[7] W.J. Wolfe and K. Jones, "Camera calibration using the perspective view of a triangle," in *Proc. SPIE Conf. Automated Inspection Measurement,* vol. 730, Cambridge, MA, Oct. 28–30, 1986.
[8] J. Tietz and T. Richardson, "Development of an autonomous video rendezvous and docking system," Final Reps. Contract NAS8–34679, Martin Marietta Denver Aerospace, June 1982, June 1983, Jan. 1984.
[9] S. Linnainmaa, D. Harwood, and L. Davis, "Pose determination of a three dimensional object using triangle pairs," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 10, no. 5, Sept. 1988.
[10] D. Thompson and J. Mundy, "Three dimensional model matching from an unconstrained viewpoint," *Proc. Int. Conf. Robotics Automation,* 1987, pp. 208–220.
[11] J. Mundy, A. Heller, and D. Thompson, "The concept of an effective viewpoint," presented at the DARPA Image Understanding Workshop, Cambridge MA, Apr. 1988.
[12] Y. Hung, P. Yeh, and D. Harwood, "Passive ranging to known planar point sets," presented at the IEEE Int. Conf. Robotics and Automation, St. Louis, MO, Mar. 25–28, 1985.
[13] K. Cronils and P. Goode, "Location of planar targets in three space from monocular image," in *Proc. Goddard Conf. Space Applications Of Artificial Intelligence Robotics,* Greenbelt, MD, May 13–14, 1987.
[14] R. Haralick, "Determining camera parameters from the perspective projection of a rectangle," Tech. Note, Virginia Polytech. Inst., Blacksburg, VA, June 1982.
[15] W.J. Wolfe, D. Mathis, C. Weber, and M. Magee, "Integration of model-based computer vision and robotic planning," in *Proc. SPIE Cambridge Symp. Advances in Intelligent Robotic Systems,* Cambridge, MA, Nov. 6–11, 1988.
[16] W.J. Wolfe, G. White, and L. Pinson, "A multisensor robotic locating system and the camera calibration problem," in *Proc. SPIE Conf. Intelligent Robots Comput. Vision,* vol. 579, Cambridge, MA, Sept. 16–20, 1985.
[17] R. Thom, *Structural Stability And Morphogenesis.* New York: W.A. Benjamin, 1975.
[18] W.J. Wolfe, D. Mathis, C. Weber, and M. Magee, "Locating known objects in 3-D from a single perspective view," in *Proc. SPIE Cambridge Symp. Advances in Intelligent Robotic Systems,* Cambridge, MA, Nov. 6–11, 1988.
[19] S. Barnard, "Interpreting perspective images," *Artificial Intelligence,* pp. 435–462, Nov. 1983.
[20] J. Yuan, "A general photgrammetric method for determining object position and orientation," *IEEE Trans. Robotics Automat.,* pp. 129–142, Apr. 1989.

Trinocular Stereo Vision for Robotics

Nicholas Ayache and Francis Lustman

***Abstract*— We present an original approach for building a three-dimensional description of the environment of a robot using three cameras. The main advantages of trinocular versus binocular stereo are *simplicity, reliability,* and *accuracy.* We believe that these advantages now make trinocular stereo vision 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.**

***Index Terms*—Computer vision, edge segments, mobile robots, stereo vision, 3-D maps, trinocular.**

Manuscript received August 25, 1989; revised August 8, 1990. Recommended for acceptance by R.J. Woodham. This work was supported in part by Esprit Project P940.

The authors are with INRIA, BP 105, 78153 Le Chesnay Cedex, France.

IEEE Log Number 9040366.

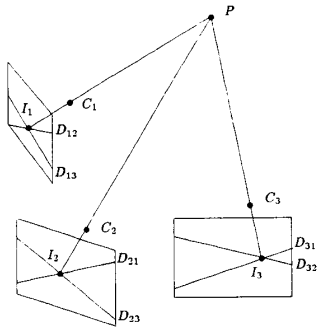

Fig. 1. Geometric constraints of trinocular stereo vision.

I. Introduction

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

Most of the research on passive stereo vision 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 reduces 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]–[10].

For a discussion of both geometric and heuristic constraints used in binocular stereo vision and for a review of research on this topic, one can refer to [11], for example. A nonexhaustive list of publications on the subject is given by the following references [12]–[22]. Last but not least, the work of [23] on binocular stereo vision pioneered the work on trinocular stereo vision presented here, and the following references were kindly suggested by one of the reviewers [24]–[28].

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

II. Geometry of Trinocular Stereo Vision

Fig. 1 illustrates the geometric constraints of trinocular stereo vision. 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, and I_3 form a triplet of *homologous* image points.

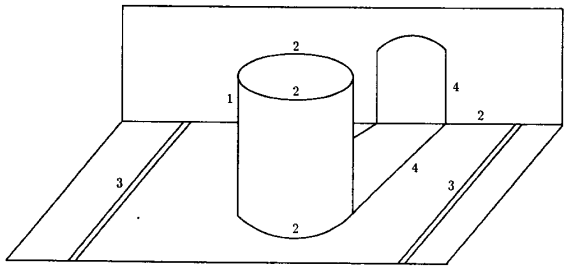

Fig. 2. Different types of edges.

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 *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 *conjugate 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 conjugate epipolar lines.

As one can see in Fig. 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}.

III. Image Representation

The matching algorithm does not run 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. Fig. 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 (types 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.
- *Compactness:* The information contained in the edges is not only very significant but also much more compact in terms of storage and computational burden for matching.
- *Richness of Attributes:* Many useful features can be attached to the edge segments to help solve the stereo matching process. These features can be geometric (length, angle), intensity-based (average contrast along the segment, average intensity of the neighboring regions), or structure-based (edge chains, neighborhoods).
- *Density:* This representation is structured but nevertheless rather dense over the image, therefore enabling enough information to be kept over the whole image.
- *Accuracy:* Our purpose in performing stereo matching is to be able to reconstruct the 3-D 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 subpixel accuracy.
- *Ease of Computation:* There are a number of ways to easily and reliably extract edges from an image.

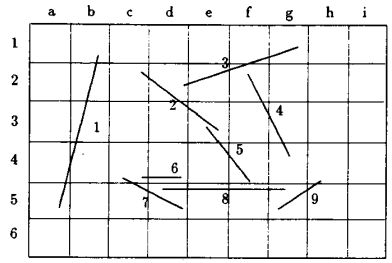

Fig. 3. Computation of buckets and neighborhoods. Neighborhoods: $V(1)$ = Nil; $V(2)$ = {3,5}; $V(3)$ = {2,4}; $V(4)$ = {3}; $V(5)$ = {2}; $V(6)$ = {7}; $V\{8\}$ = {7,9}; $V(9)$ = {8}. Lists of connected neighbors: {1}; {2,3,4,5}; {6,7,8,9}.

