$\begin{aligned} X_u^\infty \wedge X_v^\infty &= (\chi_u^i \, A_i^\infty) \wedge (\chi_v^j \, A_j^\infty) \\ &= \sum_{i_1} (\chi_u^{i_2} \, \chi_v^{i_3} - \chi_u^{i_3} \, \chi_v^{i_2}) \; A_{i_2}^\infty \wedge A_{i_3}^\infty \\ &= \Lambda_i^{uv} \, L_\infty^i \; ; \quad L_\infty^{i_1} \equiv A_{i_2}^\infty \wedge A_{i_3}^\infty \end{aligned} \tag{62}$$

with

$$\Lambda_{i_1}^{uv} \equiv \chi_u^{i_2}\,\chi_v^{i_3} - \chi_u^{i_3}\,\chi_v^{i_2} \tag{63}$$

Each of the $\{\Lambda_i^{uv}\}$ has to be zero if $X_u^\infty \wedge X_v^\infty = 0$. Therefore, from an analytical point of view, the expression we should try to minimise for each parallel line pair $\{L_u, L_v\}$ is

$$\Delta^{uv} :\ \varphi_j \longrightarrow \sum_{i=0}^{3} (\Lambda_i^{uv})^2. \tag{64}$$

6.2.2 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 (63) they should be normalised to improve the precision of the equation on a computer. We normalise the $\{\chi_u^i\}$ in the following way.

$$\hat{\chi}_u^i \equiv \frac{\chi_u^i}{\sqrt{\sum_i (\chi_u^i)^2}} \tag{65}$$

Therefore, the minimisation function we will use is

$$\Delta^{uv} :\ \varphi_j \longrightarrow \sum_{i=0}^{3} (\hat{\Lambda}_i^{uv})^2. \tag{66}$$

where

$$\hat{\Lambda}_{i_1}^{uv} \equiv \hat{\chi}_u^{i_2}\,\hat{\chi}_v^{i_3} - \hat{\chi}_u^{i_3}\,\hat{\chi}_v^{i_2} \tag{67}$$

Since the $\{\hat{\chi}_u^i\}$ are normalised it might seem possible on first sight to use as minimisation function

$$\Delta^{uv} :\ \varphi_j \longrightarrow \sum_{i=0}^{3} (\hat{\chi}_u^i - \hat{\chi}_v^i)^2. \tag{68}$$

This would be faster to calculate and also have much simpler derivatives. However, this equation is sensitive to an overall sign change of the $\{\hat{\chi}^i\}$, but we are only interested in whether two lines are parallel, not in whether the vectors that define them point in the same or in opposite directions.

6.2.3 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.

The partial derivatives of the $\{\hat{\chi}^i_u\}$ are

$$\partial_{\varphi_r} \hat{\chi}^i_u = \frac{D^{ir}_u}{\sqrt{\sum_k (\chi^k_u)^2}} - \frac{\chi^i_u \sum_k (\chi^k_u D^{kr}_u)}{\sqrt[3]{\sum_k (\chi^k_u)^2}}, \tag{69}$$

and the partial derivatives of the $\{\hat{\Lambda}^{uv}_i\}$ are

$$\partial_{\varphi_r} \hat{\Lambda}^{uv}_{i_1} = (\partial_{\varphi_r} \hat{\chi}^{i_2}_u)\, \hat{\chi}^{i_3}_v + \hat{\chi}^{i_2}_u\, (\partial_{\varphi_r} \hat{\chi}^{i_3}_v) - (\partial_{\varphi_r} \hat{\chi}^{i_3}_u)\, \hat{\chi}^{i_2}_v - \hat{\chi}^{i_3}_u\, (\partial_{\varphi_r} \hat{\chi}^{i_2}_v). \tag{70}$$

Now we can calculate the partial derivatives of the minimisation function Δ^{uv}.

$$\partial_{\varphi_r} \Delta^{uv} = 2 \sum_{i=0}^{3} (\hat{\Lambda}^{uv}_i\, \partial_{\varphi_r} \hat{\Lambda}^{uv}_i) \tag{71}$$

6.2.4 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.

If we choose to fix the depth of point X_{u1} it follows from equation (58) that the following condition must hold.

$$\begin{aligned} 0 &= \zeta_{1\,u1}\, \bar{\alpha}^i_{u1}\, \varphi_i - \zeta_{2\,u1} - \bar{\alpha}^4_{u1} \\ &= \xi^i_{u1}\, \varphi_i - \omega_{u1} \end{aligned} \tag{72}$$

with

$$\xi^i_{u1} \equiv \zeta_{1\,u1}\, \bar{\alpha}^i_{u1}\ ; \quad \omega_{u1} \equiv \zeta_{2\,u1} + \bar{\alpha}^4_{u1} \tag{73}$$

With regard to equation (5) a good choice for $\bar{\alpha}^4_{u1}$ is -0.5, which means that $\hat{\alpha}^3 = 2$. However, if the point we chose to fix has a much larger depth value than the other points we are trying to reconstruct, then some points may be reconstructed to lie *behind* the optical centre. In this case, we will invariably get a bad reconstruction.

We can rewrite equation (72) as

$$\bar{\xi}^i_{u1}\,\varphi_i = 1\ ; \quad \bar{\xi}^i_{u1} \equiv \frac{\xi^i_{u1}}{\omega_{u1}} \tag{74}$$

Of course, we have to make sure that $\omega_{u1} \neq 0$, which means we have to be somewhat careful with the choice of $\bar{\alpha}^4_{u1}$.

Our minimisation routine should only search over that parameter space where equation (74) is satisfied. We can achieve this by reparameterising the equation.

$$\bar{\xi}^i_{u1}\,(\tau_1\,\Phi^1_i + \tau_2\,\Phi^2_i + \tau_3\,\Phi^3_i) = 1\ ; \quad \tau_3 \equiv 1 \tag{75}$$

with

$$\varphi_j = \tau_i\,\Phi^i_j \tag{76}$$

and

$$\bar{\xi}^i_{u1}\,\Phi^1_i = 0\ ; \quad \bar{\xi}^i_{u1}\,\Phi^2_i = 0\ ; \quad \bar{\xi}^i_{u1}\,\Phi^3_i = 1 \tag{77}$$

We have replaced the $\{\varphi_i\}$ with the $\{\tau_i\}$. Therefore, $\{\tau_1, \tau_2\}$ are now the free parameters, while τ_3 is fixed at unity.

