Trinocular stereovision for robotics

Nicholas Ayache, F. Lustman

INRIA

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

Rapports de Recherche

N° 1086

Programme 6
Robotique, Image et Vision

TRINOCULAR STEREOVISION FOR ROBOTICS

Nicholas AYACHE
Francis LUSTMAN

Septembre 1989

RR_1086

Trinocular Stereovision for Robotics :

Vision Stéréoscopique Trinoculaire pour la Robotique *

Nicholas Ayache Francis Lustman

July 1988

INRIA
BP 105
78153 Le Chesnay Cédex

Abstract

We present an original approach to build a three dimensional description of the environment of a robot using three cameras.

The main advantages of trinocular vs. binocular stereo are *simplicity, reliability and accuracy*. We believe that these advantages now makes trinocular stereovision of practical use for many robotics applications.

The technique has been successfully applied to several indoor and industrial scenes. Experimental results are presented and discussed.

Key-words: Trinocular Stereovision, 3D Maps, Edge Segments, Computer Vision, Mobile Robots.

Nous présentons une approche originale pour construire une description tridimensionnelle de l'environnement d'un robot en utilisant trois caméras.

Les avantages décisifs de la stéréovision trinoculaire sur la stéréovision binoculaire sont la simplicité, la fiabilité et la précision. *Nous pensons que ces avantages rendent maintenant possible l'utilisation effective de la vision stéréoscopique passive pour de nombreuses applications industrielles.*

Cette technique a été appliquée avec succès à des expériences impliquant diverses scènes d'intérieur, ainsi que des scènes industrielles. Ces expériences sont décrites et commentées.

Mots-Clés: *Stéréovision trinoculaire, Cartes visuelles 3D, Segments de contours, Vision par Ordinateur, Robots Mobiles.*

---

*this work was partially supported by esprit project P940.

1 Introduction

Stereovision is a technique to build a three dimensional description of a scene observed from several viewpoints. It is quoted as passive if no additional lighting of the scene, for instance by a laser beam is required. So defined, passive stereovision happens to be very attractive for many applications in robotics, including 3D object recognition and localization as well as 3D navigation of mobile robots.

Most of the research on passive stereovision has been devoted to binocular vision for which two cameras are observing the same scene from two slightly different viewpoints. As soon as two image points are matched, i.e. identified as corresponding to the same physical point, it is possible to compute the three dimensional coordinates of this physical point.

Unfortunately the matching problem is difficult. This is mainly because the *geometric* constraints of binocular stereo are not sufficient to impose a unique solution; several *heuristic* constraints must be added to compute a *plausible* matching solution.

Using a third camera increases the geometric constraints, and allows to reduce the influence of heuristics in stereo-matching. Presently, following Yachida [1,2] an increasing number of studies are devoted to trinocular vision. A review of some of these techniques can be found in [3] which includes most of the following publications [4,5] [6] [7] [8] [9] [10].

For a discussion of both geometric and heuristic constraints used in binocular stereovision and for a review of research on this topic, one can refer to [11] for instance. A non-exhaustive list of publications on the subject is given by the following references [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22]. Last but no least, the work of [23] on binocular stereovision pioneered the work on trinocular stereovision presented here.

The paper is organized as follows: first we explicit what is needed to constrain the stereo matching problem. This includes geometry of trinocular stereovision, representation of images, calibration, rectification, and spatial reconstruction. Then, we can detail the matching algorithm and the validation procedure. Finally experimental results are presented and discussed. We conclude by a summary and future research.

2 Geometry of Trinocular Stereovision

Figure 1 illustrates the geometric constraints of trinocular stereovision. Camera i ($i = 1$, 2 or 3) is represented by its optical center C_i and its image plane $\mathcal{P}_i$. Given a scene point P, its image I_i by camera i is given by the intersection of the line PC_i with the plane $\mathcal{P}_i$. This is the classical pinhole model. Points I_1, I_2 et I_3 form a triplet of *homologous* image points.

Given a pair (i, j) of cameras and a physical point P, the *epipolar plane* $\mathcal{Q}_{ij}$ is defined by the triplet of points (C_i, P, C_j). The intersection of this epipolar plane with camera plane $\mathcal{P}_i$ is the

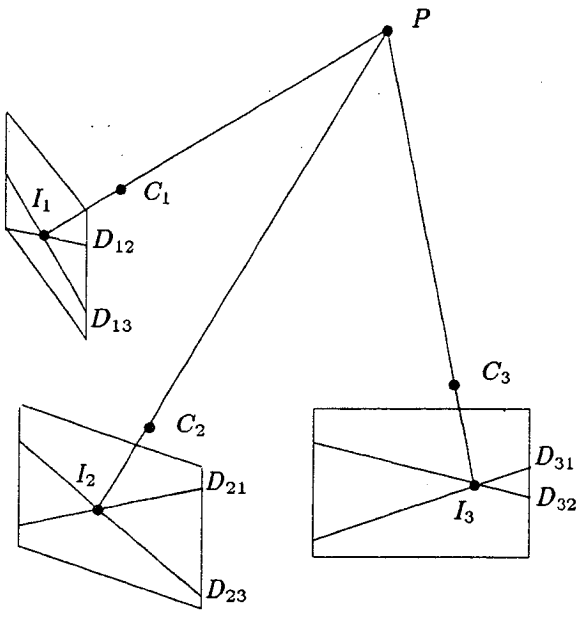

Figure 1: Geometric constraints of trinocular stereovision

epipolar line D_{ij}, while its intersection with camera plane $\mathcal{P}_j$ is the epipolar line D_{ji}. D_{ij} and D_{ji} are called *conjugated epipolar lines.* Any point I_i on D_{ij} (resp. I_j on D_{ji}) has its homologous image point I_j on D_{ji} (resp. I_i on D_{ij}). Therefore, using two cameras, the search for homologous image points is a search along conjugated epipolar lines.

As one can see on figure 1, a scene point P produces three pairs of homologous epipolar lines. When the image points (I_i, I_j, I_k) form a triplet of *homologous* image points, then I_i is necessarily located at the intersection of the epipolar lines D_{ij} and D_{ik} respectively defined by I_j and I_k. Therefore the search for homologous image points between two images can now be reduced to a simple verification at a precise location in the third image. For instance checking that (I_1,I_2) form a pair of homologous image points consists in verifying the presence of I_3 at the intersection of D_{31} and D_{32}.

3 Image representation

The matching algorithm does not perform directly on the image, but on a symbolic representation of it.

For a number of reasons, we have come to use linear edge segments :

- physical meaning and reliability (cf. figure 2) : most of the edges come from physical phenomena such as changes in reflectance (type 3), changes in illumination (type 4), and changes

of the surface normal (type 1 and 2). Except in the case where the observed surface recedes away smoothly (type 1), in which case the detected edges in the two images may not be exactly the image of the same part of the object, the edges provide a good and reliable source of information.

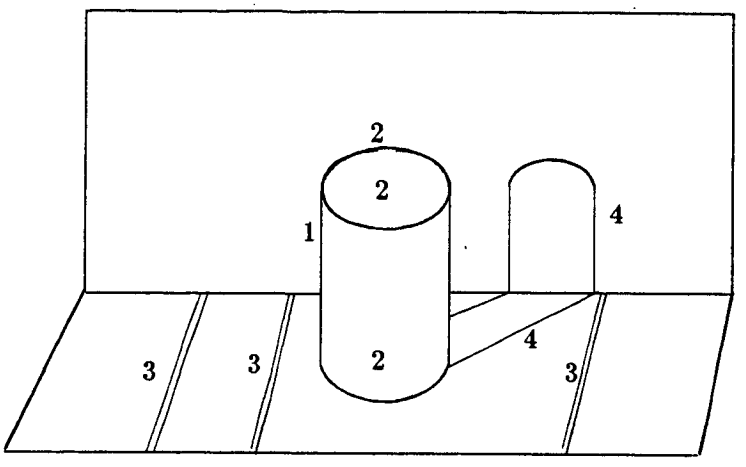

Figure 2: Different types of edges

- compactness : the information contained in the edges is not only very significant but also much more compact in terms of storage and matching computational burden.
- richness of attributes : many useful features can be attached to the edge segments to help solving the stereo matching process. These features can be, geometric (length, angle), intensity-based (average contrast along the segment, average intensity of the neighbouring regions) or structure-based (edge chains, neighborhoods).
- density : this representation is structured but nevertheless rather dense over the image, therefore enabling to keep enough information over the whole image.
- accuracy : our purpose in performing stereo matching is to be able to reconstruct the $3D$ environment accurately. Edges can be reliably and accurately extracted and, as we shall see, the least-squares approximation used during polygonal approximation enables us to get a sub-pixel accuracy.
- easiness : there are a number of ways to easily and reliably extract edges from an image.

3.1 Edge extraction

Edge points are first computed using a very efficient recursive filter developed by Deriche[24] after Canny's ideas[25]. The edge pixels thus obtained are structured into edge chains using a program developped by Giraudon [26], in turn approximated by line segments using a program developped by Berthod. The interested reader will find in [27] a good review of some techniques for polygonal approximation.

Let us note that, to get a better accuracy in $3D$ reconstruction, our ultimate goal, we perform a least-square approximation to fit a 2D linear segment betwen each pair of successive breakpoints. This allows us to obtain sub-pixel accuracy.

3.2 Features

For each of the segments, a number of features are computed. Among all possible ones, we use the following set :

- length.
- angle[1]
- average gradient magnitude along the segment.

3.3 Buckets

As we show later, the stereo-matching algorithms often requires to access segments lying in a given region of the image. We therefore need to structure the image to optimize this operation. A very simple and efficient way to proceed is to compute buckets, i.e. superimpose a virtual grid composed of square windows on the image and compute, for each window, the list of segments intersecting it. Accessing a segment in a given region of the image is then reduced to accessing the segments of the buckets covering this area of the image. This structure is computed in linear time with respect to the number of segments. Figure 3 shows the principle of the method.

Furthermore, the buckets allow us to define a neighborhood structure. The neighborhoods are defined by the buckets : two segments are neighbors if and only if they share a common bucket. To obtain better neighborhoods, one can superimpose two sets of partially overlapping buckets, as it is shown in Figure 4.

Typically, we used 16x16 buckets for the matching phase, and 8x8 buckets for the validation, because we want to get enough neighbours.

[1] we use segments oriented by the gradient, i.e. the orientation is computed modulo 2π.

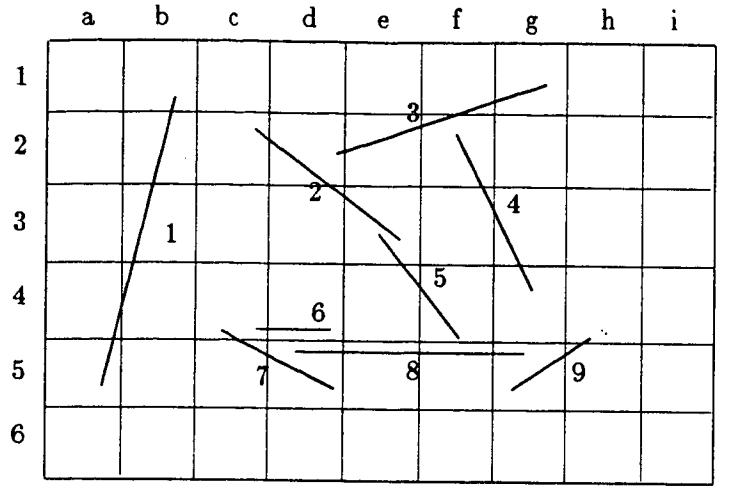

Figure 3: Computation of buckets and neighbourhoods :

Neighborhoods: V(1)=Nil; V(2)={3,5}; V(3)={2,4}; V(4)={3}; V(5)={2}; V(6)={7}; V(8)={7,9}; V(9)={8};

Lists of connected neigbours : {1}; {2,3,4,5}; {6,7,8,9}

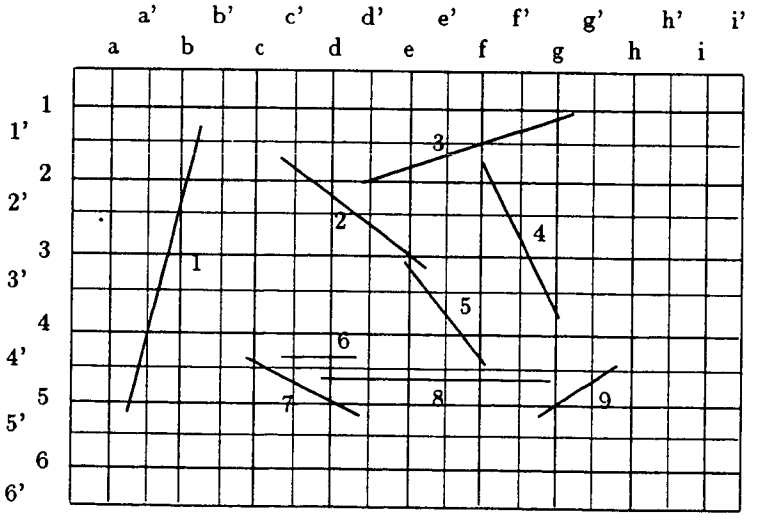

Figure 4: Better definition of neighbourhoods, using two sets of overlapping buckets :

Neighborhoods: V(1)=Nil; V(2)={3,5}; V(3)={2,4}; V(4)={3}; V(5)={2,8}; V(6)={7,8}; V(8)={5,6,7,9}; V(9)={8};

Lists of connected neigbours : {1}; {2,3,4,5,6,7,8,9}

4 Calibration

4.1 Image Modelling

Let us choose one of the cameras, characterized by its optical center C and its image plane $\mathcal{P}$, and let us model the image formation process. A point P in the observed scene is projected on point I of the camera retina. The relationship between P and I is modelled as a linear transformation in projective coordinates. If we denote $I^* = (U, V, S)^t$ the projective coordinates of I and $(x, y, z)^t$ the coordinates of P, the following relation holds:

$$I^* = \begin{pmatrix} U \\ V \\ S \end{pmatrix} = T \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

where T is a 3×4 matrix usually called the *perspective matrix* of the considered camera.