A. Edge Extraction

Edge points are the first computed using a very efficient recursive filter developed by Deriche [29] after Canny's ideas [30]. The edge pixels thus obtained are structured into edge chains using a program developed by Giraudon [31], in turn approximated by line segments using a program developed by Berthod. The interested reader will find in [32] a good review of some techniques for polygonal approximation.

Let us note that, to get better accuracy in 3-D reconstruction, our ultimate goal, we perform a least-squares approximation to fit a 2-D linear segment between each pair of successive breakpoints. This allows us to obtain subpixel accuracy, although the accuracy of each original edge point was only one pixel.

B. 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.

C. Buckets

As we show later, the stereo-matching algorithms often require assessing 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. Fig. 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 shown in Fig. 4.

Typically, we used 16×16 buckets for the matching phase, and 8×8 buckets for the validation, because we want to get enough neighbors.

IV. Calibration

A. Image Modeling

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

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

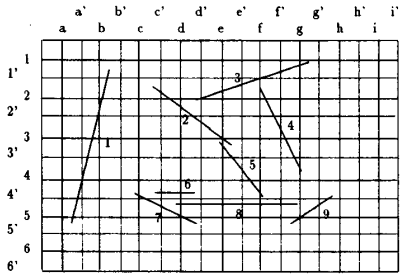

Fig. 4. Better definition of neighborhoods, using two sets of overlapping buckets. Neighborhoods: $V(1)$ = Nil; $V(2) = \{3,5\}$; $V(3) = \{2,4\}$; $V(4) = \{3\}$; $V(5) = \{2,8\}$; $V(6) = \{7,8\}$; $V\{8\} = \{5,6,7,9\}$; $V(9) = \{8\}$. Lists of connected neighbors: $\{1\}$; $\{2,3,4,5,6,7,8,9\}$.

camera retina. The relationship between P and I is modeled 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

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

B. Determining the Perspective Matrix T

In the experiments conducted in our laboratory [33], [34], T is obtained by analyzing a calibration pattern which is a grid painted on a planar surface. The 3-D position of the intersection points of the grid are well known in an absolute 3-D 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:

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

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

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

In theory, six noncoplanar points are sufficient for determining T uniquely [35]. In practice, several dozen points are available, allowing for a global or recursive least squares estimation of T.

V. 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.

[2]On the discussion of this constraint, see [33], [34].

A. Determining Optical Centers

The 3-D coordinates $(x_{C_i}, y_{C_i}, z_{C_i})$ of the optical center C_i of camera i (modeled 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 $(x_{C_i}, y_{C_i}, z_{C_i})$.

B. 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 3-D scene points P having the same image I. If we look at Fig. 1 we see that D is simply the straight line defined by I and C_i.

To determine D analytically, let us rewrite the system of (1) which relates point I to points P in the form

$$\begin{aligned} \left(t_1^i - u_i t_3^i\right)^t P + t_{14}^i - u_i t_{34}^i = 0 \\ \left(t_2^i - v_i t_3^i\right)^t P + t_{24}^i - v_i t_{34}^i = 0 \end{aligned}$$

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 collinear to D is the cross-product of the normals to the planes:

$$n = \left(t_1^i - u_i t_3^i\right) \times \left(t_2^i - v_i t_3^i\right)$$

which yields

$$n = u_i t_2^i \times t_3^i + v_i t_3^i \times t_1^i + t_1^i \times t_2^i$$

which can be written

$$n = N_i I_i^* \quad (2)$$

with

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

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.

C. 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 pint 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

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

If we denote

$$F_j^* = T_j' n \quad (3)$$

where T_j' is the 3×3 submatrix obtained from T_j by suppressing its last column, and

$$E_j^* = T_j \begin{pmatrix} C_i \\ 1 \end{pmatrix} \quad (4)$$

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

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

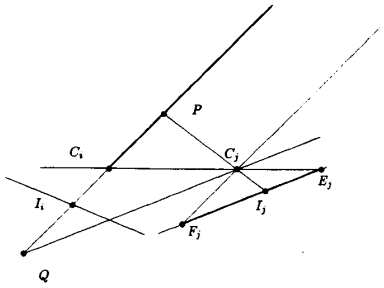

Fig. 5. Computation of an epipolar interval: image point I_i can be matched only with points I_j lying on the epipolar interval F_jE_j.

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 collinear to the epipolar line D_{ji} is obtained by differentiation of (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 (7) that in this case all epipolar lines are parallel to the vector

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

C. Computation of Epipolar Intervals

In practice, the homologous 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. Fig. 5).

Actually, this epipolar interval can be reduced by knowledge of the minimum and maximum possible distances 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 distances, respectively, one sees that the epipolar interval is given by $\{I_m, I_M\}$ and 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 operations must be performed:

[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 Fig. 5.

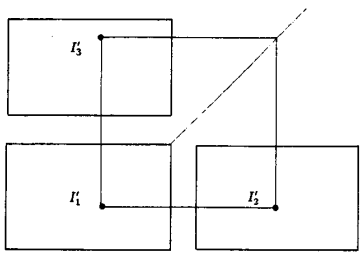

Fig. 6. After the rectification of three images: the coordinates of the homologous points $I'_1, 1'_2$, and I'_3 satisfy $v'_2 = v'_1, u'_3 = u'_1$ and $v'_3 = u'_2$.

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.

VI. Rectification of Images

A. 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 Fig. 6.

One can show [36]–[38] 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 3×3 rectification matrices call R_1, R_2, and R_3 are defined by

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

with 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

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

illustrated by Fig. 6. There exists of course a number of degenerate cases where such a rectification is not possible. They correspond to the cases where the bottom line of matrix R_i vanishes to zero.

B. Algorithmic Complexity

The rectification of l images ($l = 2$ or 3) requires the storage of l 3×3 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 stereo vision algorithms [23], [39], [40] which actually deal with linear segments.