The question now is how we can find the appropriate Φ^j_i matrix. First of all we can set the vector Φ^3_i to be the inverse of the vector $\bar{\xi}^i_{u1}$.

$$\Phi^3_i = \frac{\bar{\xi}^i_{u1}}{\sum_j (\bar{\xi}^j_{u1})^2} \quad \Longrightarrow \quad \bar{\xi}^i_{u1}\,\Phi^3_i = \frac{\sum_i (\bar{\xi}^i_{u1})^2}{\sum_j (\bar{\xi}^j_{u1})^2} = 1 \tag{78}$$

We find the remaining first two rows of Φ^j_i with the help of an SVD. We do this by creating a 3×3 matrix H whose three rows are all given by Φ^3_i. That is, $H \equiv [\Phi^3_i, \Phi^3_i, \Phi^3_i]$. Therefore, H is of rank 1 and has a nullity of 2. Applying an SVD to H will find a set of three orthogonal vectors, two of which span the null space of H, while the remaining one is just a scaled version of Φ^3_i. We first should find that scale and apply it to H. The null space of H is exactly the space we want our minimisation routine to search over. Hence, we set Φ^1_i and Φ^2_i to be the correctly scaled null space vectors found with the SVD. This satisfies equation (77), and therefore equation (75) will stay unchanged when varying τ_1 and τ_2.

6.2.5 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 (21)). For the calculation of the F 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 [26] 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 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.

6.2.6 Calculating the Fundamental Matrix

In recent times a lot of effort has gone into the analysis of the fundamental matrix (F) and the trifocal tensor (T), in order to find constraints so that they may be calculated as accurately as possible [26, 27, 23, 28, 29, 30, 31]. However, as our experimental results will show, the quality of F is not of particular importance for our reconstruction algorithm. In retrospect this will justify the simple calculation method we use for F.

We calculate F with a simple SVD method by writing the components of F as a column vector $\vec{f}$. If we have N point matches $\{\bar{\alpha}^i_n, \bar{\beta}^i_n\}$ then the F we look for has to satisfy

$$\bar{\alpha}^i_n\, \bar{\beta}^j_n\, F_{ij} = 0, \tag{79}$$

for all n. In matrix notation this can written as

$$A\,\vec{f} = 0 \tag{80}$$

with

$$A \equiv \begin{pmatrix} \bar{\alpha}_1^1 \bar{\beta}_1^1 & \bar{\alpha}_1^1 \bar{\beta}_1^2 & \bar{\alpha}_1^1 \bar{\beta}_1^3 & \bar{\alpha}_1^2 \bar{\beta}_1^1 & \dots & \bar{\alpha}_1^3 \bar{\beta}_1^3 \\ \bar{\alpha}_2^1 \bar{\beta}_2^1 & \bar{\alpha}_2^1 \bar{\beta}_2^2 & \bar{\alpha}_2^1 \bar{\beta}_2^3 & \bar{\alpha}_2^2 \bar{\beta}_2^1 & \dots & \bar{\alpha}_2^3 \bar{\beta}_2^3 \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ \bar{\alpha}_N^1 \bar{\beta}_N^1 & \bar{\alpha}_N^1 \bar{\beta}_N^2 & \bar{\alpha}_N^1 \bar{\beta}_N^3 & \bar{\alpha}_N^2 \bar{\beta}_N^1 & \dots & \bar{\alpha}_N^3 \bar{\beta}_N^3 \end{pmatrix} \tag{81}$$

For real data there will typically not be an F that satisfies equation (79) perfectly. Therefore, what we try to calculate is the F that satisfies these conditions as well as possible. This is equivalent to finding the vector that has the least influence on the range of A. This can be achieved by performing an SVD on A.

$$A = U\, D\, V^T, \tag{82}$$

where U and V are orthogonal matrices and D is a diagonal matrix [32]. The column vector in V that corresponds to the smallest diagonal value in D is the $\vec{f}$ that we are looking for. Note that at least nine point matches have to be known to find F in this way.

As was shown in chapter [20], F is of rank 2. An F found with the above method from real image point matches, usually does not satisfy this constraint. An indication that the image point matches are particularly bad is that there are two or more diagonal values in D of the same order of magnitude.

A linear method to enforce the rank of F is to project the initial F to the nearest F that satisfies the rank constraint. This may be done by performing an SVD on F, i.e. $F = UDV^T$, then setting the smallest diagonal value in D to zero, and recalculating $F = UDV^T$ with the changed D. However, this did not have any significant effect on our reconstructions.

6.3 The Minimisation Routine

We used a modified version of the *conjugate gradient* method to perform the minimisation. This modified version is called *MacOpt* and was developed by David MacKay [33]. It makes a number of improvements over the conjugate gradient method as given in [32]. We will list the most important modifications in the following.

- The initial step size in the line minimisation as given in [32] may be too big, which can result in a lot of wasted computing power. MacOpt rectifies this in two ways:

1. the initial step size of any line minimization is inversely proportional to the gradient,
2. the constant of proportionality is adopted during the minimisation.

- MacOpt uses the gradient information also for the line search. In this way the line minimum can be bracketed with only two gradient evaluations.
- MacOpt does not evaluate the function at all, but only uses the gradient information.
- The general purpose minimiser in [32] gives very high precision in the line minimisations, which is actually not necessary. MacOpt only brackets the minimum and then guesses where it is by linear interpolation.

To optimise the minimisation process we calculate the $\{D^{ij}\}$ and $\{p^i\}$ from equation (61) for each parallel line pair, *before* we start MacOpt. During the minimisation process we can then calculate the $\{\chi^i\}$ quickly with equation (60). Unfortunately, we cannot precalculate anything else because of the normalisation of the $\{\chi^i\}$ which we need to perform.

MacOpt assumes that the minimisation surface is fundamentally convex with no local minima. However, our surface is not of that shape. 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.

The problems that may occur with MacOpt and a general discussion of the minimisation surface, in particular with relation to the reconstruction, is demonstrated by the program **MVT**. This program can be downloaded from C.Perwass' home page: `www.perwass.de`.

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 tensor M^∞.

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 M^∞.

Step 5: We can now find the $\{K^b_{3_i}\}$ by minimising equation (64) or equation (66).

Step 6: Once we have found $K^b_{3_i}$ we can use it in conjunction with M^∞ in equation (58) to reconstruct any other image point matches for this camera setup. Note that this method of finding the image point depths saves us from performing an additional triangulation [34], which would be necessary if we first calculated the camera calibrations explicitly and then tried to find the image point depths.