If P is in the focal plane of the camera, (i.e. if the straight line CP is parallel to the image plane $\mathcal{P}$), then $S = 0$ and the coordinates $(u, v)^t$ of I are no longer defined. In the general case $S \neq 0$ and the image coordinates of I (usually expressed in *pixels*) are given by:

$$I = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} U/S \\ V/S \end{pmatrix}$$

4.2 Determining the perspective matrix T

In the experiments conducted in our laboratory [28,29], T is obtained by analyzing a calibration pattern which is a grid painted on a planar surface. The 3D position of the intersection points of the grid are well known in an absolute 3D coordinate frame and the grid is observed from several well defined different positions.

T is a matrix of dimension 3×4, but it is defined up to a scale factor, and one needs a constraint to specify T uniquely. The simplest constraint[2] consists in assuming that $t_{34} \neq 0$, then enforcing

$$t_{34} = 1$$

Each time an image point $I = (u, v)^t$ is matched with its corresponding scene point $P = (x, y, z)^t$, this provides the following two linear equations on the eleven unknowns remaining for determining T:

$$\begin{aligned} P^t t_1 + t_{14} - u(P^t t_3 + 1) &= 0 \\ P^t t_2 + t_{24} - v(P^t t_3 + 1) &= 0 \end{aligned} \quad (1)$$

[2] on the discussion of this constraint, see [28,29].

where t_{jk} is the element of rank (j,k) in T, and t_j is the 3-vector obtained from the first 3 elements of the j^{th} row of T:

$$t_j = (t_{j1}, t_{j2}, t_{j3})^t$$

In theory, six non coplanar points are sufficient for determining T uniquely [30]. In practice, several dozens of points are available, allowing for a global or recursive least squares estimation of T.

5 Computing epipolar constraints

We now assume that we are dealing with at least two cameras, and we compute the epipolar constraints between them. First, we compute from each matrix T_i the optical center of the cameras, then the inverse image of an arbitrary image point.

5.1 Determining optical centers

The 3D coordinates $(x_{C_i}, y_{C_i}, z_{C_i})$ of the optical center C_i of camera i (modelled by the perspective matrix T_i) are obtained by solving:

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = T_i \begin{pmatrix} x_{C_i} \\ y_{C_i} \\ z_{C_i} \\ 1 \end{pmatrix}$$

which is a system of three linear equations in the three unknowns $(x_{C_i}, y_{C_i}, z_{C_i})$.

5.2 Computing inverse images

We need to compute the straight line D which is the inverse image in the scene of a given image point I. This straight line D is composed of 3D scene points P having the same image I. If we look at figure 1 we see that D is simply the straight line defined by I and C_i.

To determine D analytically, let us re-write the system of equations 1 which relates point I to points P in the form:

$$\begin{aligned} (t_1^i - u_1 t_3^i)^t P + t_{14}^i - u_1 t_{34}^i &= 0 \\ (t_2^i - v_1 t_3^i)^t P + t_{24}^i - v_1 t_{34}^i &= 0 \end{aligned}$$

where the i index in t_j^i refers to camera i.

These are equations of two planes whose intersection defines D. A vector n colinear to D is the cross-product of the normals to the planes:

$$n = (t_1^i - u_1 t_3^i) \times (t_2^i - v_1 t_3^i)$$

which yields:

$$n = u_1\, t_2^i \times t_3^i + v_1\, t_3^i \times t_1^i + t_1^i \times t_2^i$$

which can be written

$$n = N_i I_i^* \tag{2}$$

with

$$N_i = [t_2^i \times t_3^i \quad t_3^i \times t_1^i \quad t_1^i \times t_2^i]$$

The parametric equation of the line $C_i I$ is therefore given by

$$P = C_i + \lambda\, n$$

where n is given by the previous equation and where λ is a real number.

5.3 Parametric equation of epipolar lines

It is now easy to compute the parametric equation of the epipolar line D_{ji} in image j corresponding to the image point I_i of coordinates (u_i, v_i) in image i, because D_{ji} is simply the image of the line $C_i I_i$ by camera j. Therefore D_{ji} is composed of points I_j whose projective coordinates satisfy:

$$I_j^* = T_j \begin{pmatrix} C_i + \lambda\, n \\ 1 \end{pmatrix}$$

If we denote

$$F_j^* = T_j'\, n \tag{3}$$

where T_j' is the 3x3 sub-matrix obtained from T_j by suppressing its last column, and

$$E_j^* = T_j \begin{pmatrix} C_i \\ 1 \end{pmatrix} \tag{4}$$

then we get the parametric equation of the epipolar line D_{ji} *in projective coordinates:*

$$I_j^* = E_j^* + \lambda\, F_j^*$$

Therefore, the parametric equation of D_{ji} *in image coordinates* is:

$$u_j = \frac{U_{E_j} + \lambda\, U_{F_j}}{S_{E_j} + \lambda\, S_{F_j}} \tag{5}$$

$$v_j = \frac{V_{E_j} + \lambda\, V_{F_j}}{S_{E_j} + \lambda\, S_{F_j}} \tag{6}$$

From these equations, one can see that the epipolar lines form a pencil of lines going through an epipolar center E_j which is the image of C_i in camera j. Also, F_j is the *vanishing point* corresponding

to P at an infinite distance from the cameras ($\lambda \to \infty$). One can also notice that a vector colinear to the epipolar line D_{ji} is obtained by differentiation of equations 5 and 6 with respect to λ. This yields

$$\begin{pmatrix} \Delta u_j \\ \Delta v_j \end{pmatrix} = \begin{pmatrix} U_{F_j} S_{E_j} - U_{E_j} S_{F_j} \\ V_{F_j} S_{E_j} - V_{E_j} S_{F_j} \end{pmatrix} \tag{7}$$

When $S_{E_j} = 0$, this means that the epipolar center E_j is rejected to infinity. In this case, the direction of the epipolar lines becomes independent of the coordinates (u_i, v_i) of I_i, and one can see from equation 7 that in this case all epipolar lines are parallel to the vector:

$$\begin{pmatrix} \Delta u_j \\ \Delta v_j \end{pmatrix} = \begin{pmatrix} U_{E_j} \\ V_{E_j} \end{pmatrix}$$

5.4 Computation of epipolar intervals

In practice, the homologuous I_j of I_i is constrained to belong to an *interval* of the epipolar line D_{ji}. This comes from the fact that the physical point P has to belong to only the portion of D which is *in front* of the optical center C_i. If the vector n is properly oriented on D, this constraint is equivalent to $\lambda > 0$, and produces in general[3] an interval of the form

$$I_j \in [F_j, E_j]$$

where F_j and E_j are respectively the vanishing point and the epipole previously defined (cf. Figure 5)

Actually, this epipolar interval is even reduced by the knowledge of a minimum and a maximum possible distance of the observed points from the camera. If we denote $n' = n/\|n\|$ and if we call λ_m and λ_M these minimum and maximum allowed distances respectively, one sees that the epipolar interval is given by $[I_m, I_M]$ where I_m and I_M are respectively the images by camera j of the physical points

$$P_m = C_i + \lambda_m n'$$

and

$$P_M = C_i + \lambda_M n'$$

Therefore, given a point $I_i = (u_i, v_i)^t$ and to obtain the epipolar interval $[I_m, I_M]$, the following

[3] when the focal plane of camera j intersects D at a point Q behind the optical center, i.e. such that $Q = C_i + \lambda n$ with $\lambda < 0$, see Figure 5.

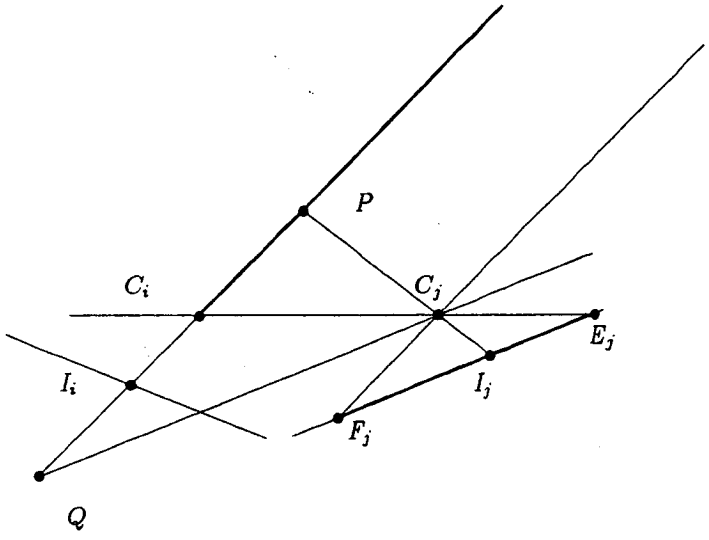

Figure 5: Computation of an epipolar interval: image point I_i can be matched only with points I_j lying on the epipolar interval $F_j E_j$

operations must be performed:

1. $n = N_i I_i^*$,
2. $n' = n/||n||$,
3. $I_m = E_j^* + \lambda_m n'$,
4. $I_M = E_j^* + \lambda_M n'$.

To be complete, this interval must then be clipped by the window corresponding to the actually observed image.

6 Rectification of images

6.1 Principle

For three cameras, it is possible to rectify the images to get horizontal epipolar lines between images 1 and 2, and vertical epipolar lines between images 1 and 3. In this case, the previous computations of epipolar segments are greatly simplified . If, in addition, the image coordinate frames are judiciously defined it is possible that the epipolar line attached to a point (u'_1, v'_1) in

image 1 be the line $v'_2 = v'_1$ in image 2 and the line $u'_3 = u'_1$ in image 3. Moreover, it is possible to obtain a very simple relationship between images 2 and 3 of the form $u'_2 = v'_3$. We are then in the situation depicted by figure 6.

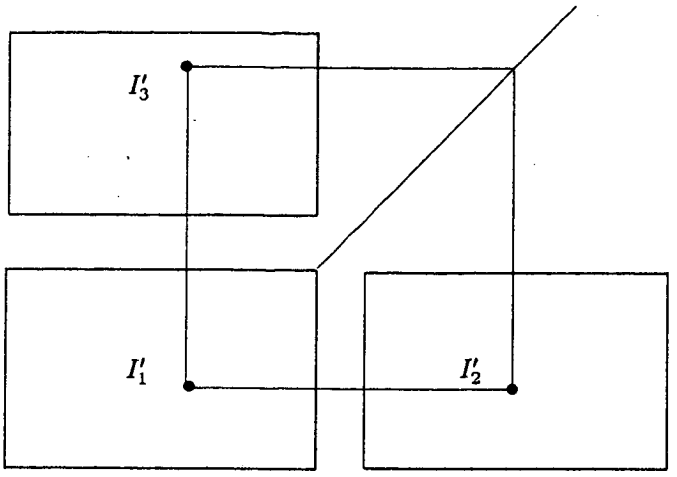

Figure 6: After the rectification of three images : the coordinates of the homologous points I'_1, I'_2 and I'_3 satisfy $v'_2 = v'_1$, $u'_3 = u'_1$ and $v'_3 = u'_2$

One can show [31,32,33] that rectification can be performed by linear transformations of the image coordinates in projective space by:

$$I'^*_i = R_i I^*_i$$

where the three 3x3 rectification matrices called R_1, R_2 and R_3 are defined by

$$R_i = \begin{pmatrix} (C_{i-1} \times C_i)^t \\ (C_i \times C_{i+1})^t \\ (C_1 \times C_2 + C_2 \times C_3 + C_3 \times C_1)^t \end{pmatrix} N_i$$

whith the conventions $i + 1 = 1$ if $i = 3$ and $i - 1 = 3$ if $i = 1$.

After the rectification of the images we have, as desired, the nice relationships:

$$\begin{aligned} v'_2 &= v'_1 \\ u'_3 &= u'_1 \\ v'_3 &= u'_2 \end{aligned} \tag{8}$$

which was illustrated by figure 6.

6.2 Algorithmic Complexity

The rectification of l images ($l = 2$ or 3) requires the storage of l 3x3 matrices, i.e. $9\,l$ parameters. Then it requires 6 multiplications, 6 additions and 2 divisions per rectified point.

As the rectification process is a linear transformation in projective space, it preserves straight lines: therefore it is sufficient to apply it to the endpoints of the linear segments of a polygonal approximation to get the endpoints of the segments of the rectified polygonal approximation. This is very useful for our stereovision algorithms [23,34,35] which actually deal with linear segments.

6.3 Example

We show in figure 7 an example of a rectified triplet of an office scene.

7 Building 3D Segments

To build a 3D map from trinocular stereovision matches, one must

1. build a 3D line whose 2D projections are known in several images,
2. determine the endpoints of a 3D segment on the computed 3D line.

These two problems are solved in turn in the following subsections.

7.1 Building 3D lines from their 2D images

The problem is to build a 3D line whose 2D projections are known in several images. More formally, given three 2D lines d_i, one seeks the 3D line D whose projections d'_i on cameras i ($i = 1,2,3$) *best* approximate the 2D lines d_i (cf. figure 8).

For doing this, one uses minimal representation of lines. Therefore, assuming d_i is not parallel to the v axis, [4] it is represented by the parameters (α_i, μ_i) such that the equation of d_i in the image plane of camera i is

$$\alpha_i u_i + v_i + \mu_i = 0$$

Assuming that D is not perpendicular to the z axis, [5] it is represented by the parameters (a, b, p, q) such that D is defined by the equations

$$\begin{cases} x = az + p \\ y = bz + q \end{cases} \tag{9}$$

[4] one uses the symmetric parametrization for lines parallel to the v axis

[5] one uses two complementary parametrization respectively for lines perpendicular to the zx or zy planes.