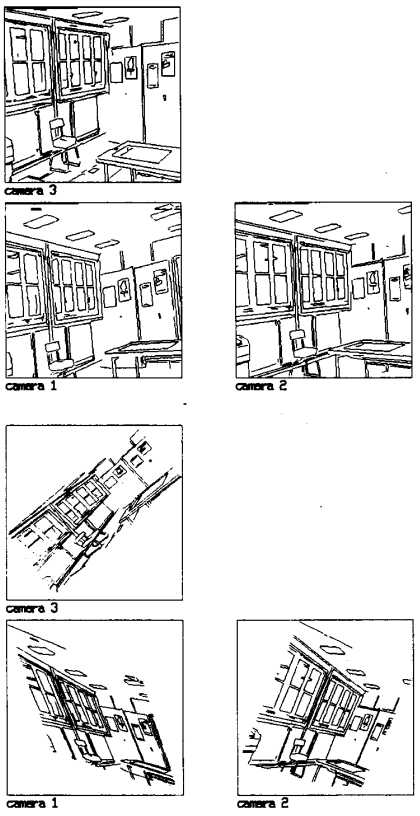

Fig. 7. Triple of linear segments of contours before and after rectification.

C. Example

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

VII. Building 3-D Segments

To build a 3-D map from trinocular stereo vision matches, one must

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

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

A. Building 3-D Lines from Their 2-D Images

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

For doing this, we used 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

[4]The symmetric parametrization is used for lines parallel to the v axis.

[5]For lines perpendicular to the zx or zy planes, a complementary parametrization is used.

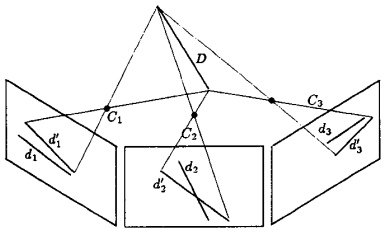

Fig. 8. Building 3-D lines from their 2-D images.

by the parameters (a, b, p, q) such that D is defined by the equations

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

We assume that the perspective transformation of each camera is represented by a 3×4 matrix T_i computed during a preliminary calibration stage [34]. If we denote by t^i_{jk} the element (j, k) in the perspective matrix T_i, then saying that the projection of D on camera i is d_i is equivalent to saying that the following two equations hold (see Appendix):

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

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

This system provides two independent 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 2-D lines, and we take them into account explicitly 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 in the form of a 4×4 symmetric covariance matrix W_D. The interested reader is referred to [37], [38], [41].

B. Computing 3-D Endpoints

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

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

This operation is repeated for the endpoints 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 corresponds to a slightly different part of the 3-D segment.

We decide to keep the 3-D segment on D which is the intersection of $A_i B_i, A_j B_j$, and $A_k B_k$. 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 3-D segment using another triplet of matches, as illustrated by Fig. 10(a). Another advantage of this solution will be explained later with the validation procedure.

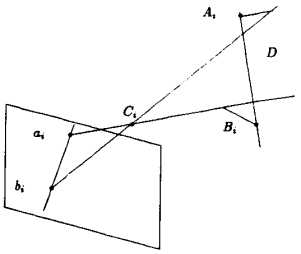

Fig. 9. Building 3-D segments from 2-D segments.

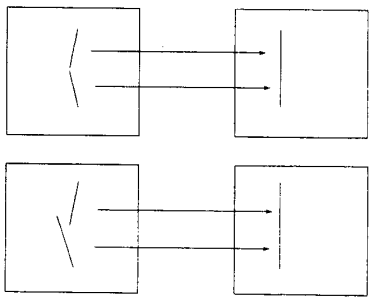

Fig. 10. Multiple matches.

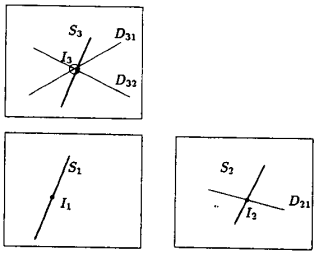

Fig. 11. Principle of trinocular stereo vision algorithm.

Finally, to compute in image i which part of the original image segment corresponds to the reconstructed 3-D segment, the endpoints of the reconstructed 3-D segment are projected onto image i. and then onto the 2-D line supporting the initial image segment.

VIII. 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.

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

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 $I_3 = D_{31} \bigcap D_{32}$, 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).

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

Fig. 12. Trinocular stereo vision algorithm.

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

A formal description of the major procedures implementing this refinements follows.

C. Main Procedure

We give in Fig. 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 midpoints 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 conjugate epipolar lines in image 1.

During the matching procedure, one of the images 2 and 3 is used to take initial hypotheses and the other to check them. As explained before, the image used for 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 Fig. 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 neighbors of each matched segment. It is described in Fig. 17, and detailed in the next section.

D. Matching Procedure

The algorithm of procedure MATCH-3 is given in Fig. 13. This procedure takes as input three images i, j, and k, a segment of S_1

△ **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 △

Fig. 13. Matching algorithm.

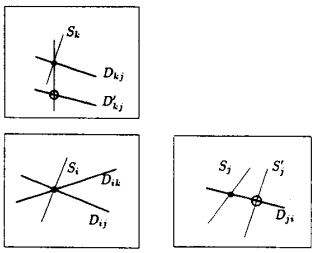

Fig. 14. Prediction of position.

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

- pass the similarity tests with S_i
- intersect the epipolar segment D_{ji}. As we have taken j such that the orientation of D_{ij} is as far as possible from 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. Fig. 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 Fig. 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} containing S_j:

$$I'_{kj} = D'_{ji} \bigcap D_{S_j}.$$

  - — Compute, in image k, the epipolar line D'_{kj} associated with I'_j.
  - — Compute the intersection I'_k, in image k , between the epipolar line D'_{ki} and D'_{kj}:

$$I'_k = D'_{ki} \bigcap D'_{kj}.$$

- The predicted orientation of S_k is then given by the orientation of $I_k I'_k$:

$$\phi_k^* = \phi(I_k I'_k).$$

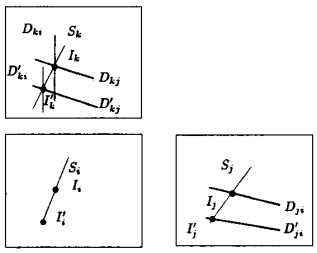

Fig. 15. Prediction of orientation.

E. Validation

Using a third camera is not always sufficient to get rid of all ambiguities and false matches. Probabilities are indeed low, but not zero, that by 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 (cf. Fig. 17):

- *The Uniqueness Constraint:* In its simplest form, it states that a given primitive must be part of at most one triplet. This 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. Fig. 10(a) shows such a case. Fig. 10(b) shows, on the contrary, an example of a wrong match violating the uniqueness constraint.