Figure 5 shows the data that has to be known and calculated as input to our minimisation routine. The image points and parallel line indices are the source data. The latter index which image point pairs form parallel lines. The inputs to the minimisation routine are the image points, the parallel line indices, the epipoles and the image points projected through the plane at infinity. The fundamental matrix and the vanishing points are only intermediate calculations to find M^∞. M^∞ is also only needed once to project the image points on image plane A onto image plane B under the P_∞-collineation.

There are quite a number of factors that influence the reconstruction. These are shown in figure 6. We can distinguish between two types of influences: those on which we have no influence once we are given our source data (i.e. image points and parallel lines), and those which depend on how we deal with the source data. The former are indicated by red, rounded boxes and the latter

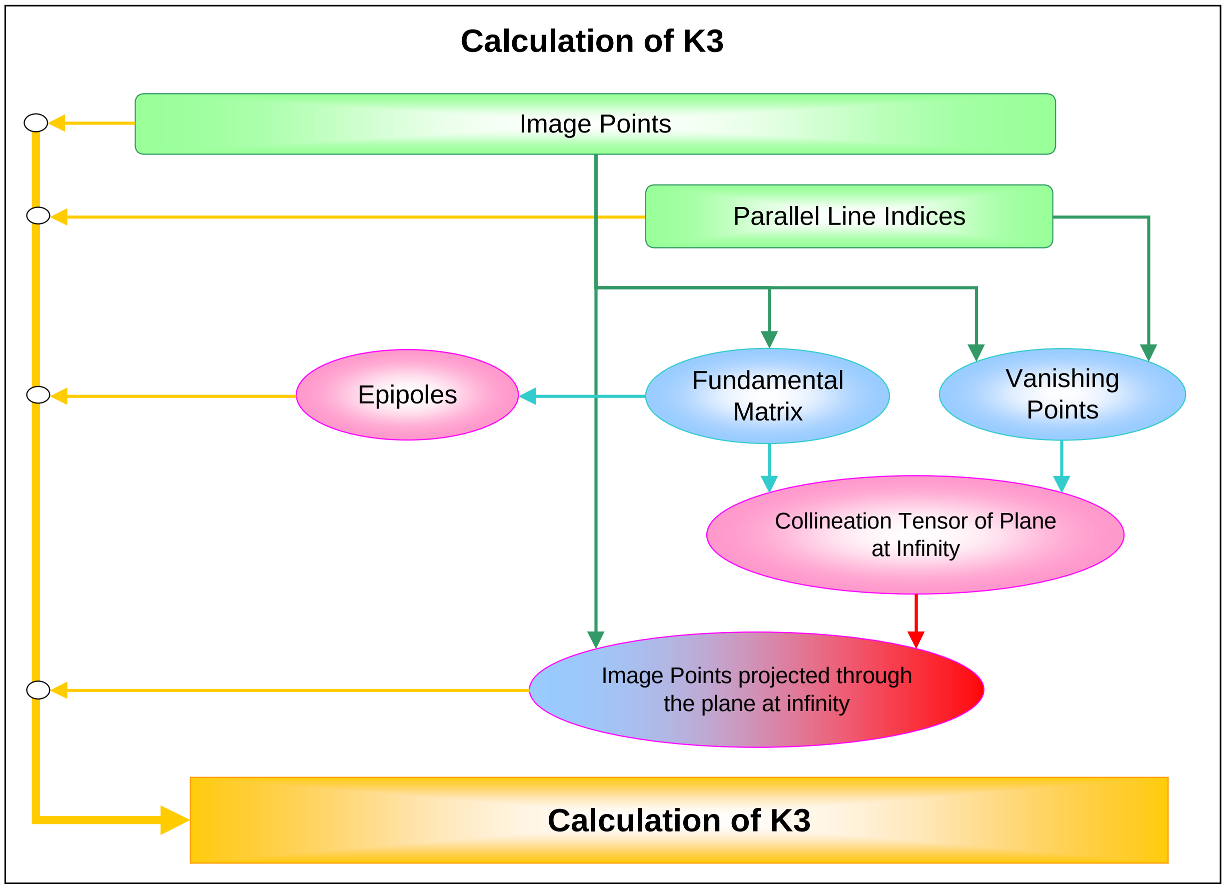


Figure 5: Data needed for calculation of $K^b_{3_j}$.