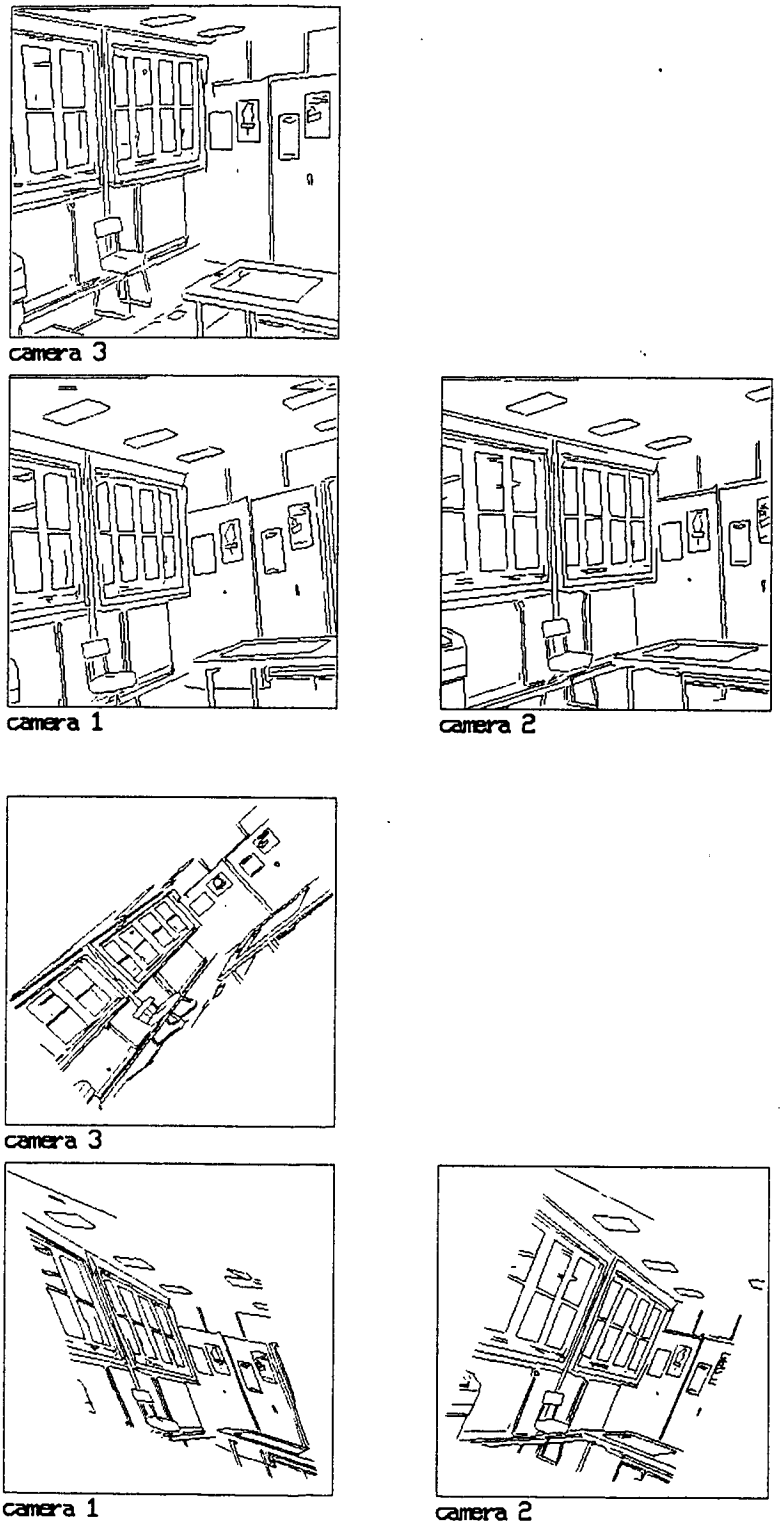

Figure 7: Triplet of linear segments of contours before and after rectification

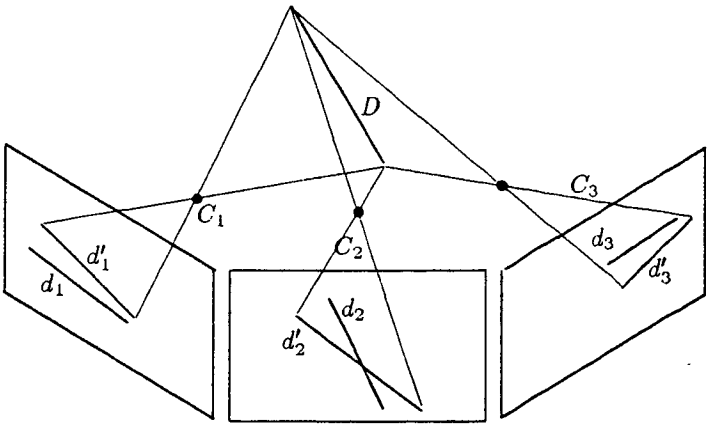

Figure 8: Building 3D lines from their 2D images

One assumes that the perspective transformation of each camera is represented by a 3x4 matrix T_i computed during a preliminary calibration stage [29]. If we denote by t^i_{jk} the element of rank (j,k) in the perspective matrix T_i, saying that the projection of D on camera i is d_i is equivalent to saying that the following two equations hold (see appendix):

$$a(\alpha_i t^i_{11} + t^i_{21} + \mu_i t^i_{31}) + b(\alpha_i t^i_{12} + t^i_{22} + \mu_i t^i_{32}) + (\alpha_i t^i_{13} + t^i_{23} + \mu_i t^i_{33}) = 0 \quad (10)$$

$$p(\alpha_i t^i_{11} + t^i_{21} + \mu_i t^i_{31}) + q(\alpha_i t^i_{12} + t^i_{22} + \mu_i t^i_{32}) + (\alpha_i t^i_{14} + t^i_{24} + \mu_i t^i_{34}) = 0 \quad (11)$$

This system provides two independant linear equations on the unknowns (a,b) and (p,q) respectively: therefore two images are enough to solve for (a,b,p,q) exactly. Given three images, the system becomes overconstrained, and one must define an error criterion.

To do so, we consider the uncertainties on the parameters of the 2D lines, and we take them into account explicitely by computing a recursive weighted least square solution (Kalman Filter approach). This approach provides not only a better estimate of (a,b,p,q) (compared to a simpler least-square) but also an estimate of its quality under the form of a 4x4 symetric covariance matrix W_D. The interested reader is referred to [32,33,36].

7.2 Computing 3D endpoints

Having computed the parameters of a supporting 3D line, one must use the endpoints of the 2D image segments to define the enpoints of a 3D segment. For each endpoint I_i of a 2D segment in image i, we compute the 3D line D_i supported by $C_i I_i$ and the 3D point P_i of D which is closest to D_i.

Therefore, given the two endpoints a_i and b_i of a 2D segment, one obtains the enpoints A_i and B_i of a 3D segment *supported by* D. This is illustrated by figure 9.

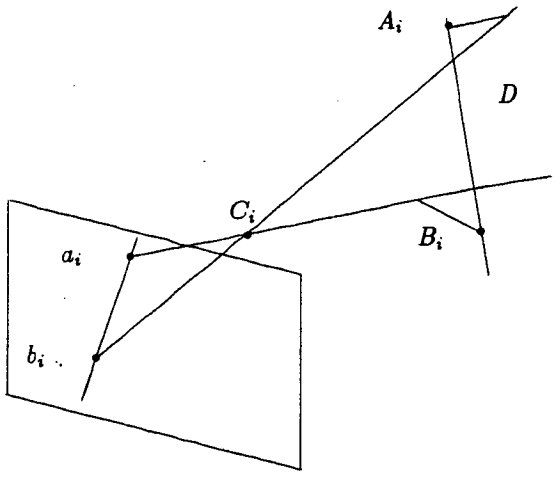

Figure 9: Building 3D segments from 2D segments

This operation is repeated for the enpoints of the corresponding segment in images j and k. Because of segmentation errors, the endpoints computed from different images do not match, which means that each of the three image segments correspond to a slightly different part of the 3D segment.

We decide to keep the 3D segment on D which is the intersection of A_iB_i, A_jB_j and A_kB_k. Thus, we reconstruct the interval on D which is seen simultaneously by the three cameras. This solution does not prevent us from reconstructing the remaining parts of a 3D segment using another triplet of matches, as illustrated by figure 10.a. Another advantage of this solution will be explained later with the validation procedure.

Finally, to compute in image i which part of the original image segment corresponds to the reconstructed 3D segment, one projects the endpoints of the reconstructed 3D segment on image i, and then on the 2D line supporting the initial image segment.

8 Matching

We present first a simplified algorithm which gives the flavor of the matching procedure. Then we provide a detailed presentation of the actually implemented procedure, which takes advantage of a number of refinements

8.1 Simplified Algorithm

The matching algorithm takes as input three sets of linear segments $\{S_1\}, \{S_2\}, \{S_3\}$ coming from images 1, 2 and 3 respectively, and builds as output a set of triplets of matched segments

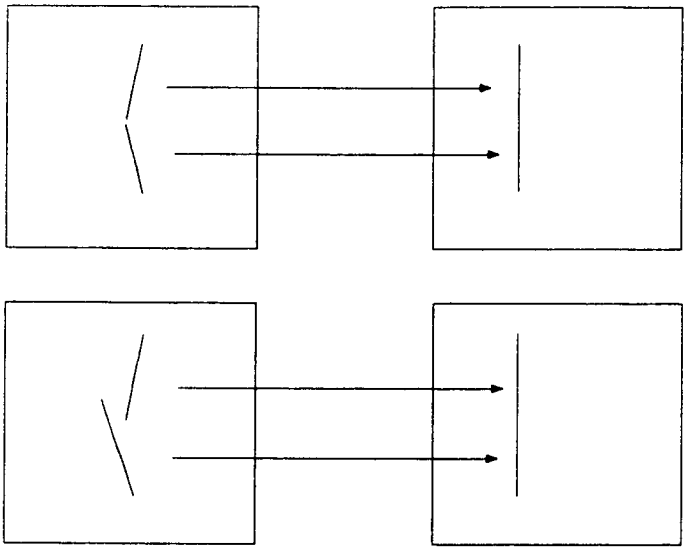

Figure 10: Multiple matches

$\{(S_1, S_2, S_3)\}$.

The initial scheme of the algorithm is the following (cf. Fig. 11):

- For each segment S_1 of image 1, compute the intervals of epipolar lines D_{21} and D_{31} in images 2 and 3 attached to the midpoint I_1 of S_1 and corresponding to a tolerated interval of distances $[\lambda_m, \lambda_M]$ (cf. calibration section).
- For each segment S_2 in image 2 intersecting D_{21} in I_2, compute the epipolar line D_{32} in image 3 attached to I_2. Let $I_3 = D_{31} \cap D_{32}$, and predict the orientation ϕ_3^* of S_3 in image 3 from the orientations ϕ_1 of S_1 and ϕ_2 of S_2.
- For each segment S_3 in image 3 of orientation ϕ_3, if the distance $\delta(I_3, S_3) < \varepsilon_\delta$ and if $\phi_3^* - \phi_3 < \varepsilon_\phi$, then form the triplet (S_1, S_2, S_3).

8.2 Ideas for Refinements

The previous algorithm can be improved by adding the following refinements :

- First, we could choose as image 2 the image for which the orientation of the corresponding epipolar line D_{12} in image 1 is the farther from the orientation of segment S_1. Doing so, we should avoid the search for the intersection of parallel lines in image 2, and also optimize the localization of the intersection point I_2.
- Second, we should take into account the structure of buckets previously computed to speed up the selection process of S_2 and S_3.

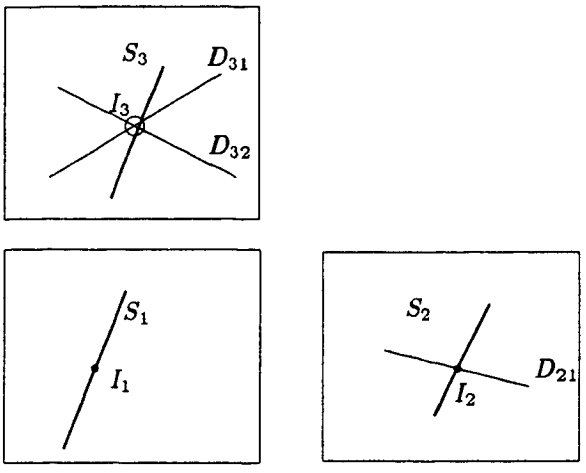

Figure 11: Principle of trinocular stereovision algorithm

- Finally, due to potential unfortunate coincidences, a validation procedure sould be applied at the end.

A formal description of the major procedures involved to implement this refinements is now provided.

8.3 Main Procedure

We give in figure 12 the description of the main matching procedure called STEREO-3. This procedure takes a triplet of images $(1, 2, 3)$ as input, and builds a list of matches $\{(S_1, S_2, S_3)\}$.

To do this, it selects every segment S_1 of the first image and computes the epipolar intervals D_{21} and D_{31} of the midpoint I_1 of S_1 in the second and third images. These intervals correspond to the possible positions of the points homologous to I_1. It also computes the orientations α_{12} and α_{13} of the conjugated epipolar lines in image 1.

During the matching procedure, one of the images 2 and 3 is used to make initial hypotheses and the other to check them. As explained before, the image used to do the initial hypotheses must be such that it is the one in which the orientation of the corresponding angle α_{12} or α_{13} is farthest from the direction of S_1. The matching procedure MATCH-3 is described in figure 13 and returns a list of matched segments $\{(S_1, S_2, S_3)\}$ associated to S_1.

The last part of the algorithm is the validation part, which makes a compatibility test with the neighbours of each matched segment. It is described in figure 17, and detailed in the next section.