  We must therefore generalize the notion of uniqueness 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 objects are smooth, then two segments belonging to the same object and close to each other in the image will be reconstructed nearby 3-D segments. Therefore, if a match (S_1, S_2, S_3) reconstructing a 3-D segment is correct, we should be able to find some of the 2-D neighbors of the image segments S_i reconstructed into 3-D segments close to S. We cannot impose that all 2-D neighbors 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 neighbors to S be under a threshold.

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

The notion of neighborhood 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 3-D segments. It would be rigorous but inefficient from

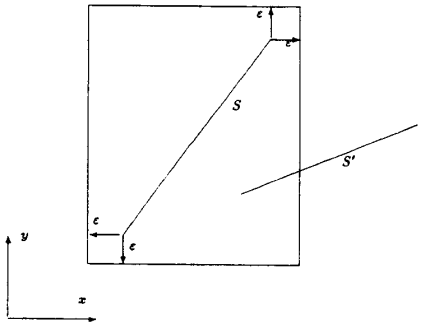

Fig. 16. S' is ϵ-compatible with S iff it intersects the box.

⊙⊙⊙ **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 ⊙⊙⊙

Fig. 17. Validation algorithm.

a computational point of view, as we do not really need a very accurate numerical criterion of compatibility. We therefore define a 3-D 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 3-D rectangle containing S, whose edges are parallel to the axes of coordinates, and extended by ϵ at each endpoint, as illustrated in Fig. 16.

Given a distance tolerance ϵ, we are now able to compute the ratio of the ϵ-compatible neighbors 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 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 favoring longer matches over shorter ones.

Recently, we improved and simplified the validation procedure by attaching a covariance matrix to each segment endpoint and computing an appropriate Mahalanobis distance. The interested reader should refer to [42].

IX. Experimental Results

We have tested this algorithm on a number of industrial and office scenes, with the following results.

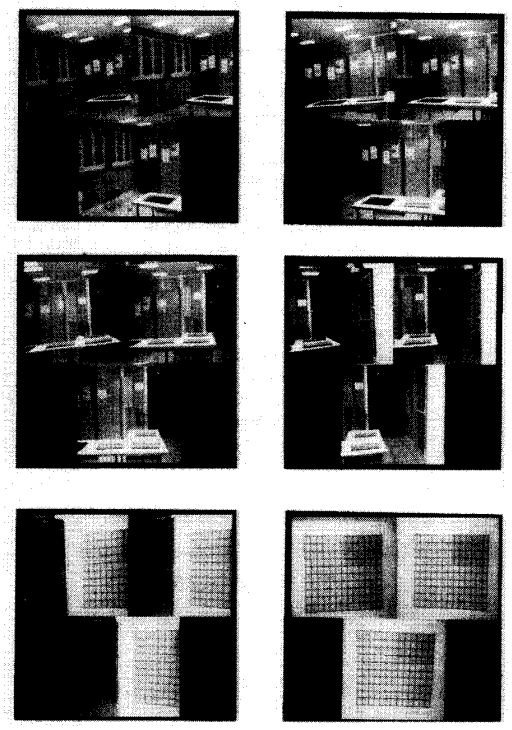

Fig. 18. Triplets of images taken in the robotics room.

A. Office Scenes

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

Fig. 20 presents the triplets of segments matched by the trinocular stereo vision program.

Using another program computing the displacement between the different frames, we are able to build the 3-D reconstruction of the whole robotics room: Fig. 21 shows the view from above, with a commented sketch of this room in Fig. 22. (Details on the construction of this global 3-D map can be found in the book by N. Ayache [43], [44] and in previous papers [45], [41].)

Table I 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.

B. Industrial Scenes

The trinocular stereo vision algorithm described above is particularly well adapted to scenes with long line segments. Nevertheless, it works also on images with curved edges, but with more computations because the polygonal approximations contain more segments. Figs. 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 Fig. 26.

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

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

It might nevertheless seem that more matchings could be done. This comes from the fact that some small segments are very noisy and cannot be predicted accurately enough in the third camera. If the

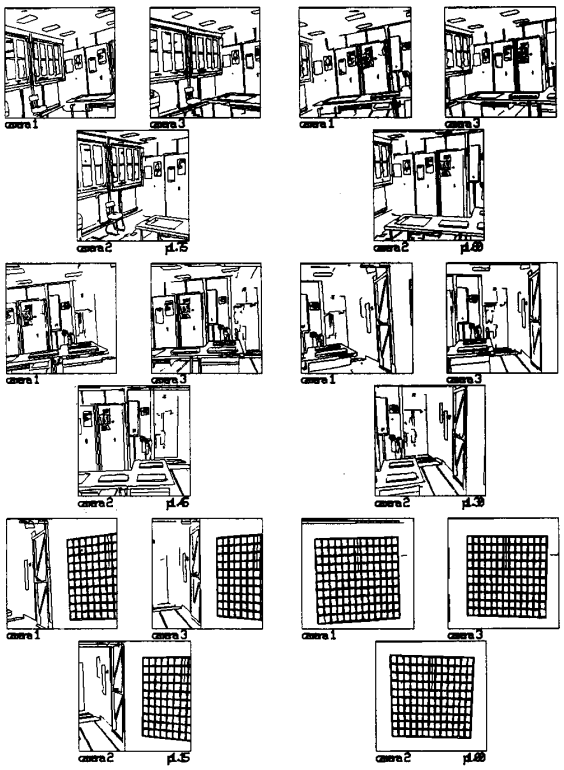

Fig. 19. Polygonal approximation of the contours.

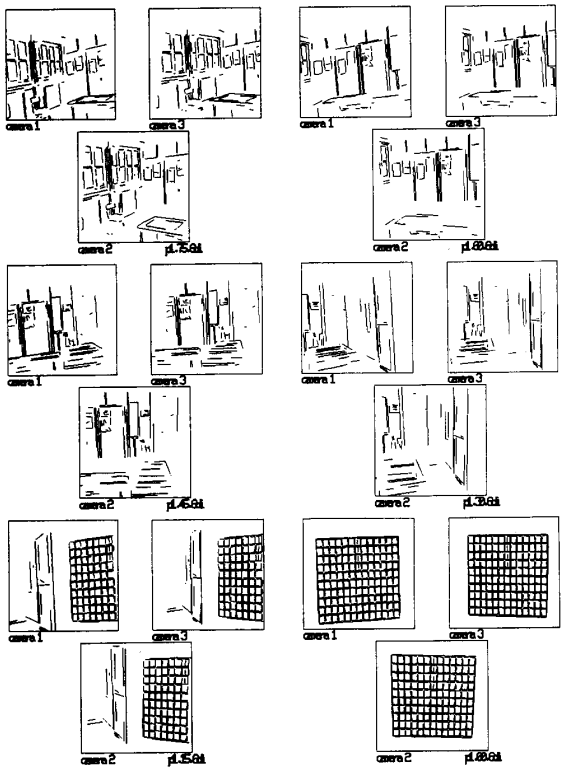

Fig. 20. Matched segments.

thresholds were relaxed so that these segments were matched, errors would be introduced elsewhere. Moreover, those segments would reconstruct very noisy 3-D segments.