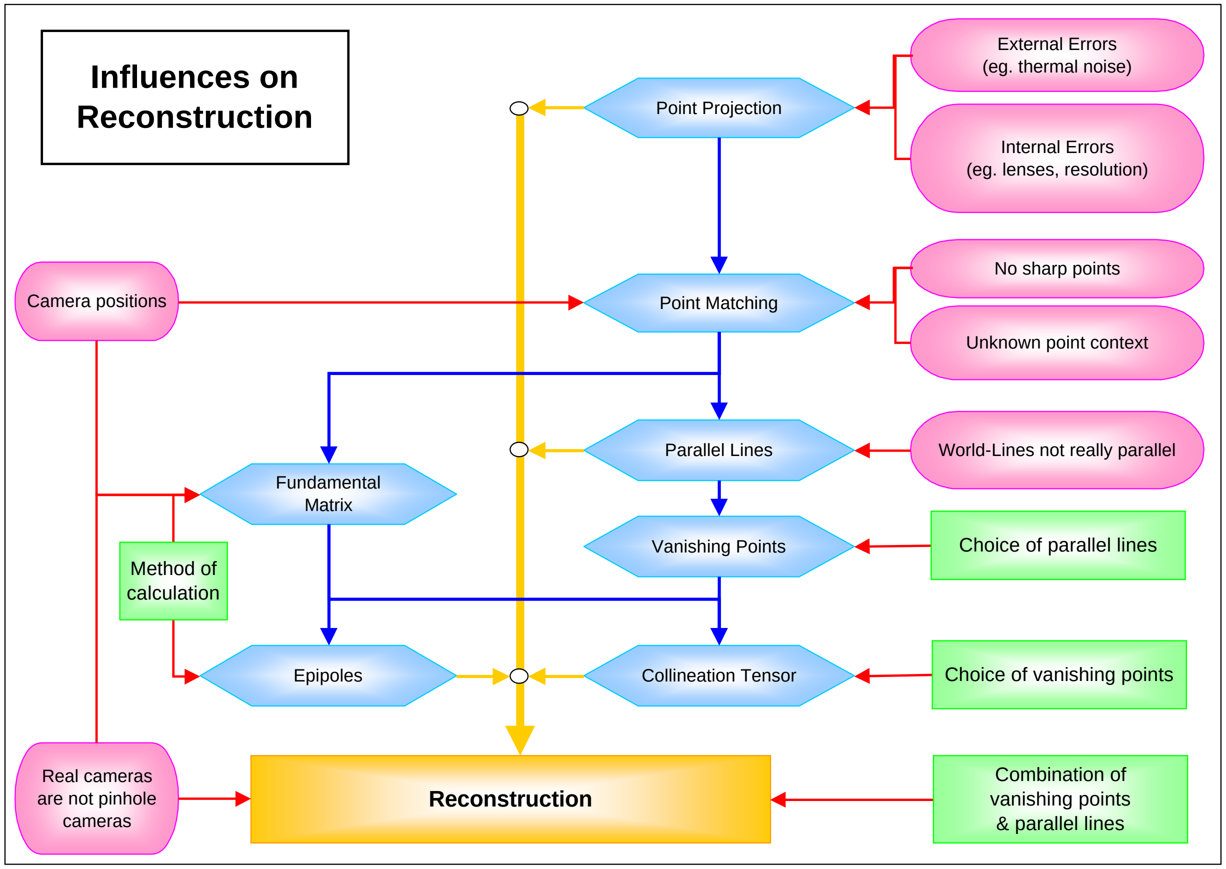

Influences on Reconstruction
Point Projection
External Errors (eg. thermal noise)
Internal Errors (eg. lenses, resolution)
Camera positions
Point Matching
No sharp points
Unknown point context
Parallel Lines
World-Lines not really parallel
Fundamental Matrix
Vanishing Points
Choice of parallel lines
Method of calculation
Epipoles
Collineation Tensor
Choice of vanishing points
Real cameras are not pinhole cameras
Reconstruction
Combination of vanishing points & parallel lines

Figure 6: Influences on the reconstruction. Red, rounded boxes show those influences on which we have no influence whereas green, square boxes indicate choices that can be made.

by green, square ones. The pointed boxes indicate the calculations which we need to perform before we can start the minimisation routine.

If we have real data we can only try to improve the reconstruction by varying the choice of parallel lines, the choice of vanishing points and the method of calculation of the fundamental matrix and the epipoles. However, if we use synthetic data, we have a handle on all influences shown.

7.1 Synthetic Data

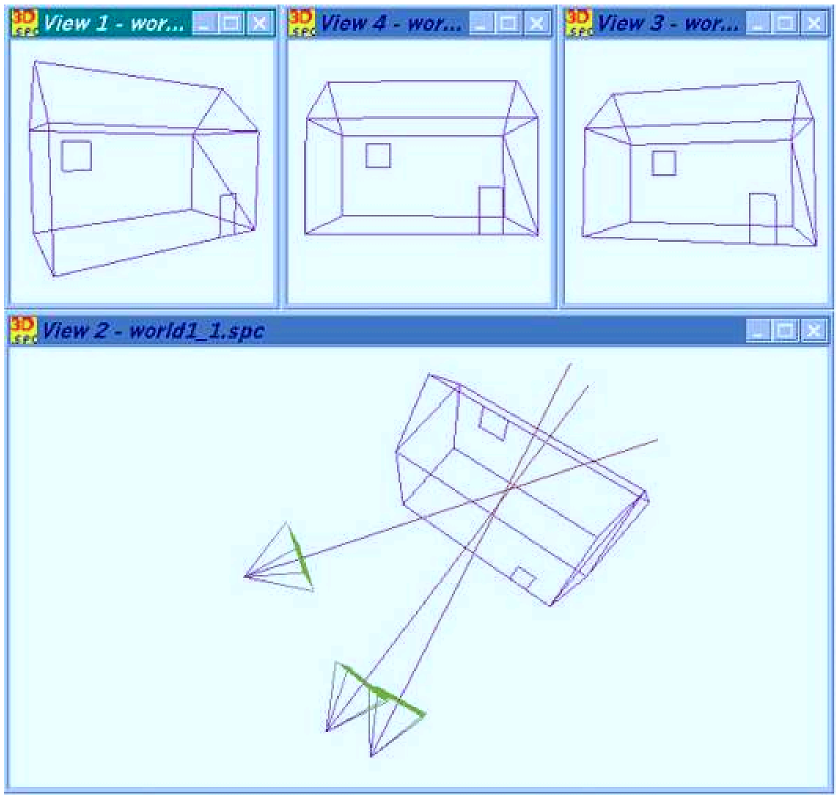


Figure 7: The synthetic data was created from projections of the house onto the cameras.

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

The lower picture in figure 7 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 $\{K^b_{3_i}\}$ 100 times, each time with different errors, to obtain a statistically meaningful result. Each calculation of the $\{K^b_{3_i}\}$ 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 $\{K^b_{3_i}\}$. 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 $\{K^b_{3_i}\}$), and also over all calculations of the $\{K^b_{3_i}\}$ for each mean deviation of the induced error. The former will be called the "RMS/Trial" and the latter the "Total RMS".

Figure 8 shows the results when using an orthogonal set of vanishing points and figure 9 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 cameras 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 M^∞ 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.

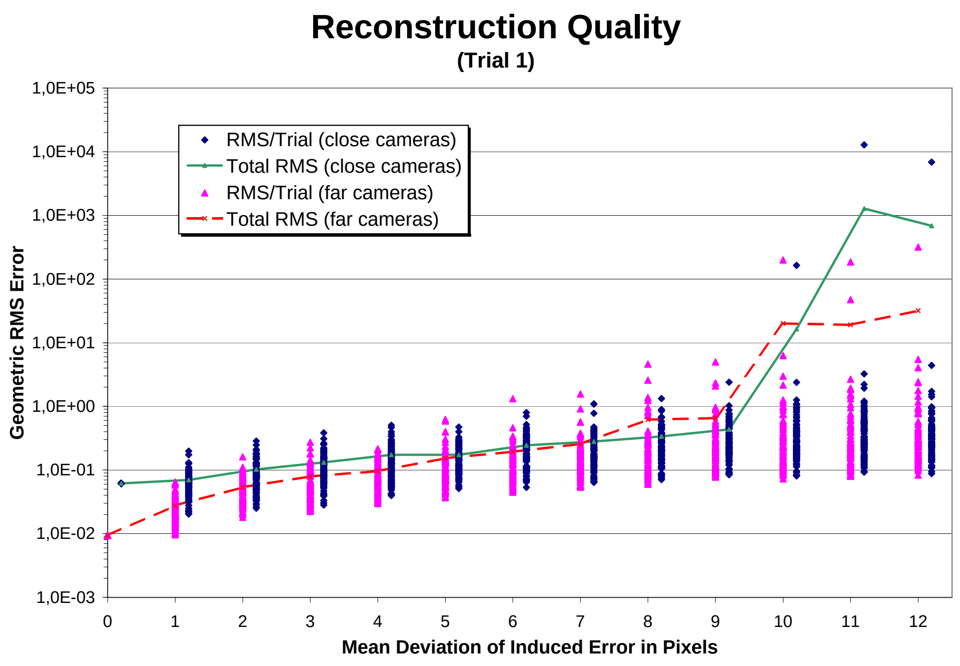


Figure 8: Comparison of reconstruction quality for first trial.