⊙⊙⊙ **Procedure** STEREO-3 (1,2,3)

For each segment S_1 of image 1

- determine the epipolar intervals D_{21} (resp. D_{31}) corresponding, in image 2 (resp. 3) to the midpoint I_1 of S_1 for the allowed disparity interval.
- Compute the angle between S_1 and the epipolar lines D_{12} et D_{13} in image 1:
  - α_{12} = ANGLE (S_1, D_{12});
  - α_{13} = ANGLE (S_1, D_{31})
- **If** $|\alpha_{12}| > |\alpha_{13}|$
  - **then** $\{(S_1, S_2, S_3)\} \leftarrow$ MATCH-3 $(1, 2, 3, S_1, D_{21}, D_{31})$
  - **else** $\{(S_1, S_2, S_3)\} \leftarrow$ MATCH-3 $(1, 3, 2, S_1, D_{31}, D_{21})$

EndFor

For each of the matched triplets (S_1, S_2, S_3)

- VALIDATE-3(S_1, S_2, S_3);
  enforce the continuity constraint by using neighbours

EndFor

EndProcedure STEREO-3 ⊙⊙⊙

Figure 12: Trinocular Stereovision algorithm

8.4 Matching procedure

The algorithm of procedure MATCH-3 is given in figure 13. This procedure takes as input three images i, j and k, a segment S_i of image i and two epipolar segments D_{ji} and D_{ki}, corresponding to the midpoint I_i of S_i. It returns a list of matched triplets $\{(S_i, S_j, S_k)\}$ associated to the input segment S_i.

The first part of the algorithm is the search, in image j, for potential matches S_j to S_i. Segments S_j must

- go through the similarity tests with S_i
- intersect the epipolar segment D_{ji}. As we have taken j such that the orientation of D_{ij} be as far as possible to the orientation of S_i, the intersection I_j between D_{ji} and S_j can in general be accurately computed.

The second part of the algorithm checks the validity of each of the potential matches (S_i, S_j) in image k. We use a procedure of prediction of the position and orientation of the segment S_k in image k.

- the prediction of the position is made by computing the intersection I_k, in image k, of the epipolar lines respectively associated to the midpoint of S_i and its potential homologous point I_j in image j (cf. figure 14 in which we show the computation of this predicted point for two candidate segments S_j and S'_j).
- the prediction of the orientation is done in the following manner (see figure 15 for the notations):
  - Compute the epipolar lines D'_{ji} and D'_{ki} respectively associated, in images j and k, to one of the extremities I'_i of S_i.
  - Compute the intersection point I'_j, in image j, between the epipolar line D'_{ji} and the line D_{S_j} bearing S_j:
$$I'_j = D'_{ji} \cap D_{S_j}$$
  - Compute, in image k , the epipolar line D'_{kj} associated to I'_j.
  - Compute the intersection I'_k, in image k, between the epipolar line D'_{ki} and D'_{kj}.
$$I'_k = D'_{ki} \cap D'_{kj}$$
  - The predicted orientation of S_k id then given by the orientation of $I_k I'_k$:
$$\phi^*_k = \phi(I_k I'_k)$$

△ **Procedure** MATCH-3 $(i, j, k, S_i, D_{ji}, D_{ki})$

- Compute the list $\{\mathcal{B}_j\}$ of buckets intersected, in image j, by the epipolar interval D_{ji}.

  For each intersected bucket $\mathcal{B}_j$, determine the set of segments $\{S_j\}$ verifying the three following tests :

  1. $|\phi_j - \phi_i| < \theta_{Max}$ /*Compare segments orientations*/
  2. $\Delta l_{Min} < |l_j/l_i| < \Delta l_{Max}$ /*Compare segments lengths*/
  3. $I_j = (S_j \cap D_{ji}) \neq \emptyset$ /*Verify epipolar constraints*/

  EndFor

  For each of the segments S_j having passed these tests,

  - Compute, in image k, the epipolar interval D_{kj} associated to I_j.
  - Compute, in image k, the intersection point of the two epipolar intervals $I_k = D_{ki} \cap D_{kj}$ and the bucket $\mathcal{B}_k$ associated to I_k.
  - Predict the orientation ϕ_k^* and the length l_k^* of S_k.

    For each of the segments S_k in bucket $\mathcal{B}_k$, select those passing the three following tests:

    1. $|\phi_k - \phi_k^*| < \theta_{Max}^*$ /*Check predicted orientation*/
    2. $\Delta l_{Min} < |l_k/l_k^*| < \Delta l_{Max}$ /*Check predicted length*/
    3. $d(I_k, S_k) < d_{Max}$ /*Check predicted position*/

    Endfor

  Endfor

- **Return**$(\{(S_i, S_j, S_k)\})$ the list of matching triplets.

EndProcedure MATCH-3 △

Figure 13: Matching algorithm

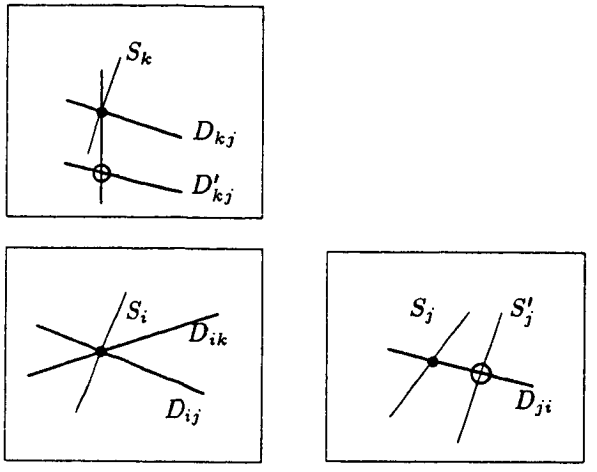

Figure 14: Prediction of position

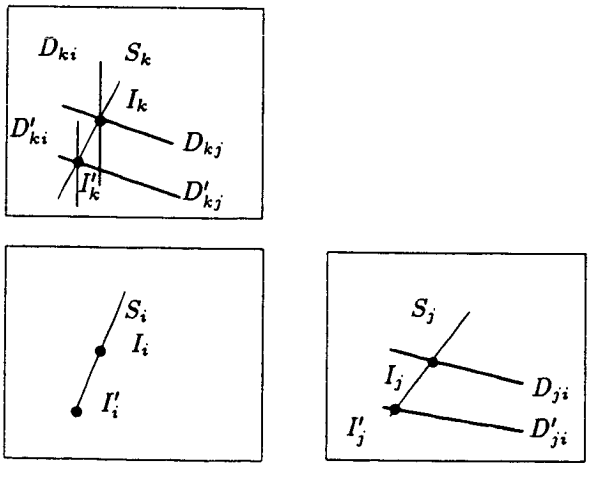

Figure 15: Prediction of orientation

8.5 Validation

Even using a third camera is not always enough to get rid of all ambiguities and false matches. Probabilities are indeed low, but not zero, that using an incorrect match between the first two cameras, we find a compatible segment at the predicted location in the third image.

How can we get rid of those wrong matches ? For this we enforce two constraints :

- the unicity constraint : in its simplest form, it states that a given primitive must be part of at most one triplet. This definition does not take into account the fact that there may be errors in segmentation during the polygonal approximation phase, so that in some cases, a segment must be allowed to match several segments. Figure 10.a shows such a case. Figure 10.b shows, on the contrary, an exemple of a wrong match violating the unicity constraint.

  We must therefore generalize the notion of unicity for line segments. This is why we first compute, for a segment in a matched triplet, the "matching length" involved for this precise triplet. This is done by clipping the segment by the epipolar lines corresponding to the endpoints of the other segments.

- the regularity constraint : if we assume that the objets looked at are smooth, then two segments belonging to the same object and close to each other in the image will be reconstructed as close 3D segments. Therefore, if a match (S_1, S_2, S_3), reconstructing a 3D segment S, is correct, we should be able to find some of the 2D neighbours of the image segments S_i reconstructed into 3D segments close to S. We cannot impose that all 2D neighbours satisfy such a constraint, because there may be discontinuities at the borders of objects, but we can impose that a given percentage of them be such that the distance of the reconstructed neighbours to S be under a threshold.

Following these two constraints, we compute, as a criterion, the ratio of the 2D neighbors reconstructing into a 3D segment sufficiently close to S to the total number of neighbors.

The notion of neighbourhood has been previously defined, and we will now see how to evaluate a distance between two segments. The distance could be defined as the shortest distance between points of the two 3D segments. It would be rigorous but inefficient from a computational point of view, as we do not really need a very accurate numerical criterion of compatibility. We therefore define a 3D segment S' to be ε-compatible with a segment S if and only if S' intersects the rectangular box formed from S by building a 3D rectangle containing S, whose edges are parallel to the axes of coordinates, and extended of ε at each endpoint, as illustrated by figure 16.

Given a tolerated distance ε, we are now able to compute the ratio of the ε-compatible neighbours to the total number of neighbors. The exact value of ε depends on the kind of scenes we observe, in the sense that it must reflect the typical scale of objects, but it is not a critical threshold. If this ratio is too low, lower than 0.25 for our system, then we get rid of the corresponding

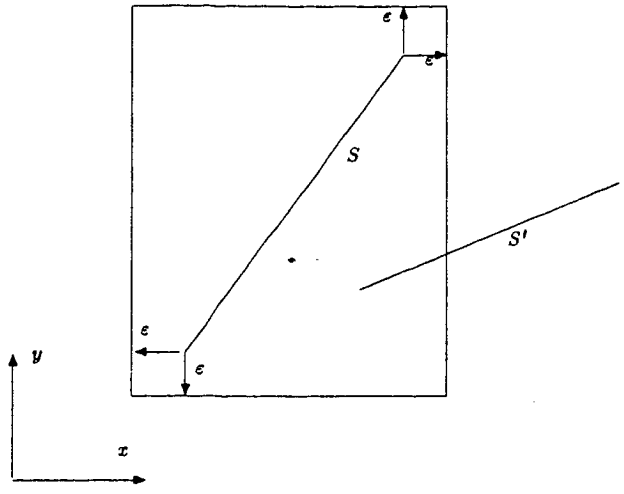

Figure 16: S' is ε-compatible with S iff it intersects the box

match. If there is an ambiguity between two matches, we keep the best with respect to the criterion of compatibility weighted by "matching length" of the segment, thus privileging longer matches against shorter ones.

9 Experimental results

We have tested this algorithm on a number of industrial and office scenes, and we present now some of these results.

9.1 Office scenes

Figure 18 presents triplets of images taken in a robotics laboratory, for different positions of the robot. Figure 19 presents the polygonal approximations of the image contours, extracted by a chain of programs by Rachid Deriche, Gérard Giraudon and Marc Berthod, from INRIA.

Figure 20 presents the triplets of segments matched by the trinocular stereovision program.

Using another program computing the displacement between the different frames, we are able to build the 3D reconstruction of the whole robotics room : figure 21 shows the view from above, and figure 22 brings some explainations in the form of a commented sketch of this room. (Details on the construction of this global 3D map can be found in the book of N. Ayache [37,38] and in previous papers [39,36].

Table 1 gives the performance features of the algorithm on these six different scenes. The programs are written in C and run on a SUN-3 workstation.

⊙⊙⊙ **Procedure** VALIDATE-3 (1,2,3)

for each triplet of segments (S_1, S_2, S_3)

- **for** each neigbour triplet of segments (S'_1, S'_2, S'_3) i.e. such that S'_1 is neighbour of S_1 in image 1 or S'_2 is neighbour of S_2 in image 2 or S'_3 is neighbour of S_3 in image 3

  if (S'_1, S'_2, S'_3) is ε-compatible with (S_1, S_2, S_3)

  then increment the number of compatible neighbors of (S_1, S_2, S_3).

  endfor

- Compatibility $c_{(S_1,S_2,S_3)}$ = Compatible neighbours / Total number of neighbours.
- **if** $c_{(S_1,S_2,S_3)} \leq$ compatibility-threshold **then** get rid of (S_1, S_2, S_3).

endfor

for each triplet of segments (S_1, S_2, S_3)

for each neigbour triplet of segments (S'_1, S'_2, S'_3)

if (S'_1, S'_2, S'_3) is ambiguous with (S_1, S_2, S_3) and $c_{(S_1,S_2,S_3)} \ll c_{(S'_1,S'_2,S'_3)}$

then get rid of (S_1, S_2, S_3).

endfor

endfor

if undecidable ambiguities remain **then** get rid of the ambiguous triplets.

EndProcedure VALIDATE-3 ⊙⊙⊙

Figure 17: Validation algorithm

	Nb. Segments			*Nb. Matches*		
	image 1	*image 2*	*image 3*	*Segments*	*Points*	*CPU Time*
Scene 1	312	337	336	160	3300	3s
Scene 2	283	266	280	111	2832	2s
Scene 3	262	240	284	110	3021	2s
Scene 4	203	199	205	75	2522	1s
Scene 5	393	405	371	320	4906	6s
Scene 6	548	531	536	539	7200	10s

Table 1: Performance of the trinocular stereovision algorithm on the six indoor scene