It should be noted that these results allow for quantitative analyses on the observed objects. Pavel Grossmann [46] has developed a program to use the reconstructed 3-D 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 1 mm.

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

X. Conclusion

We have presented a new trinocular stereo vision 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 then 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 as follows.

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

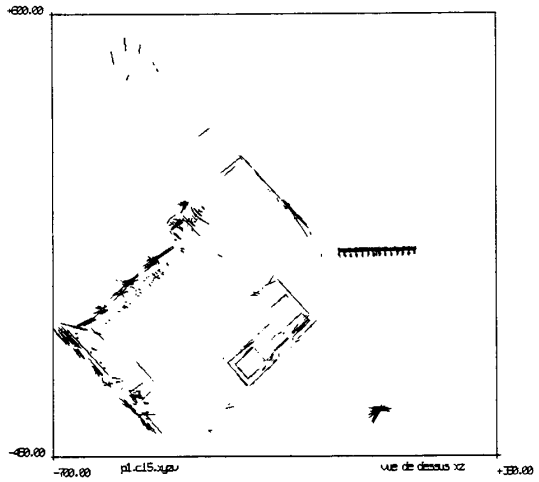

Fig. 21. View from above of the robotics room.

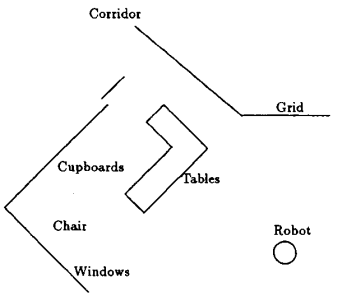

Fig. 22. Commented sketch of the view from above.

TABLE I
PERFORMANCES OF THE TRINOCULAR STEREO VISION ALGORITHM ON THE SIX INDOOR SCENES

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

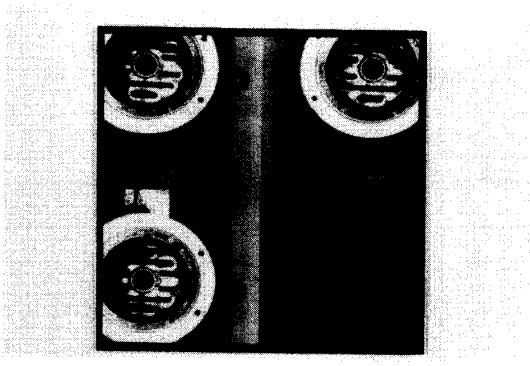

Fig. 23. Triplet of images of an industrial object (ELSAG).

Current and future developments involve the parallel implementation of the algorithm on a multiprocessor machine to perform stereo-matching and 3-D reconstruction at the rate of 5 Hz. This is done within a European Esprit Project (Project P940 involving ELSAG, GEC, INRIA, MATRA, NOESIS, University of Cambridge, University of Genova) where preprocessing (edge extraction, edge linking, polygonal approximation) is performed by dedicated hardware at the rate of 24 Hz. At the time of writing (July 1990), the production of 3-D maps of an indoor environment at the rate of 4 Hz is almost achieved [42], and will be fully described in the near future.

APPENDIX
COMPUTING A 3-D LINE D FROM ITS 2-D PROJECTIONS d_i

We assume that the perspective transformation of each camera is represented by a 3×4 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

$$u_i' = \frac{(at_{11}^i + bt_{12}^i + t_{13}^i)z + pt_{11}^i + qt_{12}^i + t_{14}^i}{(at_{31}^i + bt_{32}^i + t_{33}^i)z + pt_{31}^i + qt_{32}^i + t_{34}^i}$$

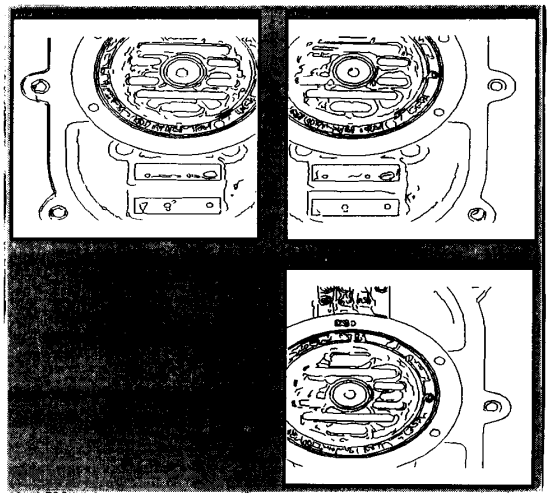

Fig. 24. Contours of a triplet of images of an industrial object.

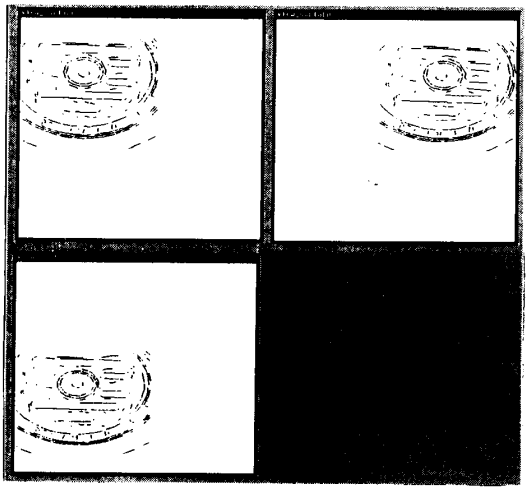

Fig. 25. Matched segments.

$$v_i' = \frac{(at_{21}^i + bt_{22}^i + t_{23}^i)z + pt_{21}^i + qt_{22}^i + t_{24}^i}{(at_{31}^i + bt_{32}^i + t_{33}^i)z + pt_{31}^i + qt_{32}^i + t_{34}^i}$$

where t_{jk}^i is the element (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:

$$\alpha_i\left(at_{11}^i + bt_{12}^i + t_{13}^i\right) + \left(at_{21}^i + bt_{22}^i + t_{23}^i\right) + \mu_i(at_{31}^i + bt_{32}^i + t_{33}^i) = 0$$

$$\alpha_i\left(pt_{11}^i + qt_{12}^i + t_{14}^i\right) + \left(pt_{21}^i + qt_{22}^i + t_{24}^i\right) + \mu_i(pt_{31}^i + qt_{32}^i + t_{34}^i) = 0.$$