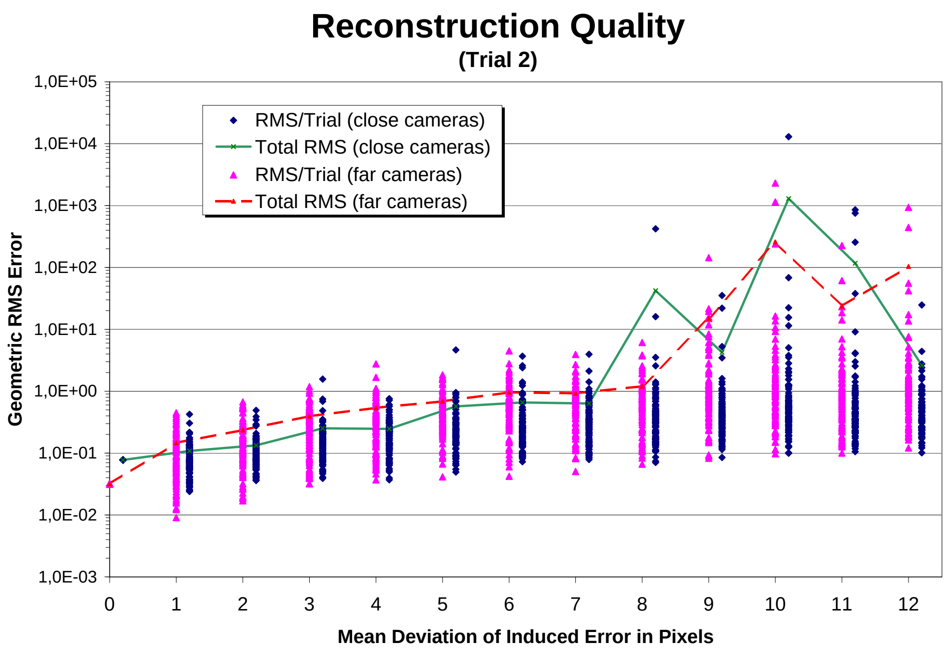


Figure 9: Comparison of reconstruction quality for second trial.

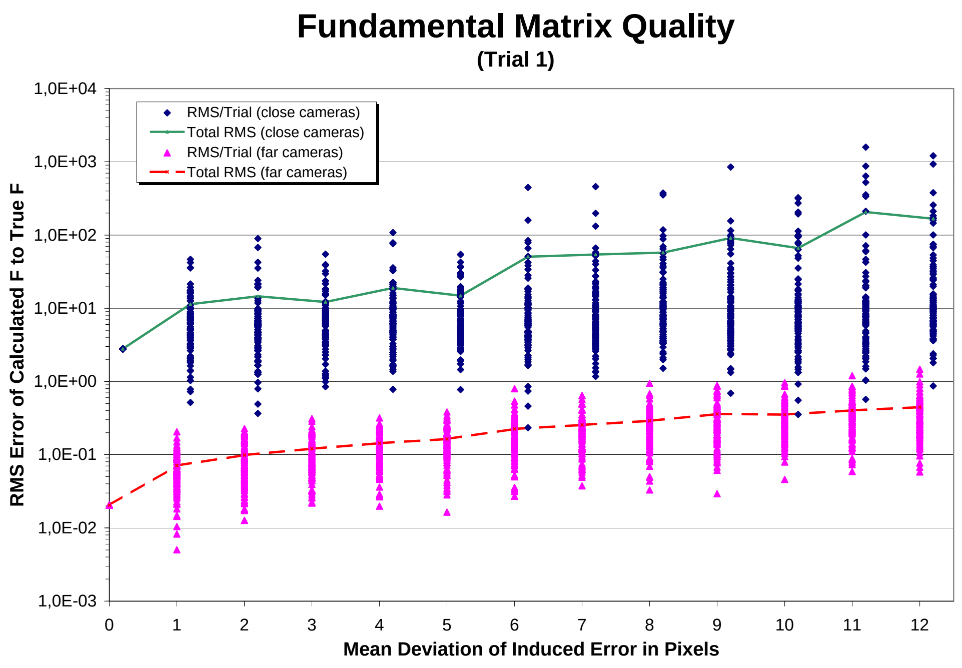


Figure 10: Comparison of fundamental matrix quality for first trial.

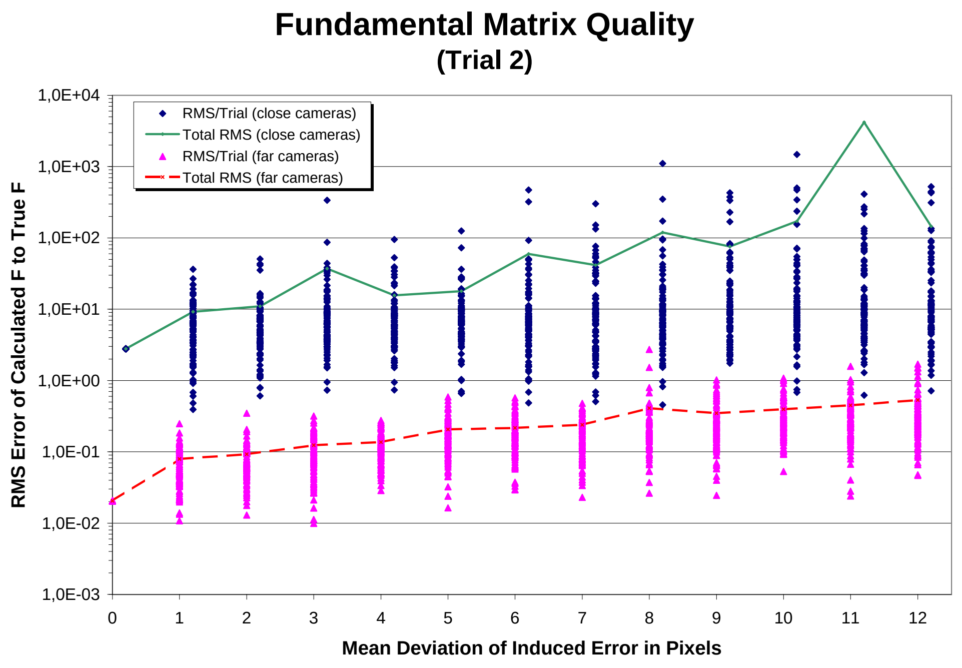


Figure 11: Comparison of fundamental matrix quality for second trial.