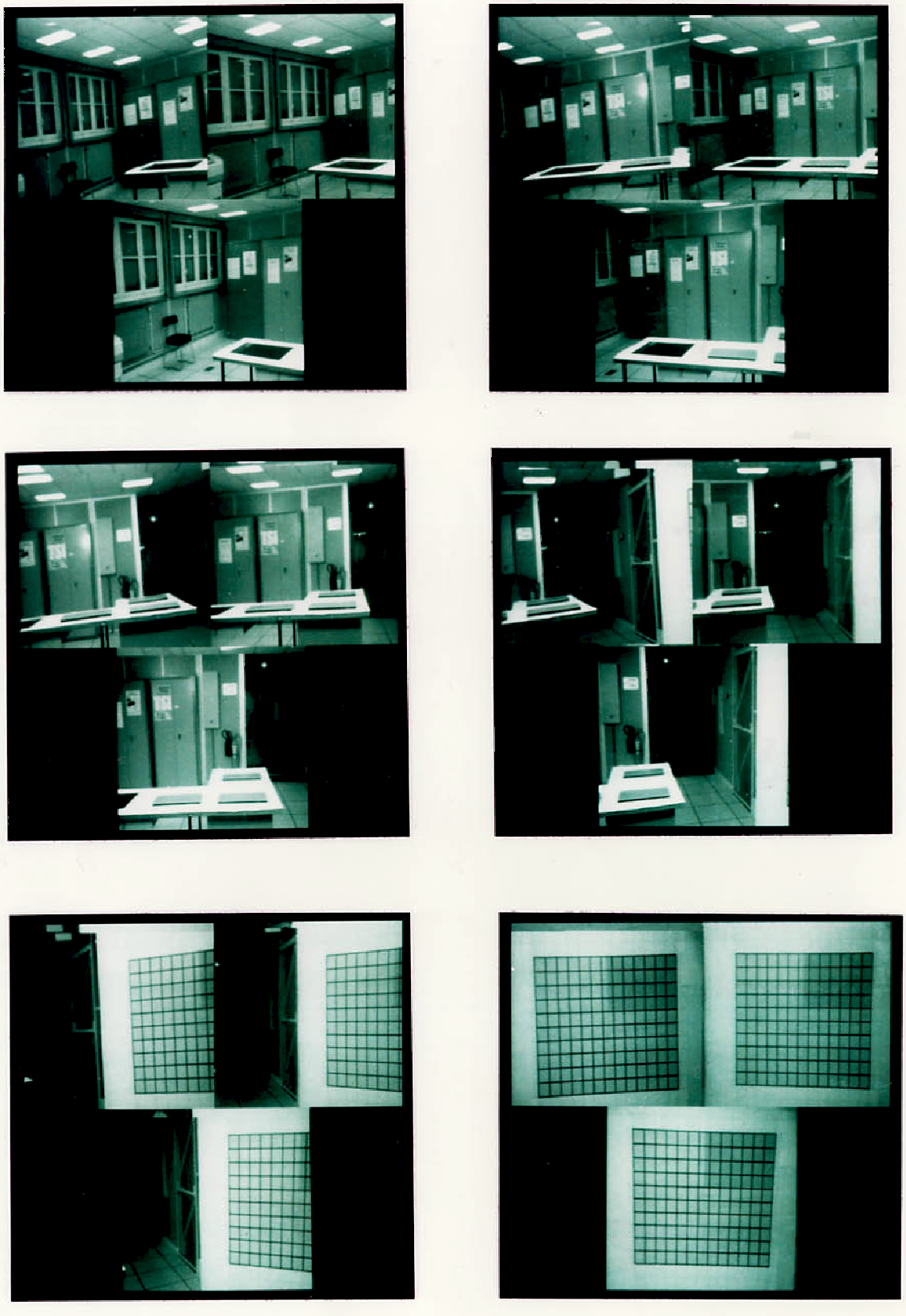

Figure 18: Triplets of images taken in the robotics room

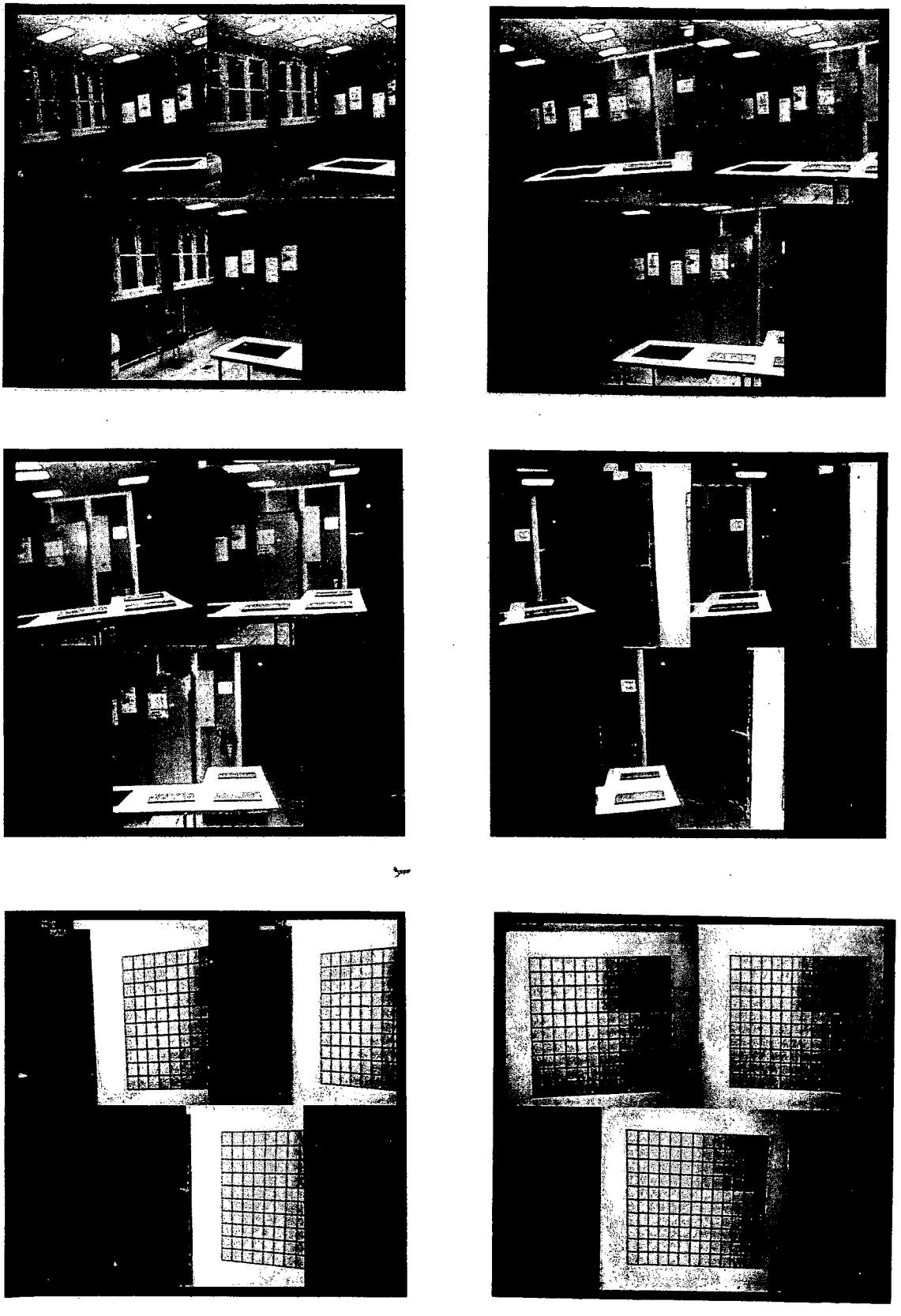

Figure 18: Triplets of images taken in the robotics room

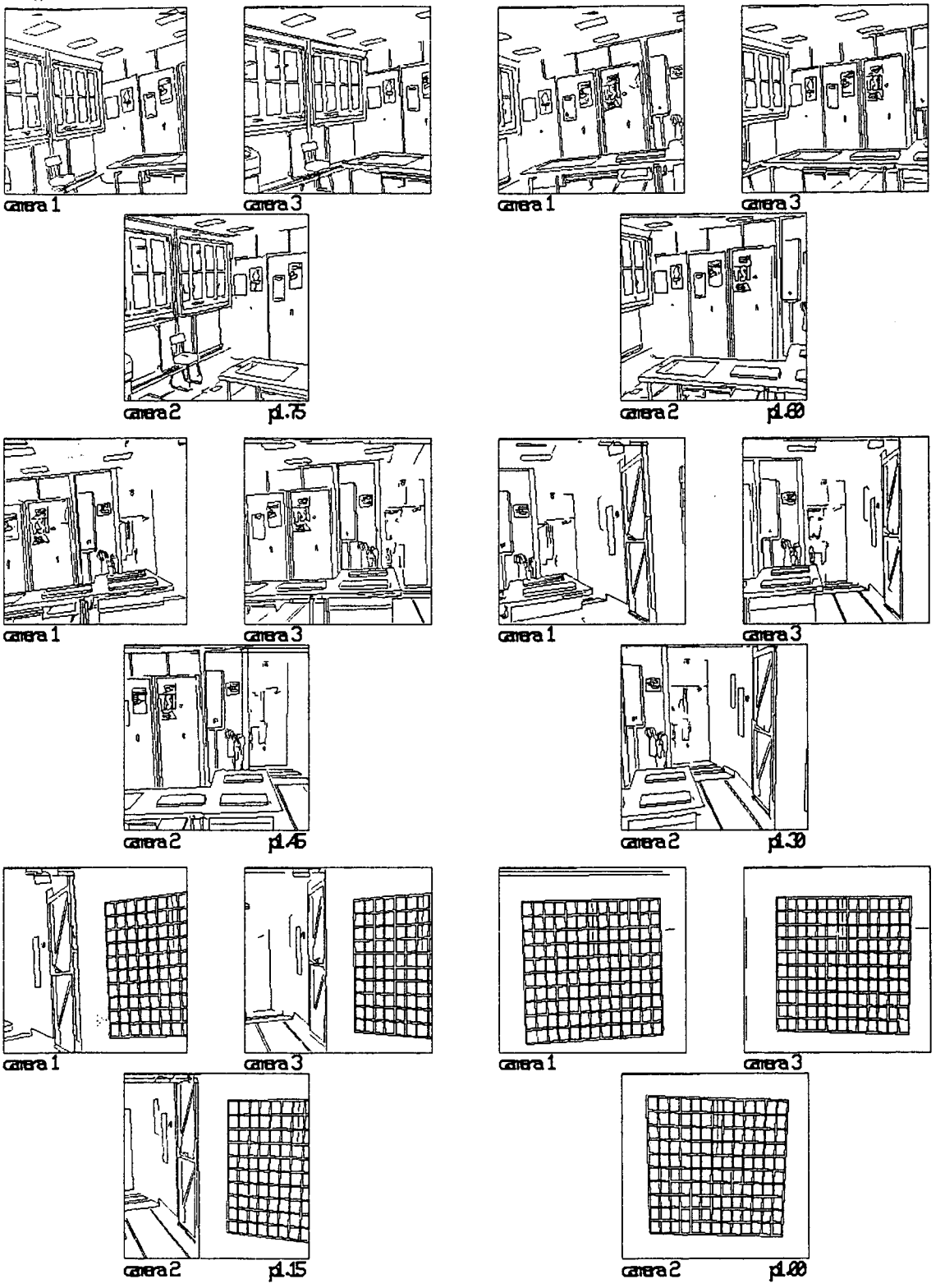

Figure 19: Polygonal approximation of the contours

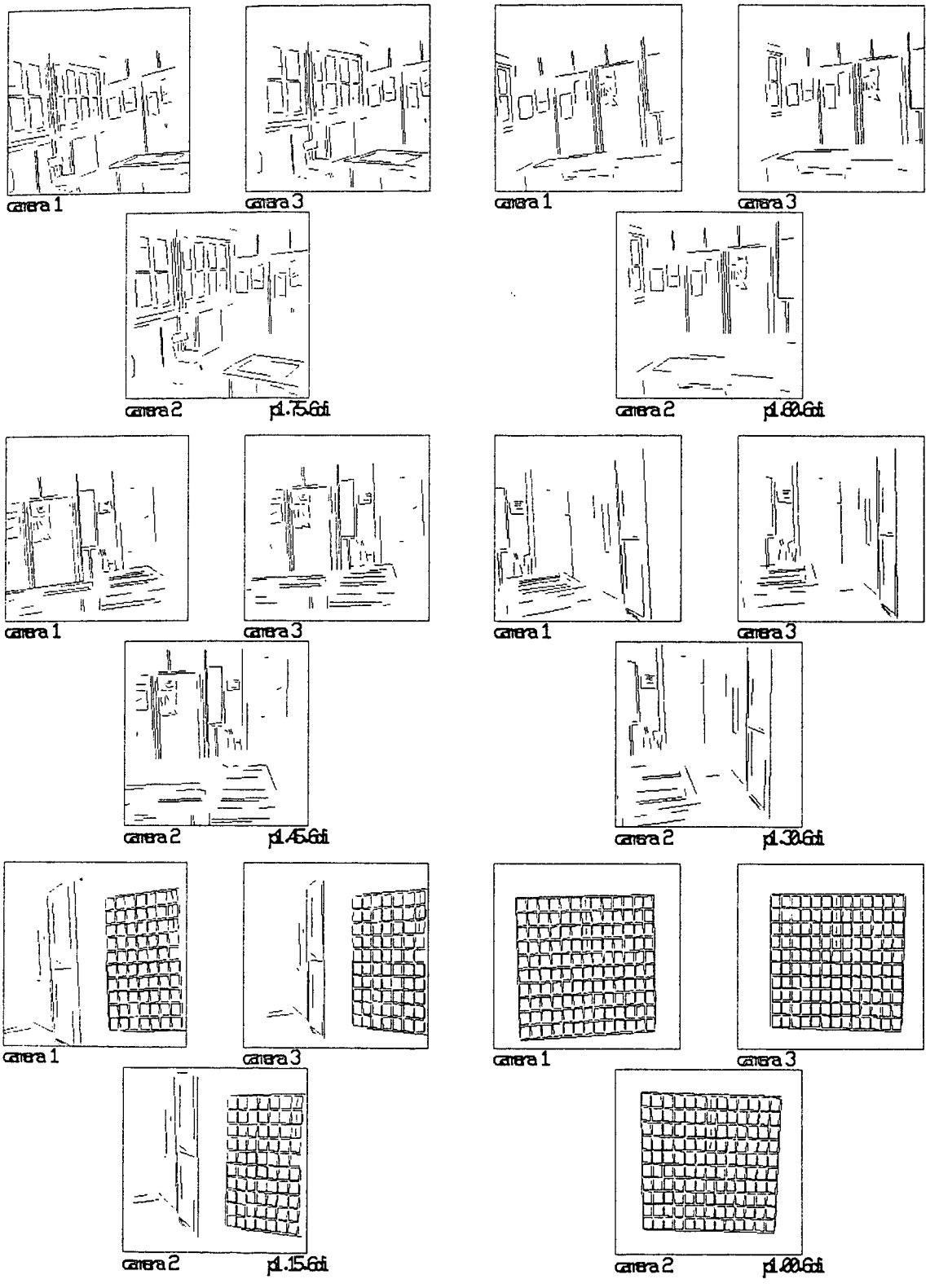

Figure 20: Matched segments

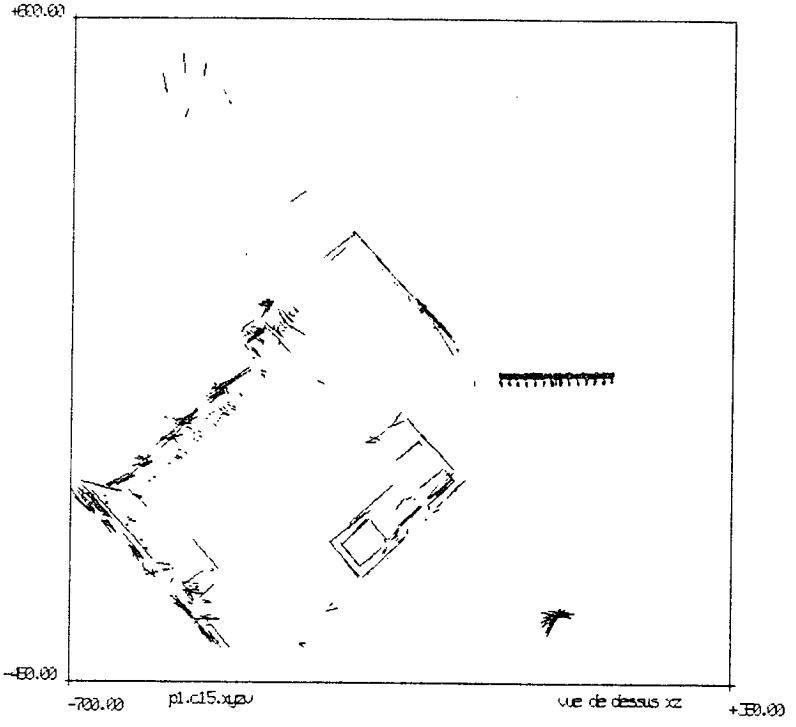

Figure 21: View from above of the robotics room

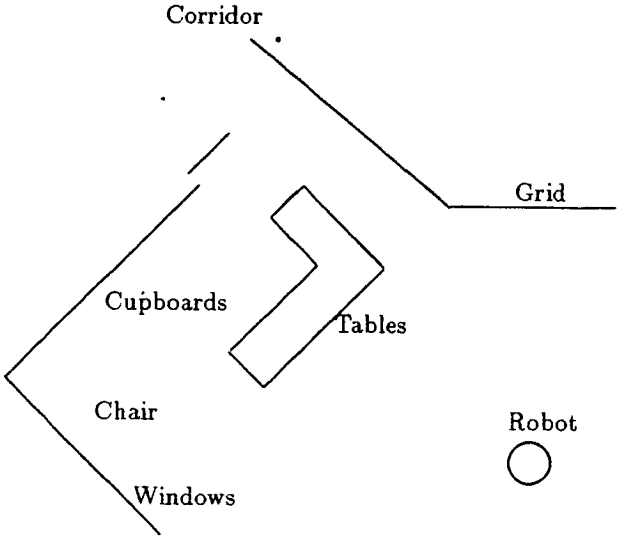

Figure 22: Commented sketch of the view from above

9.2 Industrial scenes

The previously described trinocular stereovision algorithm is particularly well adapted to scenes with long line segments. Nevertheless, it works also on images with curved edges, with more computations because the polygonal approximations output more segments. Figures 23, 24 and 25 present, respectively, a triplet of images of an industrial object, its contours and the matched segments.

Triplets of images representing a cone, a cylinder and a sphere are shown on figure 26.

Figures 27 to 30 show the original contours and the results of the matching on a sphere, a cone and a cylinder.

Though the polygonal approximations include a greater number of smaller segments, matching results remain correct.

It might nevertheless seem that more matchings coud be done. That comes from the fact that some small segments are very noisy and cannot be predicted accurately enough in the third camera. If the thresholds should be relaxed so that these segments be matched, errors would be introduced elsewhere. Moreover, those segments would reconstruct very noisy 3D segments.

It should be noted that these results allow for quantitative analyses on the observed objects. Pavel Grossmann [40], has developed a program to use the reconstructed 3D segments for detecting quadrics. It recognizes, for example, a portion of a sphere in the first example, and estimates its radius to be approximately 46 mm, with an accuracy of one millimeter.

Finally, the last figure shows two projections of the reconstructed 3D segments of the cylinder.

10 Conclusion

We presented a new trinocular stereovision technique. It can be summarized by the following stages:

- Calibration: a preliminary procedure allows for the computation and the storage of the parameters required to determine epipolar geometry between cameras.
- Preprocessing: a graph based description of a polygonal approximation of the contours is extracted from each image. Then images are rectified to simplify the epipolar geometry.
- Hypotheses Prediction-Verification: triplets of potential matches are derived from the previously constructed graphs by simple geometric verifications.
- Validation: local consistency checks are performed to remove erroneous matches.

The main features of the method are

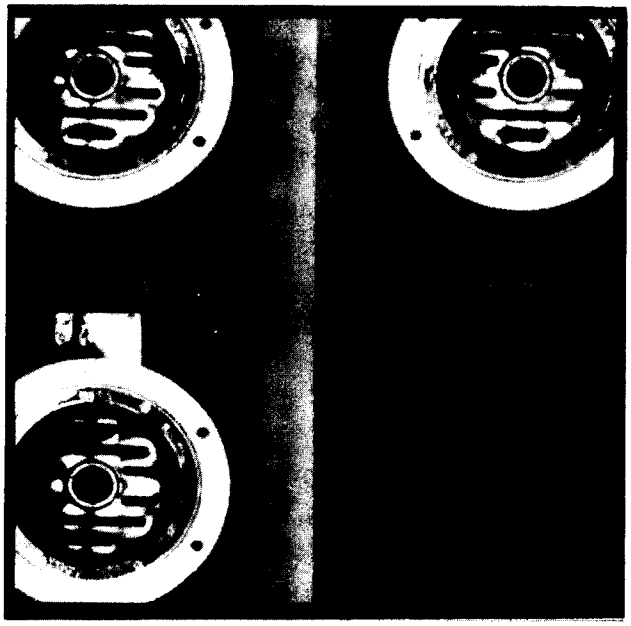

Figure 23: Triplet of images of an industrial object (ELSAG)