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

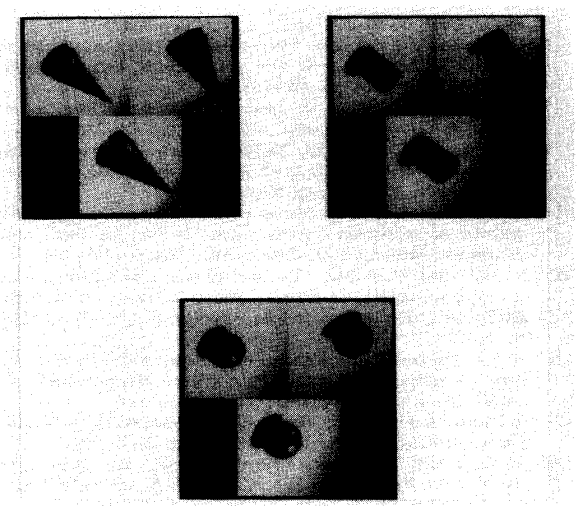

Fig. 26. Triplet of images of a cone, a cylinder, and a sphere (ELSAG).

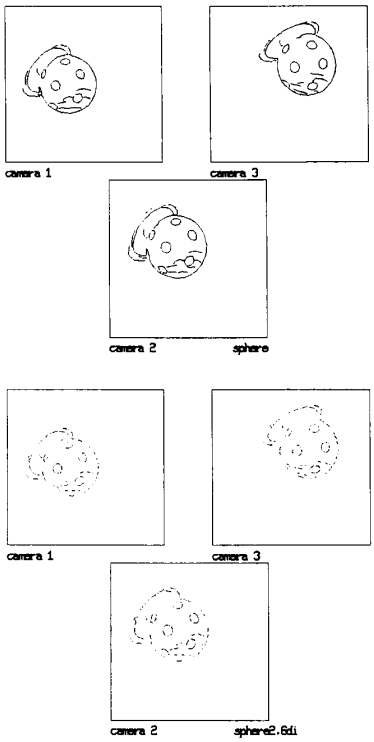

Fig. 27. Contours (a) and matches (b) of a triplet of images of a sphere.

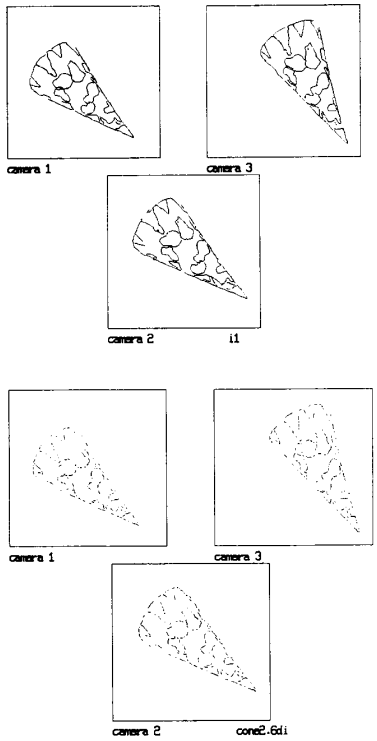

Fig. 28. Contours (a) and matches (b) of a triplet of images of a cone.

ACKNOWLEDGMENT

The stereo and 3-D 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 M. Berthod, N.-E. Deriche, and G. Giraudon, from INRIA.

The rectification scheme has been developed and improved in collaboration with C. Hansen, from the University of Utah. The current hardware implementation is performed by G. Randall at INRIA, and the software integration by S. Foret at Noesis.

Finally, we wish to thank O. Faugeras, head of the Vision and Robotics Laboratory at INRIA–Sophia Antipolis, for his constant interest and scientific support, the reviewers for their valuable suggestions, P. Sander for his careful corrections, and N. Gaudechoux for her precious help in preparing this document.

REFERENCES

[1] M. Yachida, "3-D data acquisition by multiple views," in *Robotics Research: Third Int. Symp.*, O. D. Faugeras and G. Giralt, Eds. Cambridge, MA: MIT Press, 1986, pp. 11–18.

[2] M. Yachida, Y. Kitamura, and M. Kimachi, "Trinocular vision: New approach for correspondence problem," in *Proc. Int. Conf. Pattern Recognition*, IEEE, Paris, France, Oct. 1986, pp. 1041–1044.

[3] M. Pietikainen and D. Harwood, "Progress in trinocular stereo," in *Proc. NATO Advanced Workshop Real-Time Object and Environment Measurement and Classification*, Maratea, Italy, Aug. 31–Sept. 3, 1987.

[4] M. Ito and A. Ishii, "Three-view stereo analysis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, pp. 524–531, July 1986.

[5] ——, "Range and shape measurements using three-view stereo analysis," in *Proc. Int. Conf. Computer Vision and Pattern Recognition*, IEEE, Miami Beach, FL, 1986, pp. 9–14.

[6] A. Gerhard, H. Platzer, J. Steurer, and R. Lenz, "Depth extraction by stereo triples and a fast correspondence estimation algorithm," in *Proc. Int. Conf. Pattern Recognition*, IEEE, Paris, France, Oct. 1986, pp. 512–515.

[7] E. Gurewitz, I. Dinstein, and S. Sarusi, "More on the benefit of a third eye for machine stereo perception," in *Proc. Int. Conf. Pattern Recognition*, IEEE, Paris, France, Oct. 1986, pp. 966–968.

[8] Y. Ohta, M. Watanabe, and K. Ikeda, "Improving depth map by right-angled trinocular stereo," in *Proc. Int. Conf. Pattern Recognition*, IEEE, Paris, France, Oct. 1986, pp. 519–521.

[9] M. Pietikainen and D. Harwood, "Depth from three-camera stereo," in *Proc. Int. Conf. Computer Vision and Pattern Recognition*, IEEE, Miami Beach, FL, 1986, pp. 2–8.