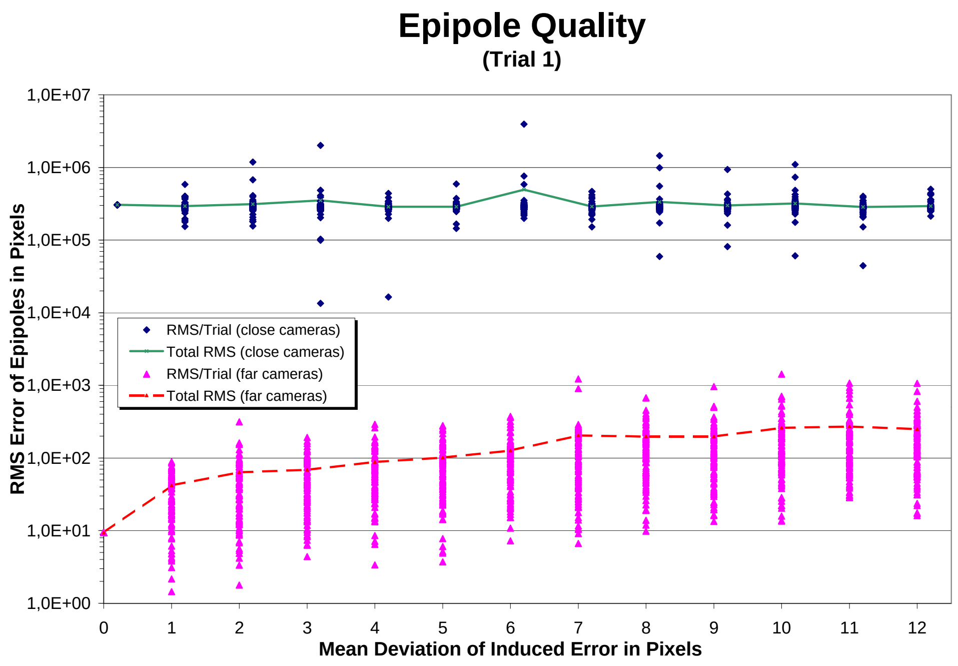


Figure 12: Comparison of epipole quality for first trial.

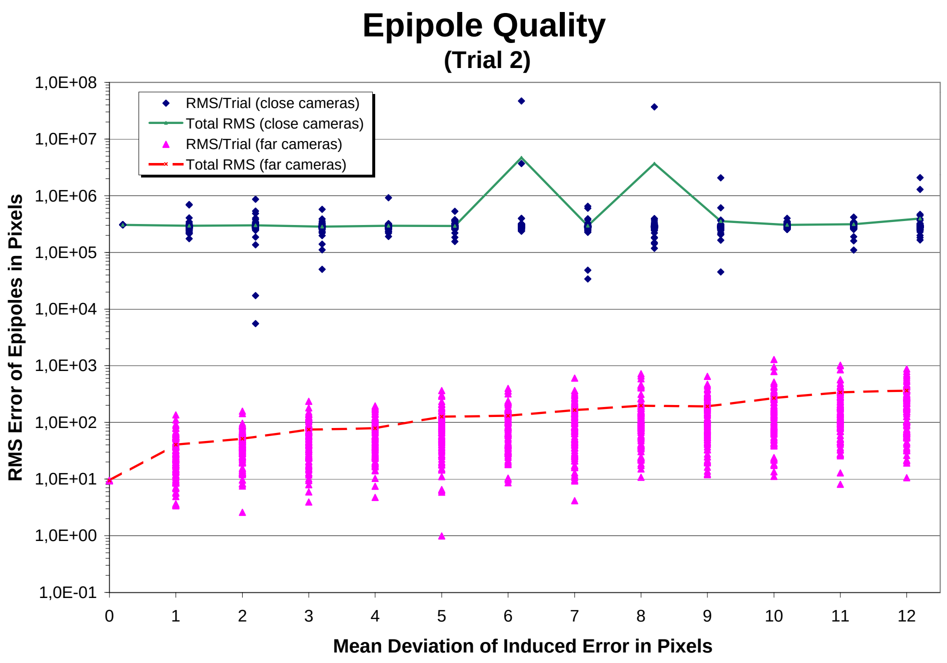


Figure 13: Comparison of epipole quality for second trial.

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. Since our calibration algorithm is very fast it seems feasible to employ maximum entropy methods. We will discuss this in future work.

Figures 10 and 11 show the quality of the fundamental matrices that were used for the respective reconstructions in figures 8 and 9. The quality measure is the root mean square of the difference between the true and the calculated fundamental matrix components.

The most obvious feature of the graphs is that the fundamental matrices are of much better quality for the far cameras setup. Still, comparing figures 10 and 11 with figures 8 and 9 we can also see that this large difference in quality is not translated directly into a quality difference between the reconstructions. A similar effect can be seen for the epipole quality.

Figures 12 and 13 show the quality of the epipoles corresponding to the reconstructions whose quality is shown in figures 8 and 9. The quality measure is the root mean square of the difference between the true and the calculated epipole coordinates.

The most obvious feature is again that the epipoles are much better for the far cameras setup, but they are still quite bad in an absolute sense. This seems to indicate that the error in estimation of epipoles is higher if the epipoles are further away from the image centre (i.e. have larger coordinates) as they are for the close cameras setup. Again the reconstruction quality does not seem to be influenced a lot by the errors in the epipoles.

In conclusion we can say that calculating an accurate F and thus accurate epipoles, is not very important for our reconstruction algorithm. The knowledge of parallel lines seems to make up for the bad quality of F and the epipoles in the minimisation. This is an interesting insight considering the amount of research that goes into calculating F, or the trifocal tensor, which is a related problem, as accurately as possible [26, 27, 35, 36, 23, 28, 29, 30, 31].

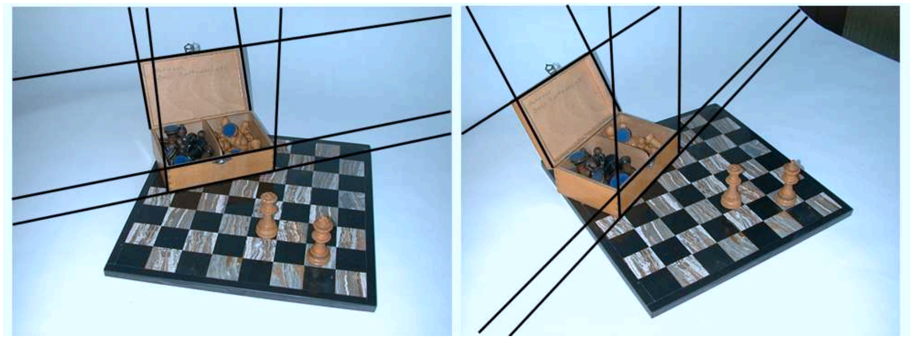


Figure 14: Initial images with parallel lines used for the calculation of the vanishing points and minimisation function indicated.

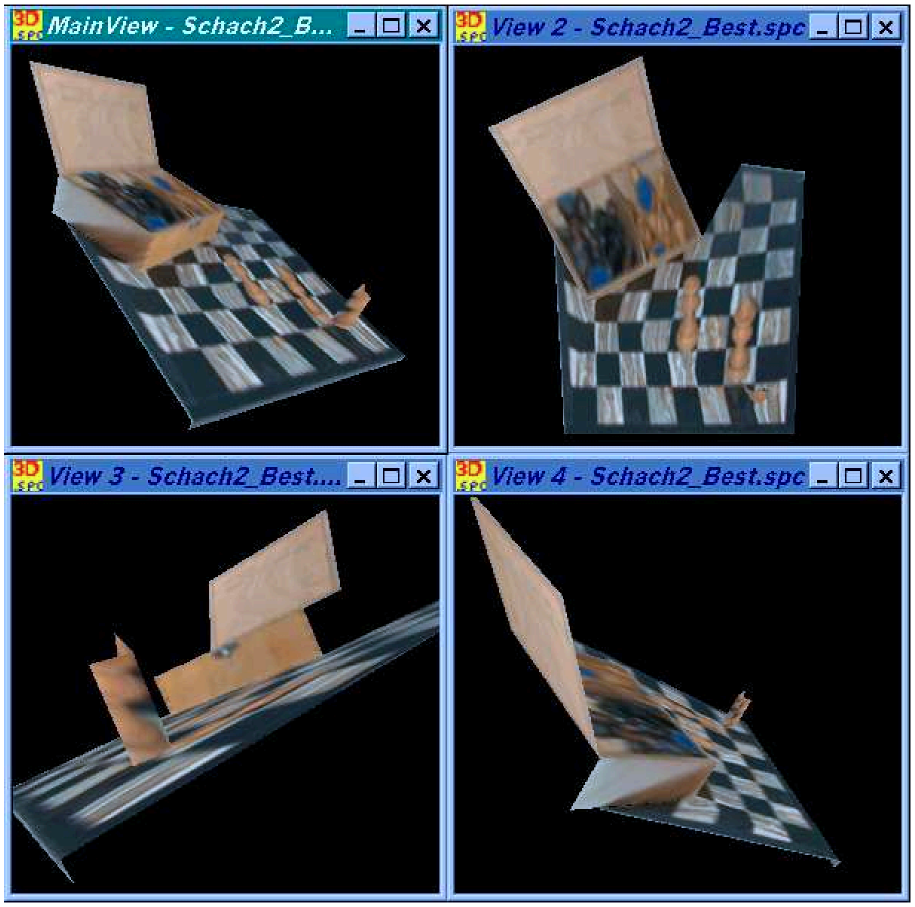

MainView - Schach2_B...
View 2 - Schach2_Best.spc
View 3 - Schach2_Best....
View 4 - Schach2_Best.spc

Figure 15: Reconstruction of the chessboard (Schachbrett).

7.2 Real Data

The real test for any reconstruction algorithm is the reconstruction of a real world scene. Figure 14 shows two views of a chessboard which we used for reconstruction[12]. The original images had a resolution of 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[13] can be seen in figure 15.

The different views of the reconstruction show that the chessbox was reconstructed quite well. However, the chessboard is not really square. Remember, though, that we have only used two line pairs and one line triplet to find three vanishing points. Furthermore, only two of the three vanishing points relate to orthogonal directions in $\mathcal{E}^3$. The reconstruction could 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, is not reconstructed particularly well. This is because it is very difficult to find matching point sets for round objects.

8 Conclusions

We have presented here an algorithm for the 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[14] find the internal calibration, rotation and translation of the second camera relative to the first one. Assuming that the first camera defines an orthonormal frame of $\mathcal{E}^3$ we can then create a 3D-reconstruction.

Disadvantages of our algorithm are that the internal calibration of the first camera is not found, and that the algorithm is not automatic. This is because apart from the point matches, combinations of vanishing points and parallel

[12]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.

[13]This and other reconstructions, as well as some more analysis of the reconstruction algorithm are demonstrated by the program **MVT**. You can download this program from C.Perwass' home page: `www.perwass.de`.

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

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.

The advantages of the algorithm are, first of all, that it is fast and robust. Also only a minimum amount of data has to be known. In particular, nothing has to be known about the camera positions or the cameras themselves.

Another interesting aspect of our algorithm is that there does not seem to be an easy way to derive its most important parts other than with GA. This is especially true for our expression of the P_∞-collineation (Ψ^∞) in terms of a camera matrix. The derivation of this equation was only possible because in GA we are working with the underlying basis and not only with the coordinates, as in matrix and tensor methods.

Faugeras discusses in [16] a 3D Euclidean interpretation of Ψ^∞. He finds that Ψ^∞ is proportional to a rotation matrix which "is transforming the directions of the axes of the first coordinate system into those of the second." This also follows directly from our expression. However, we also get the additional insight of how all this can be expressed in terms of a camera matrix, which forms the basis of our reconstruction algorithm.

Furthermore, although the derivation of M^∞ is somewhat tedious, it follows from a straightforward application of the intersection of lines and planes. Since GA gives intersection points directly as algebraic objects which can be further manipulated, all the necessary intersection calculations can be combined and reduced to a single tensor. This is certainly not easily possible with matrix methods.

We believe that apart from presenting a good reconstruction algorithm we have also shown that GA is a very useful tool which allows us to gain more geometric insight than with standard matrix methods, in an easy (for "GAers") and straightforward, since geometric, manner.

References

[1] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, pp. 133–135, 1981.

[2] J. Lasenby, W. J. Fitzgerald, A. Lasenby, and C. Doran, "New geometric methods for computer vision: An application to structure and motion estimation," *International Journal of Computer Vision*, vol. 3, no. 26, pp. 191–213, 1998.

[3] J. Lasenby and E. Bayro-Corrochano, “Computing Invariants in Computer Vision using Geometric Algebra,” Technical Report CUED/F - INFENG/TR. 224, Cambridge University Engineering Department, 1997.

[4] A. Heyden and G. Sparr, “Reconstruction from calibrated cameras - a new proof of the kruppa-demazure theorem,” *Journal of Mathematical Imaging and Vision*, vol. 10, pp. 1–20, 1999.

[5] O. D. Faugeras and S. Maybank, “Motion from point matches: Multiplicity of solutions,” *International Journal of Computer Vision*, vol. 4, pp. 225–246, 1990.

[6] N. Navab and O. Faugeras, “The critical sets of lines for camera displacement estimation: a mixed euclidean-projective and constructive approach,” Tech. Rep. 2305, INRIA, Sophia Antipolis, 1994.

[7] I. Shimshoni, R. Basri, and E. Rivlin, “A geometric interpretation of weak-perspective motion,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 3, pp. 252–257, 1999.

[8] B. Triggs, “Autocalibration and the absolute quadric,” in *IEEE Conf. Computer Vision & Pattern Recognition*, 1997.

[9] B. Triggs, “Autocalibration from planar scenes,” in *European Conference on Computer Vision*, 1998.

[10] B. Triggs, “Linear projective reconstruction from matching tensors,” *Image and Vision Computing*, vol. 15, pp. 617–625, 1997.

[11] C. Zeller and O. Faugeras, “Camera self-calibration from video sequences: the kruppa equations revisited,” Tech. Rep. 2793, INRIA, Sophia Antipolis, 1996.

[12] R. Enciso and T. Vieville, “Self-calibration from four views with possibly varying intrinsic parameters,” *Image and Vision Computing*, vol. 15, pp. 293–305, 1997.

[13] F. Devernay and O. Faugeras, “From projective to euclidean reconstruction,” Tech. Rep. 2725, INRIA, Sophia Antipolis, 1995.

[14] Q.-T. Luong and O. Faugeras, “Self-calibration of a stereo rig from unknown camera motions and point correspondences,” Tech. Rep. 2014, INRIA, Sophia Antipolis, 1993.

[15] E. Kruppa, “Zur Ermittlung eines Objektes aus zwei Perspektiven mit innerer Orientierung,” *Sitz.-Ber. Akad. Wiss., Wien, math. naturw. Kl. Abt. IIa.*, vol. 122, pp. 1939–1948, 1913.

[16] O. Faugeras, "Stratification of Three Dimensional Vision: Projective, Affine and Metric Representations," *Journal of the Optical Society of America - A*, vol. 12, no. 3, 1995.

[17] B. Caprile and V. Torre, "Using vanishing points for camera calibration," *International Journal of Computer Vision*, vol. 4, pp. 127–140, 1990.

[18] R. Cipolla and E. Boyer, "3d model acquisition from uncalibrated images," in *Proc. IAPR Workshop on Machine Vision Applications, Chiba, Japan*, pp. 559–568, 1998. To be published.

[19] C. Perwass, *Applications of Geometric Algebra in Computer Vision*. PhD thesis, Cambridge Universtiy, 2000.

[20] C. Perwass and J. Lasenby, "A Unified Description of Multiple View Geometry," in *Geometric Computing with Clifford Algebra* (G. Sommer, ed.), Springer-Verlag, 2000 to be published.

[21] D. Hestenes, *New Foundations for Classical Mechanics*. Dordrecht, 1986.

[22] D. Hestenes and G. Sobczyk, *Clifford Algebra to Geometric Calculus: A Unified Language for Mathematics and Physics*. Dordrecht, 1984.

[23] J. Lasenby and A. N. Lasenby, "Estimating Tensors for Matching over Multiple Views," *Phil. Trans. R. Soc. Lond. A*, vol. 356, no. 1740, pp. 1267–1282, 1998.

[24] S. F. Gull, A. N. Lasenby, and C. J. L. Doran, "Imaginary numbers are not real – the geometric algebra of space time," *Found. Phys.*, vol. 23, no. 9, p. 1175, 1993.

[25] L. Dorst, "Honing geometric algebra for its use in the computer sciences," in *Geometric Computing with Clifford Algebra* (G. Sommer, ed.), Springer-Verlag, 1999 to be publ.

[26] R. I. Hartley, "Lines and Points in Three Views and the Trifocal Tensor," *International Journal of Computer Vision*, pp. 125–140, 1997.

[27] R. I. Hartley, "In defence of the 8-point algorithm," *IEEE Int. Conf. Computer Vision*, pp. 1064–1070, 1995.

[28] C. Perwass and J. Lasenby, "A Geometric Analysis of the Trifocal Tensor," in *Image and Vision Computing New Zealand, IVCNZ'98, Proceedings* (R. K. R. Klette, G. Gimel'farb, ed.), pp. 157–162, The University of Auckland, 1998.

[29] O. Faugeras and B. Mourrain, “On the Geometry and Algebra of the Point and Line Correspondences between N Images,” Tech. Rep. 2665, INRIA, Sophia Antipolis, 1995.

[30] O. Faugeras and T. Papadopoulo, “A nonlinear method for estimating the projective geometry of three views,” Tech. Rep. 3221, INRIA, Sophia Antipolis, 1997.

[31] A. Heyden, “A Common Framework for Multiple View Tensors,” in *Computer Vision – ECCV'98* (H. Burkhardt and B. Neumann, eds.), no. 1406 in Lecture Notes in Computer Science, pp. 3–19, Springer, 1998.

[32] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in Pascal.* Cambridge University Press, 1990. ISBN 0-521-37516-9.

[33] D. MacKay, “MacOpt - a nippy wee optimizer.” http://wol.ra.phy.cam.ac.uk/mackay/c/macopt.html. Retrieved on 26 October 1999.

[34] R. I. Hartley, “Triangulation,” in *Computer Analysis of Images and Patterns* (R. Sara and V. Hlavac, eds.), no. 970 in Lecture Notes in Computer Science, pp. 190–197, Springer, 1995.

[35] R. I. Hartley, “Minimizing algebraic error,” in *Phil. Trans. R. Soc. Lond. A*, vol. 356, pp. 1175–1192, 1998.

[36] Q.-T. Luong, R. Deriche, O. Faugeras, and T. Papadopoulo, “On determining the fundamental matrix: Analysis of different methods and experimental results,” Tech. Rep. 1894, INRIA, Sophia Antipolis, 1993.