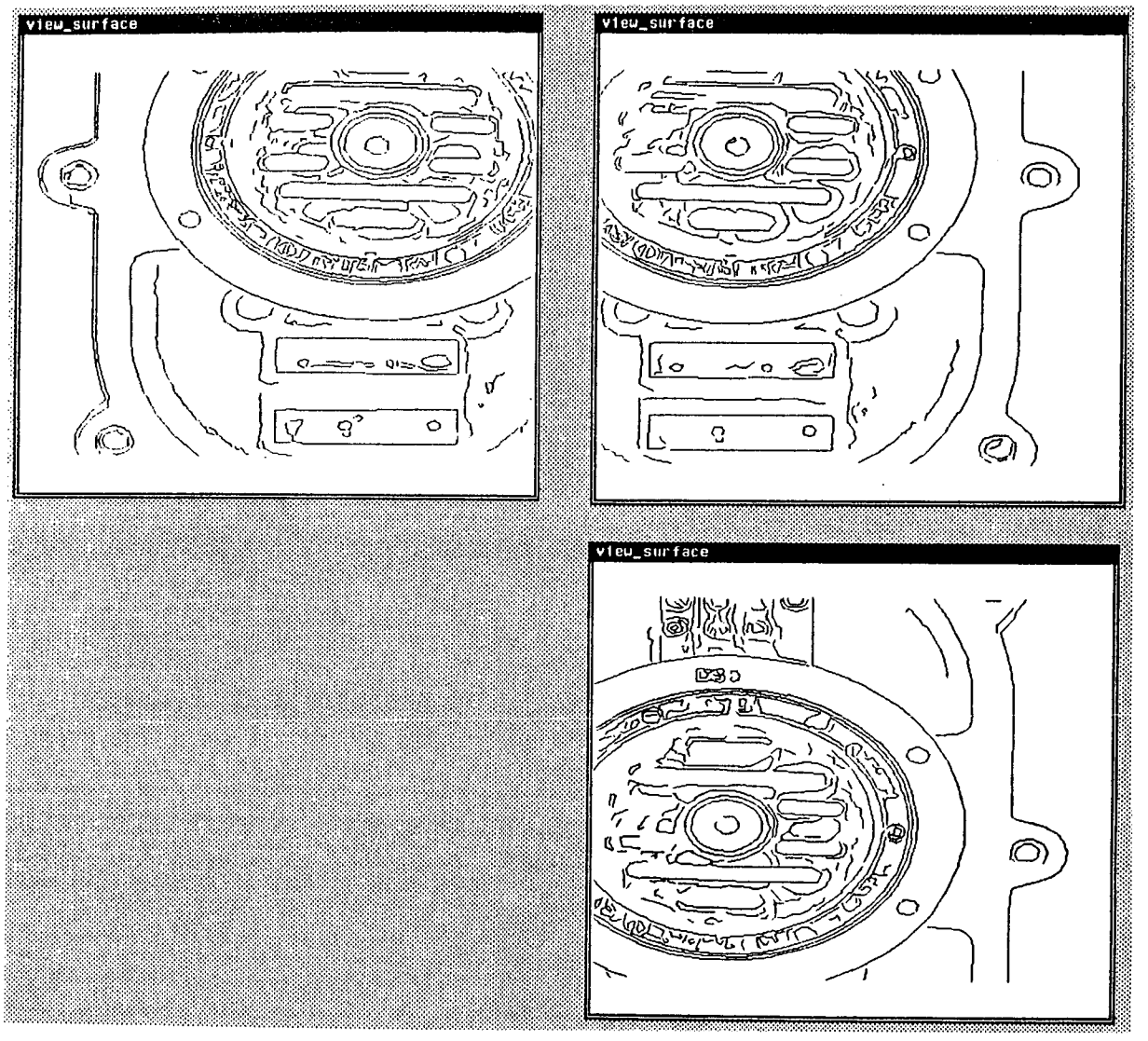

Figure 24: Contours of a triplet of images of an industrial object

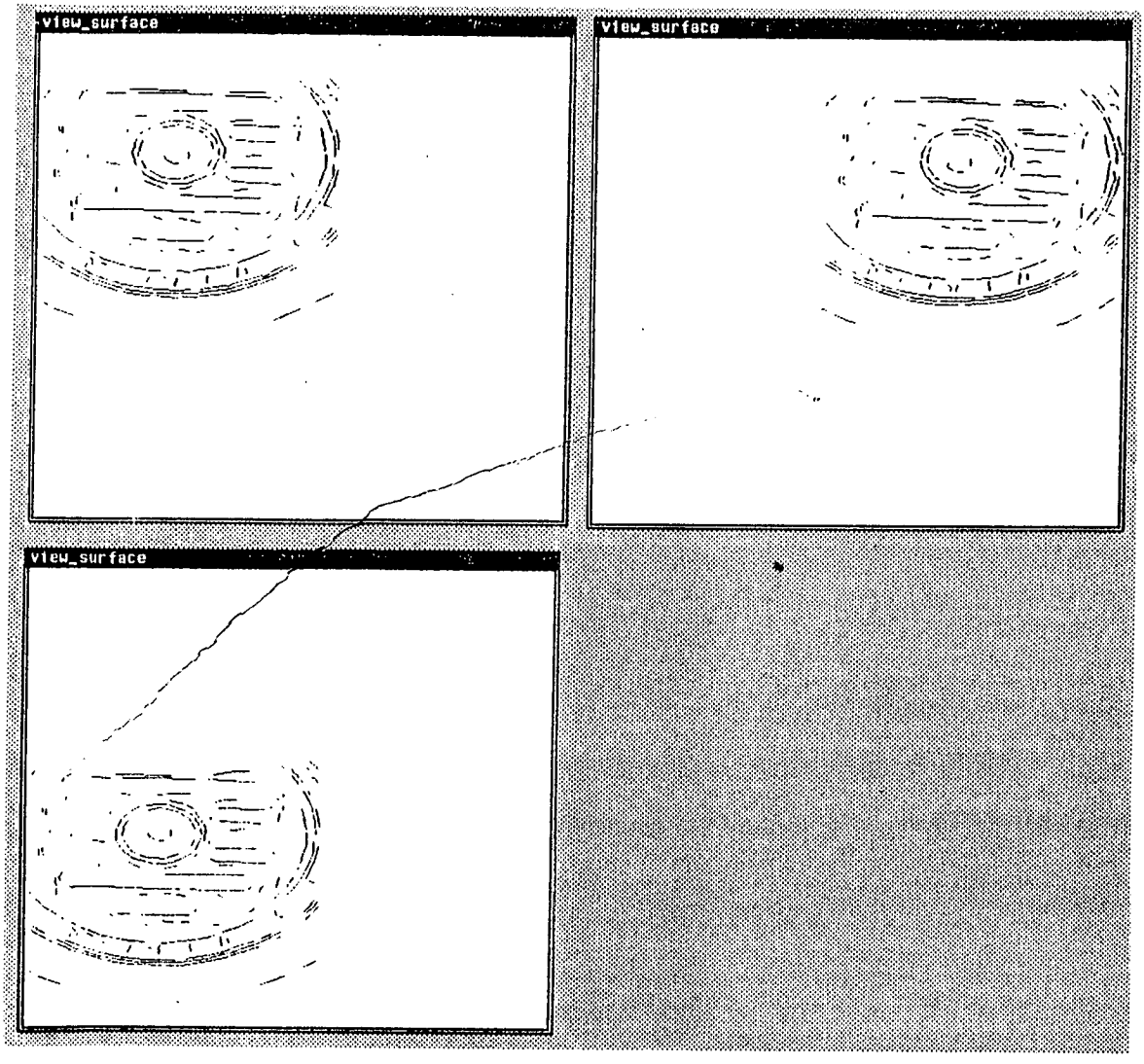

Figure 25: Matched segments

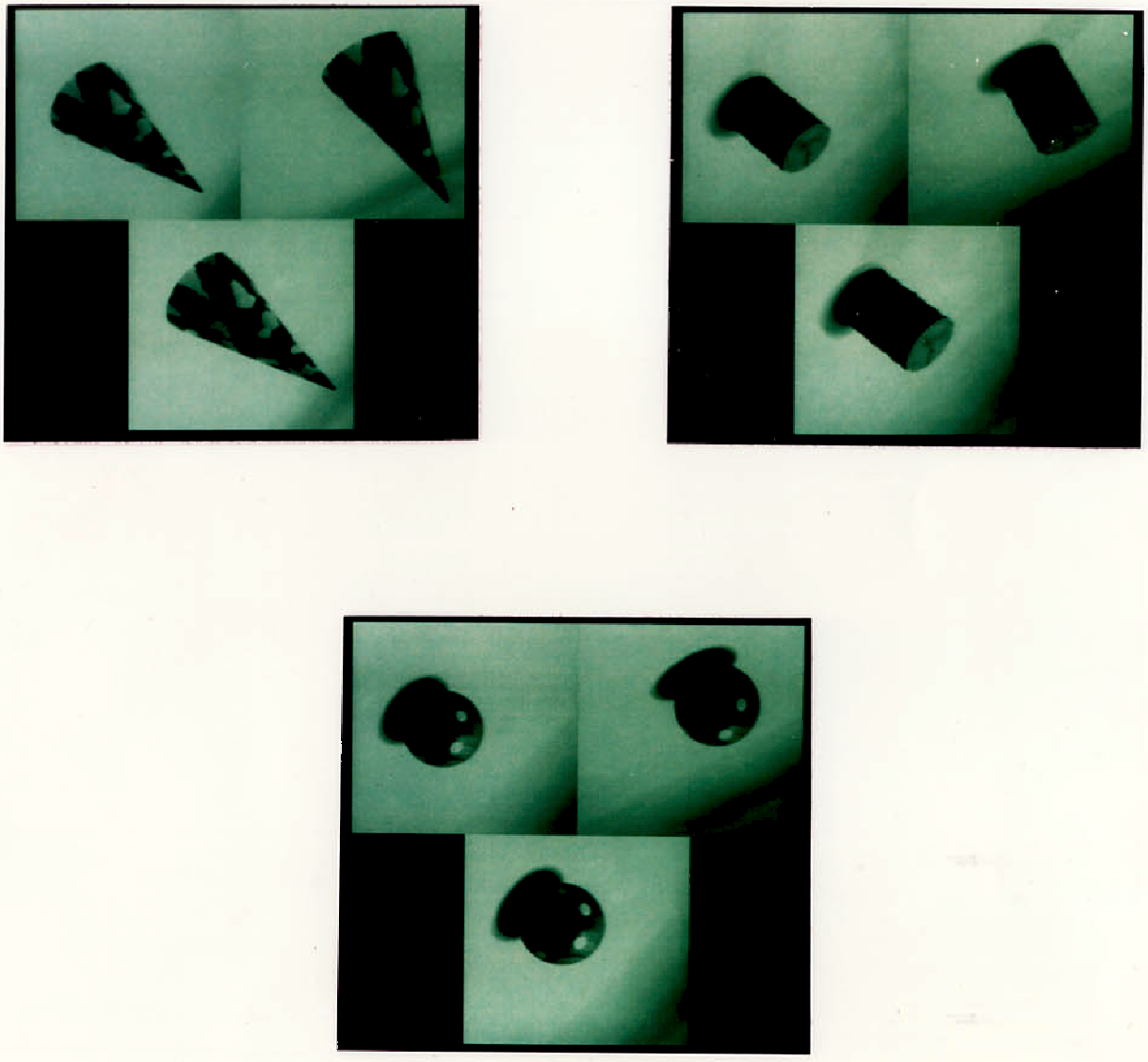

Figure 26: Triplet of images of a cone, a cylinder and a sphere (ELSAG)