[10] C. Stewart and C. Dyer, "The trinocular general support algorithm: A three camera stereo algorithm for overcoming binocular matching

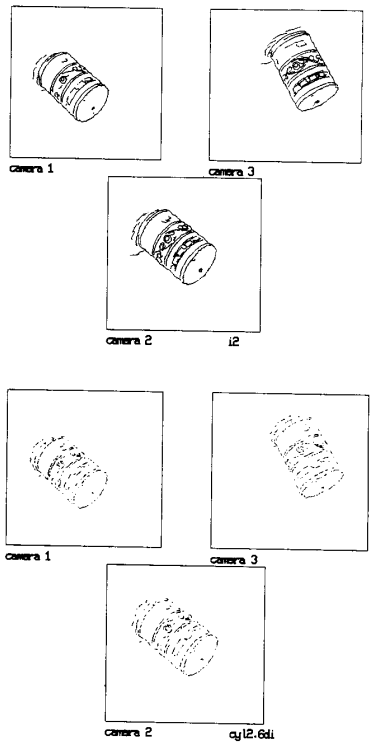

Fig. 29. Contours (a) and matches (b) of a triplet of images of a cylinder.

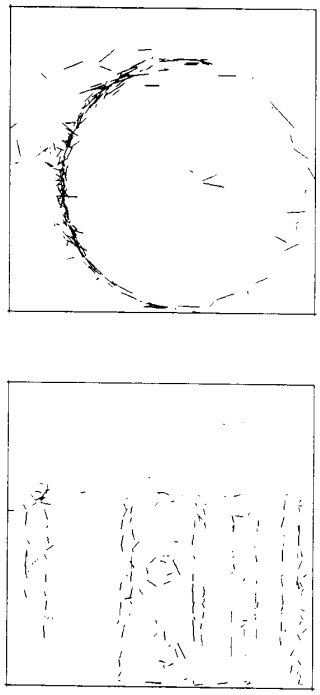

Fig. 30. View from above (a) and side view (b) of the reconstructed 3-D segments for the cylinder.

errors," Univ. Wisconsin—Madison, Tech. Rep. 768, May 1988.
[11] K. Nishihara and T. Poggio, "Stereo vision for robotics," in *Robotics Research, First Int. Symp.*, R. Paul and M. Brady, Eds. Cambridge, MA: MIT Press, 1984, pp. 489–505.
[12] D. Marr and T. Poggio, "A computational theory of human stereo vision," *Proc. Roy. Soc. London,* vol. B-204, pp. 301–328, 1979.
[13] H. Baker and T. O. Binford, "Depth from edge and intensity based stereo," in *Proc. 7th Joint Conf. Artificial Intelligence,* Vancouver, B. C., Canada, Aug. 1981, pp. 631–636.
[14] M. Berthod and P. Long, "Graph matching by parallel optimization methods: An application to stereo vision," in *Proc. Int. Conf. Pattern Recognition,* Montreal, P. Q., Canada, Aug. 1984, pp. 841–843.
[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,* Quatrième Congrès, France, 1984, pp. 71–79.
[16] W. E. L. Grimson, "Computational experiments with a feature based stereo algorithm," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. PAMI-7, no. 1, pp. 17–34, 1985.
[17] G. Medioni and R. Nevatia, "Segment-based stereo matching," *Comput. Vision, Graphics, Image Processing,* vol. 31, pp. 2–18, 1985.
[18] Y. Ohta and T. Kanade, "Stereo by intra- and inter-scanline search," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. PAMI-7, no. 2, pp. 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,* vol. 14, pp. 449–470, 1985.
[20] A. R. Saint-Vincent, "Perception et modélisation de l'environnement d'un robot mobile: Une approche par stéréovision," Thèse, Univ. Paul Sabatier, Toulouse, France, 1986.
[21] S. T. Barnard, "Stereo matching by hierarchical microcanonical annealing," in *Proc. 10th IJCAI,* Milano, Italy, Aug. 1987, pp. 832–835.
[22] T. Skordas and B. Horaud, "Stereo correspondence through feature grouping and maximal cliques," IMAG, Laboratoire d'Informatique Fondamentale et d'Intelligence Artificielle, Rapport de Recherche RR 677-I-64 LIFIA, Sept. 1987.
[23] N. Ayache and B. Faverjon, "Efficient registration of stereo images by matching graph descriptions of edge segments," *Int. J. Comput. Vision,* vol. 1, no. 2, Apr. 1987.
[24] D. J. Burr and R. T. Chien, "A system for stereo computer vision with geometric models," in *Proc. IJCAI-5,* MIT, Aug. 1977, pp. 583.
[25] D. J. Burr, "On computer stereo vision with wire frame models," Coordinated Sci. Lab., Univ. Illinois, Urbana, Rep. 805, Dec. 1977.
[26] H. H. Baker, T. O. Binford, J. Malik, and J.-F. Meller, "Progress in stereo mapping," in *Proc. Image Understanding Workshop,* Arlington, VA, June 1983, pp. 327–335.
[27] R. Jain, S. L. Bartlett, and N. O'Brien, "Motion stereo using ego-motion complex logarithmic mapping," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 9, no. 3, pp. 356–369, May 1987.
[28] D. H. Marimont, "Projective duality and the analysis of image sequences," in *Proc. Workshop Motion: Representation and Analysis,* IEEE Comput. Soc., Kiawah Island, SC, May 1986, pp. 7–14.
[29] R. Deriche, "Using Canny's criteria to derive an optimal edge detector recursively implemented," *Int. J. Comput. Vision,* vol. 2, Apr. 1987.
[30] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. PAMI-8, no. 6, pp. 679–698, 1986.
[31] G. Giraudon, "Chaînage efficace de contour," INRIA, Rapport de Recherche 605, Feb. 1987.
[32] J. G. Dunham, "Optimum uniform piecewise linear approximation of planar curves," *IEEE Trans.,* vol. PAMI-8, no. 1, pp. 66–75, Jan. 1986.
[33] G. Toscani, "Système de calibration optique et perception du mouvement en vision artificielle," Ph.D. dissertation, Paris-Orsay, France, 1987.
[34] O. D. Faugeras and G. Toscani, "The calibration problem for stereo," in *Proc. CVPR '86,* Miami Beach, FL, IEEE, 1986, pp. 15–20.
[35] O. D. Faugeras, "Artificial 3D vision," to be published.
[36] N. Ayache and C. Hansen, "Rectification of images for binocular and trinocular stereo vision," in *Proc. Int. Conf. Pattern Recognition,* Beijing, China, Oct. 1988.
[37] N. Ayache, "Construction et fusion de représentations visuelles tridimensionnelles: Applications à la robotique mobile," Thèse d'Etat, Univ. Paris-Sud, Orsay, France, May 1988; also INRIA Internal Rep.
[38] F. Lustman, "Vision stéréoscopique et perception du mouvement en vision artificielle," Ph.D. dissertation, Paris-Orsay, France, 1987.
[39] N. Ayache and F. Lustman, "Fast and reliable passive trinocular stereo vision," in *Proc. First Int. Conf. Computer Vision,* IEEE, London, England, June 1987, pp. 422–427.
[40] N. Ayache and F. Lustman, "Trinocular stereo vision, recent results," in

Proc. Int. Joint Conf. Artificial Intelligence, Milano, Italy, Aug. 1987.
[41] N. Ayache and O.D. Faugeras, "Maintaining representations of the environment of a mobile robot," in *Int. Symp. Robotics Research,* Santa Cruz ,CA, Aug. 1987.
[42] G. Randall, S. Foret, and N. Ayache, "Final steps towards real time trinocular stereo vision," in *Proc. First European Conf. Computer Vision, ECCV'90,* Antibes, France, O.D. Faugeras, Ed. New York: Springer-Verlag, Apr. 1990, pp. 601–603.
[43] N. Ayache, *Vision Stéréoscopique et Perception Multisensorielle; Applications à la robotique Mobile.* Inter-Editions, 1989.
[44] ——, *"Artifical Vison for Mobile Robots: Stereo Vision and Sensor Fusion.* Cambridge, MA: MIT Press, 1990.
[45] N. Ayache and O.D. Faugeras, "Building, registrating and fusing noisy visual maps," *Int. J. Robotics Res. (Special Issue on Sensor Data Fusion),* vol. 7, no. 6, pp. 45–65, Dec. 1988.
[46] P. Grossman, "Compact—A surface representation scheme," in *Proc. 4th Alvey Vision Conf. (AVC'88),* Manchester, England, 1988.

Shape Representation by Multiscale Contour Approximation

Ann Bengtsson and Jan-Olof Eklundh

***Abstract*—We present an approach for deriving qualitative descriptions of contours containing structures at different (unknown) scales. The descriptions are in terms of straight arcs, curved arcs with sign of curvature, corners, and points delimiting the arcs: inflexion points and transitions from straight to curved. Furthermore, the tangents at these points are derived.**

The approach is based on the construction of a hierarchic family of polygons, having the scale-space property of causality: structure can only disappear as scale goes from fine to coarse. Using the principle that structures that are stable over scale represent significant properties, the features of the descriptive representations are then derived.

***Index Terms*— Corners, hierarchic family of polygons, inflexion points, multiscale polygon approximations, qualitative contour description, scale stability, straight and curved arcs, tangent directions.**

I. Introduction

The goal of computational vision is to derive descriptions of a scene from images of it. In particular, the descriptions could be in terms of primitives representing the geometric structure of the world. There are several reasons why such descriptions are important.

In a world of coherent objects and at the level of surfaces and volumes with their bounding contours, the geometric cues given by the contours are of paramount importance. Furthermore, geometric information about, e.g., the occluding boundaries of surfaces impose very strong restrictions on the possible structure of the scene. In fact, geometric features tend to be much more useful than photometric features in the computation of what is in the scene. This seems to be true for monocular and binocular scenes as well as for time-varying scenes, see, e.g., [1]–[6].

Different approaches exist to deriving geometric structure. The work presented here should be seen in the context when the structure is reflected in image curves derived from the intensity data. The geometric properties of such curves could be obtained by direct use of the intensity information, as in the work on curve tracing and curvature by Zucker and his co-workers, see, e.g., [7], or by groupings of edge elements, see, e.g., [8].

Here we address problems appearing in the case that edges are extracted and traced, as in the work discussed, e.g., in Faugeras *et al.* [9]. Hence, we assume that there is a set of sampled contours from which we need to explicitly extract important shape information. Such structure may be straight parts of the boundaries, corners, parallelism, and symmetry. Information about curvature and direction is also important. Furthermore, surface recovery techniques may require parameterized boundaries. Finally, it is in many cases necessary to segment boundary contours into meaningful primitive parts.

Characteristic to these problems is that the data to be considered are planar curves (in the images). Moreover, these curves contain details at various levels of scale and are also contaminated by noise. There exists a need for making the information given by the curves explicitly available for further processing by deriving some abstraction of them. Such a description should be much simpler than the given representation (which is discrete) and should not be too much influenced by noise or irrelevant details in the data.

One systematic and mathematically well-founded way of finding such descriptions or simplified representations of curves and contours is to approximate them with some family of functions. We shall also propose an approach of that sort. However, there are two issues which in the standard literature on curve approximation are either not addressed or not given tractable solutions. First, most numerical techniques require that the critical points that describe the shape of the curve are given, at least as a subset of the breakpoints. If this is not the case, one might end up with hard numerical problems, e.g., in spline approximation with variable knots, which gives rise to nonlinear problems. Secondly, the methods for approximation give no hints on how the parameters should be set up to help us find the critical points. Of course, the definition of what constitutes a good description of a shape or a curve is application dependent, but certain shape features should be reliably recoverable over large parameter ranges. In particular, several of the computational tasks presented above, e.g., the check for parallelism or the computation of curvature and angles require both a smoothed and a precise description of the data. If the smoothing depends critically upon some unknown tolerance and scale parameters, the tasks become impossible without operator intervention.

In this correspondence we present an algorithm for computing shape descriptions, that addresses these issues. The method is based on multiscale approximations with lines. A crucial first step is the generation of a hierarchic family of polygonal approximations. From this *family* the descriptive shape properties are derived by analysis of the features that are stable over scale. The principle used is the principle of transformational invariance, suggesting that structures that remain invariant under a set of transformations (here: smoothings), have significance. This is related to Lowe's ideas about nonaccidentalness, [8], but has no probabilistic component. As a final postprocessing step we also show that the polygons can be converted into a spline-representation. Although visually pleasing, the latter representation does not add any significant information about the abstract shape in our present framework.

II. On Earlier Work and Our Approach

If one wants to derive descriptions of planar curves such that geometric properties of the type mentioned above are explicitly represented, one is faced with two goals of a conflicting nature. First there is a need for finding a qualitative description of overall shape. Hence some simplification and/or smoothing must take place. This goal is important for recognition and for finding global structure. Secondly, there is a need for high precision detection of certain characteristics. Earlier we mentioned straight line segments, corners,

Manuscript received March 9, 1987; revised July 12, 1990. Recommended for acceptance by C. Brown. This work was supported by the National Swedish Board for Technical Development (STU).

The authors are with the Computational Vision and Active Perception Laboratory, Royal Institute of Technology, S-100 44 Stockholm, Sweden.

IEEE Log Number 9040025.