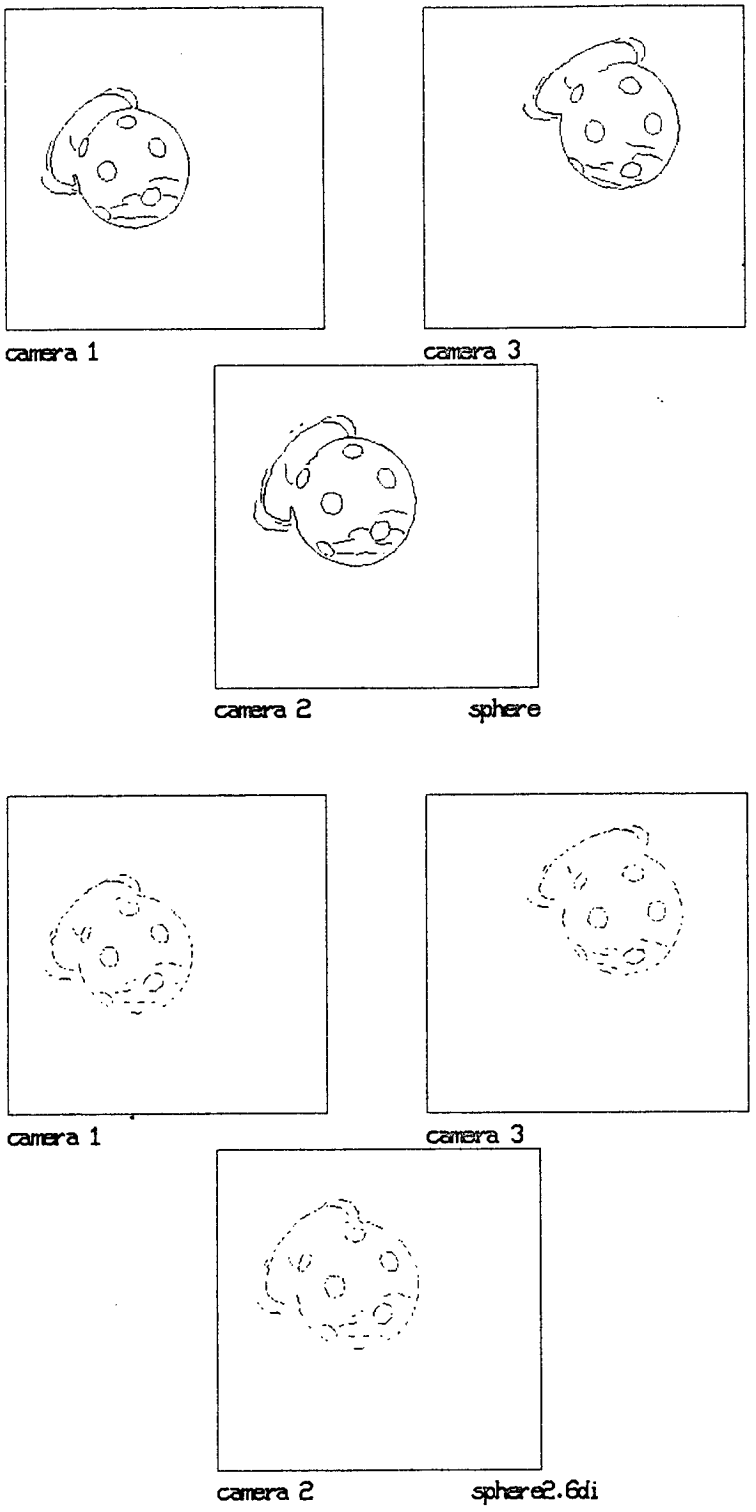

Figure 27: Contours (a) and matches (b) of a triplet of images of a sphere

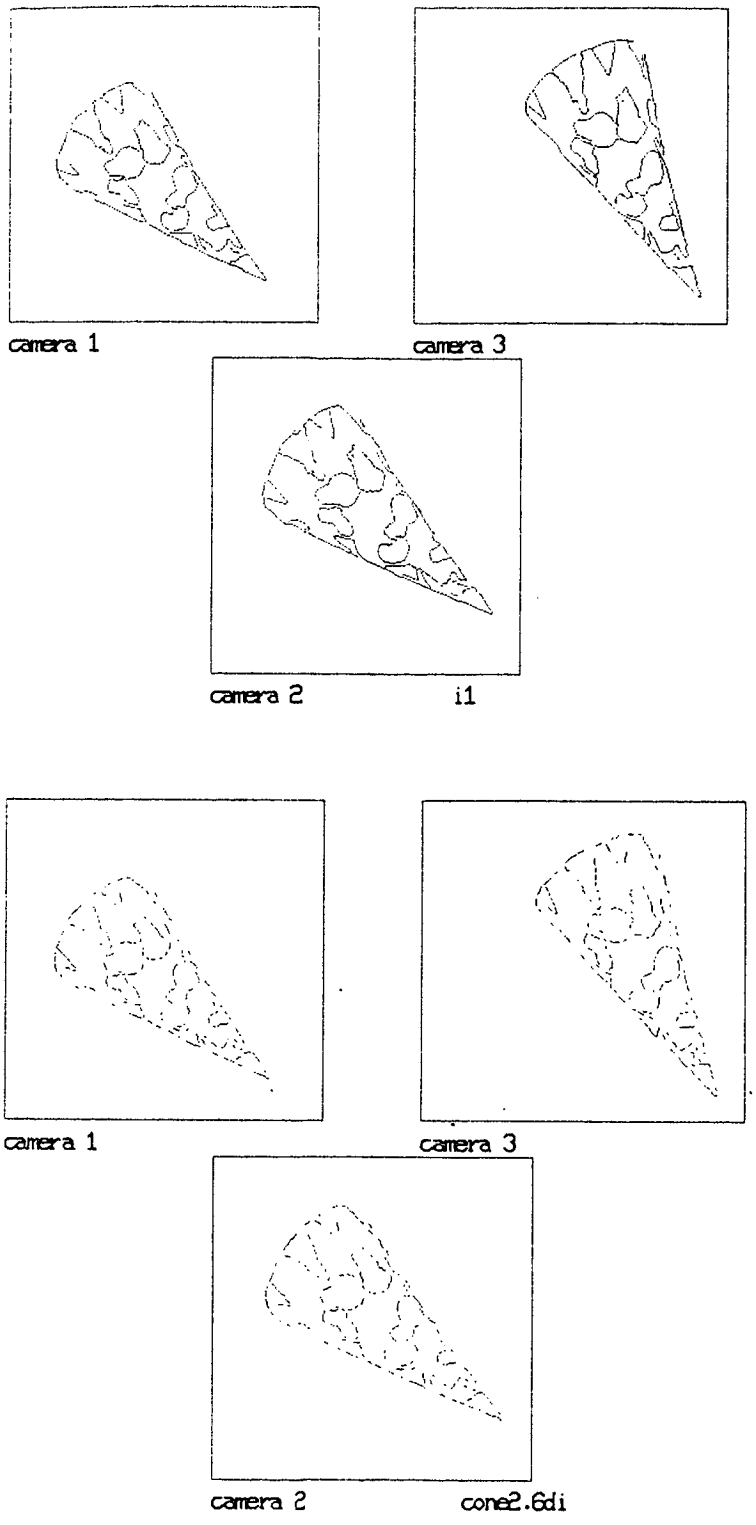

Figure 28: Contours (a) and matches (b) of a triplet of images of a cone

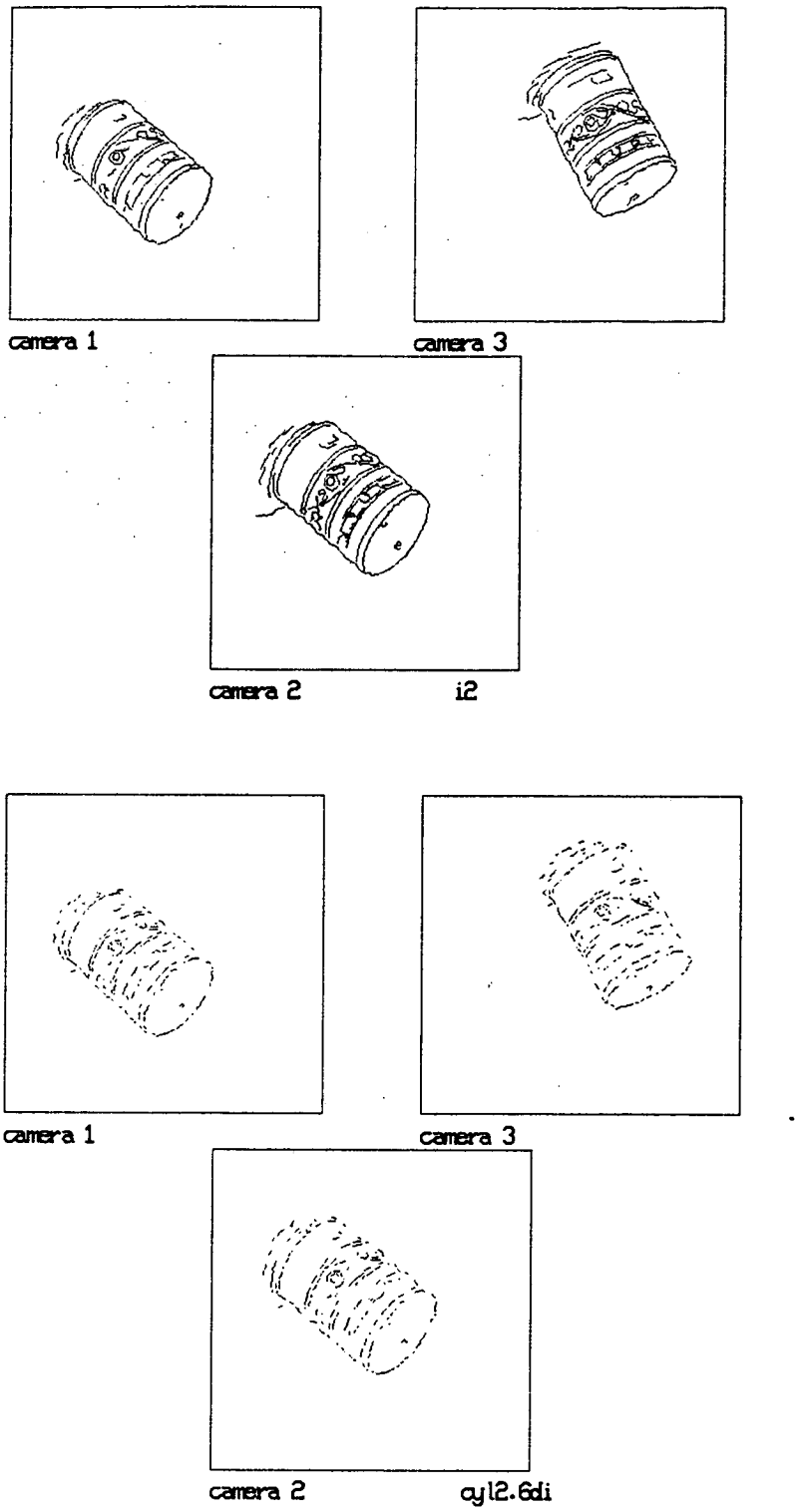

Figure 29: Contours (a) and matches (b) of a triplet of images of a cylinder

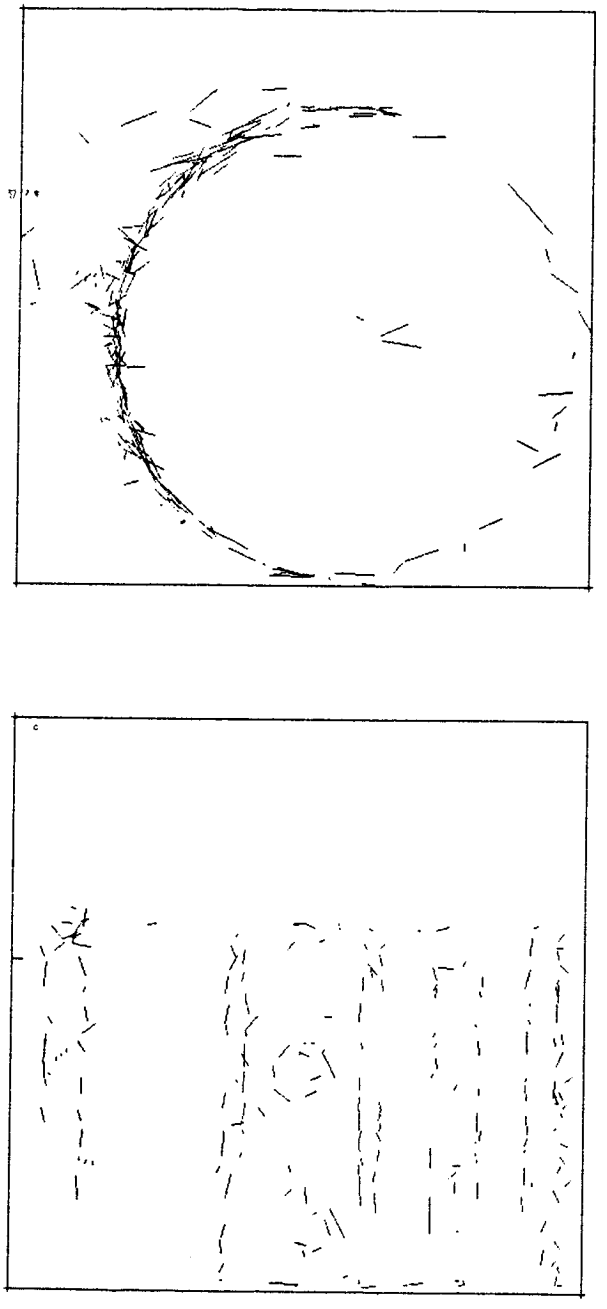

Figure 30: View from above (a) and side view (b) of the reconstructed 3D segments for the cylinder

- Flexibility: it allows for arbitrary positions of three different cameras. Calibration is obtained by a simple automatic procedure.
- Rapidity: Matching times are a typically a few seconds. Moreover, it is straightforward to implement the algorithms in parallel.
- Reliability: the use of a third camera renforces the geometric constraints, and therefore reduces the influence of heuristics in the matching process. This improves significantly the robustness of the method.
- Accuracy: the use of a third camera provides an additional measurement. This improves strongly the 3D reconstruction accuracy.

Current and future developments concern the parallel implementation of the algorithm on a multiprocessor machine to perform stereo-matching and 3D reconstruction at the rate of 5Hz. This is done within a European *Esprit Project* (Project P940 involving ELSAG, GEC, INRIA, MATRA, NOESIS, Univ. of Cambridge, Univ. of Genova) where preprocessing (edge extraction, edge linking, polygonal approximation) is performed by dedicated hardware at the rate of 24 Hz. We believe that the production of 3D maps in an indoor environment at the rate of 5 Hz will be achieved within this 5 year project, i.e. by 1991. This should be reported in forthcoming reports and papers.

Acknowledgements

The stereo and 3D reconstruction programs are the end of a long chain of programs (and hopefully the start of another longer chain). We would therefore like to thank all the authors of this chain, and especially Marc Berthod, Nour-Eddine Deriche and Gerard Giraudon, from INRIA.

The rectification scheme has been developed and improved in collaboration with Chuck Hansen, from the University of Utah.

We would also like to thank Olivier Faugeras, head of the Vision and Robotics Laboratory at INRIA-Rocquencourt, for his constant interest in our project and its valuable suggestions.

A Computing a 3D line D from its 2D projections d_i

One assumes that the perspective transformation of each camera is represented by a 3x4 matrix T_i computed during a preliminary calibration stage.

Therefore, the image of a generic point $P = (x, y, z)^t$ of D by camera i is $I_i' = (u_i', v_i')^t$ such

that:

$$u'_i = \frac{(at^i_{11} + bt^i_{12} + t^i_{13})z + pt^i_{11} + qt^i_{12} + t^i_{14}}{(at^i_{31} + bt^i_{32} + t^i_{33})z + pt^i_{31} + qt^i_{32} + t^i_{34}}$$
$$v'_i = \frac{(at^i_{21} + bt^i_{22} + t^i_{23})z + pt^i_{21} + qt^i_{22} + t^i_{24}}{(at^i_{31} + bt^i_{32} + t^i_{33})z + pt^i_{31} + qt^i_{32} + t^i_{34}}$$

where t^i_{jk} is the element of rank (j,k) in the perspective matrix T_i.

Saying that I'_i belongs to d_i means that

$$\alpha_i u'_i + v'_i + \mu_i = 0$$

If the preceding relation has to be verified for any $P \in D$, except C_i, then the following two equations must hold:

$$\alpha_i(at^i_{11} + bt^i_{12} + t^i_{13}) + (at^i_{21} + bt^i_{22} + t^i_{23}) + \mu_i(at^i_{31} + bt^i_{32} + t^i_{33}) = 0$$
$$\alpha_i(pt^i_{11} + qt^i_{12} + t^i_{14}) + (pt^i_{21} + qt^i_{22} + t^i_{24}) + \mu_i(pt^i_{31} + qt^i_{32} + t^i_{34}) = 0$$

By reorganizing the coefficients, one can see that these equations are the equations 10 and 11.

References

[1] M. Yachida. 3d data acquisition by multiple views. In O.D.Faugeras and G.Giralt, editors, *Robotics Research: the Third International Symposium*, pages 11–18, MIT Press, Cambridge, Mass., 1986.

[2] M. Yachida, Y. Kitamura, and M. Kimachi. Trinocular vision: new approach for correspondence problem. In *Proc. International Conference on Pattern Recognition*, pages 1041–1044, IEEE, October 1986. Paris, France.

[3] M. Pietikainen and D. Harwood. Progress in trinocular stereo. In *Proceedings NATO Advanced Workshop on Real-time Object and Environment Measurement and classification, Maratea, Italy*, August 31 - September 3 1987.

[4] M. Ito and A. Ishii. Three-view stereo analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:524–531, July 1986.

[5] M. Ito and A. Ishii. Range and shape measurements using three-view stereo analysis. In *Proc. International Conference on Computer Vision and Pattern Recognition*, pages 9–14, IEEE, 1986. Miami Beach, Florida.

[6] A. Gerhard, H. Platzer, J. Steurer, and R. Lenz. Depth extraction by stereo triples and a fast correspondence estimation algorithm. In *Proc. International Conference on Pattern Recognition*, pages 512–515, IEEE, October 1986. Paris, France.

[7] E. Gurewitz, I. Dinstein, and S. Sarusi. More on the benefit of a third eye for machine stereo perception. In *Proc. International Conference on Pattern Recognition*, pages 966–968, IEEE, October 1986. Paris, France.

[8] Y. Ohta, M. Watanabe, and K. Ikeda. Improving depth map by right-angled trinocular stereo. In *Proc. International Conference on Pattern Recognition*, pages 519–521, IEEE, October 1986. Paris, France.

[9] M. Pietikainen and D. Harwood. Depth from three-camera stereo. In *Proc. International Conference on Computer Vision and Pattern Recognition*, pages 2–8, IEEE, 1986. Miami Beach, Florida.

[10] C. Stewart and Dyer C. *The trinocular general support algorithm : a three camera stereo algorithm for overcoming binocular matching errors.* Technical report 768, University of Wisconsin-Madison, May 1988.

[11] K. Nishihara and T. Poggio. Stereo vision for robotics. In R. Paul and M. Brady, editors, *Robotics Research, The First International Symposium*, pages 489–505, MIT Press, 1984.

[12] D. Marr and T. Poggio. A computational theory of human stereo vision. *Proc. R. Soc.*, B-204:301–328, 1979.

[13] H. Baker and T.O. Binford. Depth from edge and intensity based stereo. In *Proceedings 7th Joint Conference on Artificial Intelligence, Vancouver, Canada*, pages 631–636, August 1981.

[14] M. Berthod and P. Long. Graph matching by parallel optimization methods: an application to stereo vision. In *Proc. International Conference on Pattern Recognition*, pages 841–843, August 1984. 7th, Montreal, Canada.

[15] R. Mohr and W. Wrobel. La correspondance en stéréovision vue comme une recherche de chemin optimal. In *Proc. Reconnaissance des formes et intelligene artificielle*, pages 71–79, 1984. France, quatrième congrès.

[16] W.E.L. Grimson. Computational experiments with a feature based stereo algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7, No 1:17–34, 1985.

[17] G. Medioni and R. Nevatia. Segment-based stereo matching. *Computer Vision, Graphics, and Image Processing*, 31:2–18, 1985.

[18] Y. Ohta and T. Kanade. Stereo by intra- and inter-scanline search. *IEEE Transactions on PAMI*, 7, No 2:139–154, 1985.

[19] S.B. Pollard, J.E.W. Mayhew, and J.P. Frisby. PMF : a stereo correspondence algorithm using a disparity gradient constraint. *Perception*, 14:449–470, 1985.

[20] A.R. Saint-Vincent. *Perception et modélisation de l'environnement d'un robot mobile : une approche par stéréovision.* Thèse, "Université Paul Sabatier, Toulouse", 1986.

[21] S.T. Barnard. Stereo matching by hierarchical microcanonical annealing. In *Proceedings 10th IJCAI, Milano*, pages 832–835, August 1987.

[22] T. Skordas and R. Horaud. *Stereo Correspondence Through Feature grouping and Maximal Cliques.* Rapport de Recherche RR 677-I- 64 LIFIA, IMAG, Laboratoire d'Informatique Fondamentale et d'Intelligence Artificielle, Septembre 1987.

[23] N. Ayache and B. Faverjon. Efficient registration of stereo images by matching graph descriptions of edge segments. *The International Journal of Computer Vision*, 1(2), April 1987.

[24] R. Deriche. Using canny's criteria to derive an optimal edge detector recursively implemented. *The International Journal of Computer Vision*, 2, April 1987.

[25] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8 No6:679–698, 1986.

[26] G. Giraudon. *Chaînage efficace de contour.* Rapport de Recherche 605, INRIA, Février 1987.

[27] J.G. Dunham. Optimum uniform piecewise linear approximation of planar curves. *IEEE transactions on PAMI*, 8 (1):66–75, Janvier 1986.

[28] G. Toscani. *Système de Calibration optique et perception du mouvement en vision artificielle.* PhD thesis, Paris-Orsay, 1987.

[29] O.D. Faugeras and G. Toscani. The calibration problem for stereo. In *Proceedings CVPR '86, Miami Beach, Florida*, pages 15–20, IEEE, 1986.

[30] O.D. Faugeras. *Artificial 3D Vision.* To appear, 1988.

[31] N. Ayache and C. Hansen. Rectification of images for binocular and trinocular stereovision. In *Proc. International Conference on Pattern Recognition*, October 1988. 9th, Beijing, China.

[32] N. Ayache. Construction et fusion de représentations visuelles tridimensionnelles; applications à la robotique mobile. Thèse d'Etat, Université de Paris-Sud, Orsay, May 1988. INRIA Internal Report.

[33] F. Lustman. *Vision stéréoscopique et perception du mouvement en vision artificielle.* PhD thesis, Paris-Orsay, 1987.

[34] N. Ayache and F. Lustman. Fast and reliable passive trinocular stereovision. In *Proc. First International Conference on Computer Vision*, pages 422–427, IEEE, June 1987. London, U.K.

[35] N. Ayache and F. Lustman. Trinocular stereovision, recent results. In *Proc. International Joint Conference on Artificial Intelligence*, August 1987. Milano, Italy.

[36] N. Ayache and O.D. Faugeras. Maintaining representations of the environment of a mobile robot. In *International Symposium on Robotics Research*, August 1987. Santa-Cruz, California.

[37] N. Ayache. *Vision Stéréoscopique et Perception Multisensorielle; Applications à la robotique mobile.* Inter-Editions, 1989.

[38] N. Ayache. *Stereoscopic Vision and Multisensor Perception; Applications to Mobile Robotics.* MIT-Press, 1989. To appear.

[39] N. Ayache and O.D. Faugeras. Building, registrating and fusing noisy visual maps. *International Journal of Robotics Research*, 7(6):45–65, December 1988. Special Issue on Sensor Data Fusion.

[40] P. Grossman. Compact — a surface representation scheme. In *4th. Alvey Vision Conference (AVC'88)*, 1988. Manchester, England.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique