Using local planar geometric invariants to match and model images of line segments

Patrick Gros, Olivier Bournez, Edmond Boyer

Using Local Planar Geometric Invariants to Match and Model Images of Line Segments

Patrick Gros Olivier Bournez Edmond Boyer
Gravir - Cnrs - Inria Rhône Alpes

Matching Line Segments Using Local Planar Invariants

Gravir - Cnrs - Inria Rhône Alpes
ZIRST - 655, avenue de l'Europe - 38330 Montbonnot - France
Patrick.Gros@imag.fr

Using Local Planar Geometric Invariants to Match and Model Images of Line Segments

Abstract

Image matching consists of finding features in different images that represent the same feature of the observed scene. It is a basic process in vision whenever several images are used. This paper describes a matching algorithm for lines segments in two images. The key idea of the algorithm is to assume that the apparent motion between the two images can be approximated by a planar geometric transformation (a similarity or an affine transformation) and to compute such an approximation. Under such an assumption, local planar invariants related the kind of transformation used as approximation, should have the same value in both images. Such invariants are computed for simple segment configurations in both images and matched according to their values. A global constraint is added to insure a global coherency between all the possible matches: all the local matches must define approximately the same geometric transformation between the two images. These first matches are verified and completed using a better and more global approximation of the apparent motion by a planar homography, and an estimate of the epipolar geometry.

If more than two images are considered, they are initially matched pairwise, then global matches are deduced in a second step. Finally, from a set of images representing different aspects of an object, it is possible to compare them, and to compute a model of each aspect using the matching algorithm.

This work uses in a new way many elements already known in vision: some of the local planar invariants used here were presented as quasi-invariants by Binford and studied by Ben-Arie in his work on the Peaking Effect. The algorithm itself uses other ideas coming from the geometric hashing and the Hough algorithms. Its main limitations come from the invariants used. They are really stable when they are computed for a planar object or for many man-made objects which contain many coplanar facets and elements. On the other hand, the algorithm will probably fail when used with images of very general polyhedrons. Its main advantages are that it still works even if the images are noisy and the polyhedral approximation of the contours is not exact, if the apparent motion between the images is not infinitesimal, if they are several different motions in the scene, and if the camera is uncalibrated and its motion unknown.

The basic matching algorithm is presented in section 2, the verification and completion stages in section 3, the matching of several images is studied in section 4 and the algorithm to model the different aspects of an object is presented in section 5. Results obtained with the different algorithms are shown in the corresponding sections.

1 Introduction

Image matching is one of the most difficult basic problems in vision. It appears whenever several images are considered. For example, consider the reconstruction problem: from two images of a three-dimensional object, compute the

geometry (i.e. shape) of the object. Two problems must be solved, *matching* and *reconstruction*:

1. given a point of the object, find its projection in each image;
2. once both projections are known, compute the position of the object point.

The first problem is usually specified differently: given an image point m_1, find the corresponding point m_2 in the other image and the object point M of which they are both projections. The process of finding corresponding points m_1 and m_2 is called image matching (to distinguish it from image-model matching).

Many papers have been published in the past on the image matching problem. Many methods deal with grey level images: in this case, the algorithm tries to match every pixel of both images. Each pixel contains a value, the grey level, which encodes the light intensity received in one point of the sensor plane. These methods can be classified in three groups, as in [1].

Correlation: these use cross-correlation to measure the similarity of two image regions. Given a region in the first image, they search for the most similar region in the second image. Constraints arising from the camera geometry can be used to limit the size of the search space: epipolar geometry is a good example. Other limitations may be purely algorithmic: for example, the regions considered must have borders parallel to the image axis. Such methods have a long history in photogrammetry [2, 3] and computer vision [4]. Improvements are regularly published [5, 6, 7, 8, 9, 10].

Relaxation: in these methods, the first few matches are guessed. Constraints derived from these are then used to compute further matches. This process is repeated until no new matches can be found. For example, see Marr and Poggio [11, 12], Grimson [13, 14], and Pollard et al. [15, 16].

Dynamic programming: in this case, the matching problem is formulated as the minimization of a function of many discrete variables. Examples can be found in [17, 18].

When higher level features are available in images, such as edge points or line segments, the matching problem is slightly different. Such features are less numerous but richer in information; in particular, they have geometric characteristics that can be used to aid matching. They are also usually more reliable than grey level values. On the other hand, it is no longer possible to obtain dense matches or reconstructions with such features. Another difficult problem is relevance: a curved shape represented by line segments is hard to match because the definition of the segments is very instable.

Several methods adapted to such features have been proposed.

Prediction and verification: these methods are similar to relaxation techniques. Some initial matches are guessed, then constraints are deduced

from them, that allow the guesses to be verified and further matches to be found [19, 20].

Subgraph isomorphism: adjacency relations between image tokens are modeled as the edges or vertices of a graph, and standard methods for finding subgraph isomorphisms can be applied. These methods demand that there be little noise in the images, and use no geometric information. In addition, their complexity is often very high. Several examples can be found in [21, 22].

Geometric invariants: for images obtained by perspective projection, projective invariants can be computed to characterize - and hence to match - point and line configurations [23, 24]. Various invariants have been suggested [25, 26, 27, 28]. Until now, these methods have mainly been used for planar or very simple scenes.

Quasi-invariants : some authors also use quantities which are not exactly invariant under the projection mapping used to model the camera. These quantities are usually invariants of a more restrictive set of transformations, but remain "almost invariant" for "typical" camera displacements. Binford [29] defines quasi-invariants geometrically as measurements that are equal to a quantity computed in the scene for at least one particular transformation, and that are constant to first order when the transformation is perturbed (i.e. the first order term of the Taylor expansion of the quasi-invariant with respect to the transformation vanishes). If the second order term also vanishes, the quasi-invariant is said to be strong.

Binford proves that the angle between two segments, and the length ratio of these segments are quasi-invariants. We will use these two quantities in the present paper. He also proves that the affine coordinates defined in a plane in the scene are strong invariants. We will also use affine coordinates, but in the image rather than in the scene. These affine coordinates may correspond to non coplanar points in the scene, and are not quasi-invariants.

By design, quasi-invariants are stable for non degenerate projective transformations. This explains why they can be used successfully for matching and recognition tasks.

The peaking effect: Ben-Arie [30] studied the probability distributions of angles and length ratios of two segments from random viewing directions, and he noticed that "the probability density of the ratio of the projected angle to the original angle has a sharp peak at the unity ratio, that is to say, at the point where the projected angle equals the original angle. A similar peaking phenomenon appears in the probability density curve of the ratio of distances; the peak is at the point where the ratio of distances in the image equals the ratio of the depicted distances in the scene... the probability that a projected angle will have values between double and half of the original angle is more than 84%! The same span of proportions with regards to projected distances has a probability of more than 86%!"

He uses his statistical measurements to compute a metric for the comparison of scene and image ratios and angles. This is used in a relaxation method in his recognition system. [31, 32] describe other recognition systems based on the peaking effect. The same statistical distance is used, but within a prediction-verification scheme, where pose estimation of the object is used as verification.

In our method, we benefit fully from the peaking effect and we use the same kind of features: angles and ratios. But we use the peaking effect in a quite different way. The three previous papers describe image-3D model matching techniques, while we have an image-image matching technique. We do not compare angles computed from a 3D model to image angles, but only angles in two different images. Furthermore our system is not based on relaxation or prediction-verification, but on geometric hashing and Hough transform.

Geometric hashing: Lamdan and Wolfson [33, 34, 23] have developed an image matching method based on affine local invariants. In each image, they consider all triplets of points as affine frames, and compute the affine planar coordinates of all the points in all these frames. Then they try to find which pair of frames in different images give the same coordinates to the points. If a pair is found, the match is done. The same method is used when a new image is to be matched to an image database. In this case the system finds the database image having the best correspondence with the new image.

Lamdan and Wolfson justify the use of affine planar invariants by practical considerations. Projective invariants would have given too combinations and the similarities are sometimes too poor to describe apparent motion. Their system gives good results, but remains very computation intensive.

Our system has many similarities to the peaking effect and geometric hashing methods discussed in the literature. A more detailed comparison is presented at the end of paragraph 2.1.

Our approach. The method proposed in this paper is based on line segments, possibly linked one to the other at their endpoints. The key idea of the algorithm is to assume that the apparent motion can be approximated by a geometric transformation, i.e. a similarity or an affine transformation. Under such an assumption, planar invariants associated with simple segment configurations can be computed and matched between the different images. To insure a global coherency between all the possible matches, a global constraint is used: all the matches must define approximately the same apparent motion. The matches are then verified and completed using a better approximation of the apparent motion by a planar homography and an estimate of epipolar geometry.

The invariants we use are planar algebraic invariants. They are not necessarily invariant under perspective projection, but they are invariant between

images whenever the apparent motion can be approximated by a planar geometric transformation. They are known to be stable when computed from images of planar objects. Man-made objects usually contain many planar surfaces: the algorithm will work with them.

On the other hand, the algorithm can fail with very general polyhedral objects, but such objects are barely used in usual life. For curved objects, the tools and invariants used should be very different (see [35]).

If several images are considered, they are matched pairwise and global matches are deduced in a second stage. If the images represent different aspects of an object, the matching algorithm can be used as a measurement of similarity between images, and the images representing a same aspect can be grouped, and give rise to a 2D model of this aspect.

The main advantages of our method are the following:

1. It needs no knowledge of the camera calibration, motion or epipolar geometry. The apparent motion is not assumed to be very small, so the method is usable in cases where tracking methods would fail.

2. Objects in the scene can have independent motions. Results with two different motions are shown at paragraph 3.4.3.

3. It works with real noisy images. In such images, the polyhedral approximation of the contours is often inexact and some segments are unstable. This is particularly true if the scene contains curved objects. The topology of the graph formed by the segments is known to be very unstable. Using local quasi-invariants makes our method robust against this.

4. The use of geometric quasi-invariants associated with a few common segment configurations substantially reduces the number of possible matches, so the complexity is much smaller than that of subgraph isomorphism techniques.

The main limit of the method comes from the approximation of the apparent motion. The validity of this approximation is quite clear for planar objects according to Ben-Arie and Binford, but it is hard to determine the exact set of 3D objects for which the algorithm will not fail.

The paper is organized as follows. Section 2 describes the matching algorithm for two images. Section 3 shows how these initial matches can be improved and completed using global constraints. Section 4 considers the case of several images and section 5 shows how to go from image matching to object modeling.

2 Matching two images

2.1 Matching algorithm

This section considers the case where two images of line segments are to be matched. Such images are usually obtained from grey level images by edge

extraction [36, 37], and polygonal approximation of the edge lines. Segment extremities may be used as junctions between segments, and each image may be seen as a graph of segments and vertices. FIG. 3 shows some examples. The problem is to match the segments and their extremities between images.

It is clear that corresponding features do not have the same pixel coordinates in the two images. The coordinate difference is called apparent motion.

Apparent motion is not a 2D geometrical transformation. For example, two scene points can give the same projection in one image and different projections in the other one. However, for generic views of nearly coplanar ensembles of features, the apparent relative motion is well approximated by a 2D transformation. Our system uses 2D similarity or affine models to approximate the relative motion of small ensemble of nearby line segments and points.

The stages of the algorithm are as follows:

1. One of the two kinds of approximation (similarity or affine) is chosen, and local invariants are calculated for the salient feature configurations: angles and length ratios for similarities, and affine coordinates for affine transformations.

2. These invariants are matched between the two images according to their values and to thresholds derived from the noise level.

3. When the invariants of two configurations match, the similarity or affine transformation between the configurations is computed.

4. The transformations are represented as points in a parameter space: four parameters for similarities (two translations, one rotation, and one scaling factor), and six parameters for affine transformations (four parameters for the linear part and two for the translation).

5. Correct matches define transformations close to the best approximation to the apparent motion, so points in the parameter space tend to cluster in a small region. On the other hand, false matches are not correlated with one another and their corresponding points are scattered across the parameter space. Thus we search for a small region of parameter space where the point accumulation is maximal. This allows us to compute an approximation to the apparent motion and to separate correct matches from false ones.

6. Invariant matches give rise to configuration matches, and feature matches are deduced from these. In case of ambiguity, only the most probable matches are considered as correct: for example, if a feature a is matched 5 times to feature b, and only once to feature c, the match (a, b) is considered to be correct.

The stages are explained in detail below, for similarities and then for affine transformations. Finally, we show some experimental results. But first we compare our method with geometric hashing and peaking effects methods.

Comparison with Wolfson's geometric hashing. Geometric hashing, as presented in [33, 34, 23], deals with affine planar invariants. Points are extracted from both images. Each triplet of points is considered as an affine basis and the affine coordinates of all the other points are computed in these frames. The coordinates obtained in different frames in each image are then compared to see if any two frames correspond. If this is the case, point matches may be deduced directly from the comparison.

The main similarities with our method are the use of local planar invariants which are not object invariants, and the use of a more global approximation to the apparent motion to constrain the matching.

But there are important differences: in geometric hashing no topological structure is used, while we use the segments and their adjacency to reduce the combinatorics. If the approximation of planar apparent motion is not global valid, it is very often quite accurate for nearby points linked by segments. Such points are often linked by segments or edges on the object itself, frequently corresponding to the borders of planar facets, for which a planar affine or similarity transformation is often a robust approximation to the true apparent motion.

A second difference is the method used to compare invariants. In geometric hashing there is a global comparison of the invariants, for each triplet in both images in all the possible frames in both images. This is very computation intensive. Our methods compares local invariants pairwise. After that, we use a Hough technique to find sets of coherent matches, and finally to compute the best approximation to the apparent motion. Furthermore, if objects in the scene have different motions, these correspond to different apparent motions in the images. While it is easily possible to search for two such motions in our Hough space, the comparison is much more difficult with geometric hashing.

As a conclusion, our method use several ideas also present in geometric hashing. But other methods, like the use of a Hough technique, are applied at crucial points of the algorithm and allow us to avoid the weaknesses of geometric hashing.

Comparison with peaking effect methods. Our method also shares some assumptions and ideas with the peaking effect methods. We use the same features, angles and ratios, so we benefit from the properties demonstrated in Ben-Arie's paper, but our context is quite different. We use no 3D model or data, and our matching method is based on a voting technique and requires no verification step like localization. This allows us to match against several different images in parallel, and to achieve image recognition without having to consider each possible image individually. This is also the case with geometric hashing. A recognition system based on our matching process is presented in [38].

2.2 Approximation by a similarity

Similarities are compositions of isometries and scalings. We will consider only direct similarities, i.e. compositions of rotations, translations and scalings.

These transformations define an "extended" Euclidean geometry (since Euclidean classically excludes scalings).

The two basic invariants of similarities are the angle and the length ratio defined by any two segments. Clearly, these are not invariants of the 3D object. But they are not too far from invariance: Binford has shown that they are quasi-invariants [29].

The peaking effect provides a statistical explanation of their usefulness [30]. Experimentally, matching using angles and length ratio does give good results, even if they seem at first sight to be relatively poor and coarse image features.

2.2.1 Stage 1: computing invariants

Similarity invariants may be computed from two segments or three points. To reduce the complexity of the configuration enumeration and to increase the validity of the planar approximation, only pairs of segments having an endpoint in common are considered. Let S_1 and S_2 be two such segments, P_0 be their common endpoint, and P_1 and P_2 be their other endpoints (see FIG. 1).

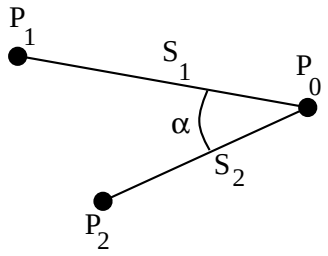

Figure 1: Invariant of two adjacent segments.

The pair is characterized by its angle α, length ratio ρ, and a "reliability weight" p.

$$\alpha = \arccos \frac{\overrightarrow{P_0P_1}.\overrightarrow{P_0P_2}}{\|\overrightarrow{P_0P_1}\|.\|\overrightarrow{P_0P_2}\|} \qquad \rho = \|\overrightarrow{P_0P_1}\|/\|\overrightarrow{P_0P_2}\| \qquad p = \|\overrightarrow{P_0P_1}\| + \|\overrightarrow{P_0P_2}\|$$

By convention, S_1 and S_2, and P_1 and P_2, are chosen so that the oriented angle $(\widehat{S_1S_2})$ is acute and counterclockwise. This simplifies the comparison of invariants and configurations. The weight factor is justified by the fact that noise and texture usually give short segments, whereas long segments are often more significant and characteristic. The weights are used mainly in the filtering stage of the algorithm.

2.2.2 Stage 2: matching invariants

To compare the configuration invariants of two images, we use two thresholds $\alpha_{\max}$ and $\rho_{\max}$. They are chosen with regard to the image noise, typically $\alpha_{\max} = 20°$ and $\rho_{\max} = 1.5$. A pair of segments in the first image with

invariants (α_1, ρ_1), is matched to a configuration in the second image with invariants (α_2, ρ_2), if three conditions are satisfied:

$$|\alpha_1 - \alpha_2| < \alpha_{\max} \qquad \rho_2/\rho_1 < \rho_{\max} \qquad \rho_1/\rho_2 < \rho_{\max} \tag{1}$$

At this stage, a configuration is often matched to 10 - 20 of other configurations.

In the implementation, the comparison process is sped up by sorting the invariants in each image by α. The lists obtained may be compared by a linear rather than a quadratic enumeration.

2.2.3 Stage 3: computing similarities

Only a small proportion of all the matches found in the previous stage are correct. To separate these from the incorrect matches, we use the fact that correct matches correspond to the same apparent motion and thus define some approximate similarities very close to that motion.

A single matched pair of configurations provides enough information to compute the similarity between its members. Indeed, each pair provides three point matches and two are sufficient to compute a direct similarity.

Let us consider two pairs of matched segments, with extremities (P_0, P_1, P_2) and (Q_0, Q_1, Q_2). The parameters of the similarity which transforming the first configuration into the second one may be computed as follows:

$$k = \frac{1}{2}\left(\frac{\|\overrightarrow{Q_0Q_1}\|}{\|\overrightarrow{P_0P_1}\|} + \frac{\|\overrightarrow{Q_0Q_2}\|}{\|\overrightarrow{P_0P_2}\|}\right)$$

$$\theta = (\widehat{\overrightarrow{P_0P_1}\,\overrightarrow{Q_0Q_1}}) + \frac{1}{2}\left[(\widehat{\overrightarrow{Q_0Q_1}\,\overrightarrow{Q_0Q_2}}) - (\widehat{\overrightarrow{P_0P_1}\,\overrightarrow{P_0P_2}})\right]$$

$$\begin{pmatrix} t_x \\ t_y \end{pmatrix} = \overrightarrow{PQ_0} \quad \text{where} \quad P = H_k \circ R_\theta(P_0)$$

R_θ represents a rotation by angle θ about the origin, and H_k represents a scaling by k centered at the origin. The translations t_x and t_y are based only on the common vertices P_0 and Q_0, because these are often more reliable than the external vertices (which may be linked to no other segment). The similarity computed with this method provides an apparent motion approximation for the two configurations it is computed from.

2.2.4 Stages 4 and 5: filtering similarities

For each pair of matched configurations, a similarity is computed using the previous method. Each similarity is defined by its four parameters (t_x, t_y, θ, k), and may be represented by a point in $\mathbb{R}^4$. Each point is weighed by the sum of the weights of the configurations used to compute the similarity.

Incorrect matches are generally not correlated one with another, so their similarities are distributed over a large region of $\mathbb{R}^4$. On the other hand, correct matches all define good approximations to the apparent motion and give points clustered in a small region of the parameter space. The aim of this stage is to search for this region.

To do this, we define a distance between similarities:

$$\mathrm{d}\left((t_x,t_y,\theta,k),(t'_x,t'_y,\theta',k')\right) = \frac{(t'_x-t_x)^2}{\lambda_x^2} + \frac{(t'_y-t_y)^2}{\lambda_y^2} + 2\left(k^2+k'^2-2kk'\cos(\theta'-\theta)\right)$$

where λ_x and λ_y are the size of the image along x and y axis respectively, and for each similarity $(t_x^0, t_y^0, \theta^0, k^0)$ we compute the sum:

$$\sum_{(t_x,t_y,\theta,k)} \frac{1}{0.5+\mathrm{d}\left((t_x,t_y,\theta,k),(t_x^0,t_y^0,\theta^0,k^0)\right)}$$

The similarity with the greatest sum is considered as the best approximation of the apparent motion. Only matches which give rise to a similarity close to this one are considered to be correct.

To limit the complexity of this computation, the implementation sorts the similarities in $\mathbb{R}^4$ according to their first coordinate, and only the similarities such that:

$$|t_x - t_x^0| < t_x^{max} \qquad |t_y - t_y^0| < t_y^{max} \qquad |\theta-\theta^0| < \theta^{max} \qquad k^0/k^{max} < k < k^0.k^{max}$$

where t_x^{max}, t_y^{max}, θ^{max} and k^{max} are four thresholds, are considered in the sum. The threshold values are typically 15 pixels, 15 pixels, 20 degrees and 1.5.

2.2.5 Stage 6: matching features

Each matching pair of configurations (P_0, P_1, P_2) and (Q_0, Q_1, Q_2) suggests three possible point matches: (P_0, Q_0), (P_1, Q_1), (P_2, Q_2). For each point, all of the suggested matches are considered and the match that occurs most often is taken to be correct. The other matches and the configurations matches from which they are deduced are suppressed. For example, given four point matches (P, Q_1), (P, Q_1), (P, Q_1), (P, Q_2), the match between P and Q_1 is assumed to be correct. The match (P, Q_2) and the configuration match from which (P, Q_2) was deduced are suppressed.

2.3 Approximation by an affine transformation

Now consider the case where the apparent motion between the images is approximated by an affine transformation, rather than by a similarity. The basic invariant of affine geometry is the length ratio of parallel segments. The above approach may be extended to affine transformations by using invariants based on parallel length ratios in place of Euclidean invariants.

As above, and as in other geometric hashing methods, these invariants are not object invariants. They are just invariants for the kind of apparent motion we assume. If this assumption is not valid at all, the method will fail. On the other hand, a difference with the extended Euclidean case is that affine invariants are projective quasi-invariant only if the corresponding 3D structure is planar (the same remark holds for the invariants used in geometric hashing).

Before matching, and without prior information, it is impossible to know which sets of segments correspond to planar structures.

The main justification for using affine invariants is that affine planar transformations provide a much wider set of transformations, that permits a better approximation of the apparent image motion.

In this section, we discuss only the differences between the similarity approximation and the affine approximation; the overall approach remains the same.

2.3.1 Stage 1: computing invariants

Three segments or four points are needed to compute an affine invariant, in the general case where the points or segments are not collinear. To limit the complexity, we consider only two kinds of configurations, called Z or Y configurations.

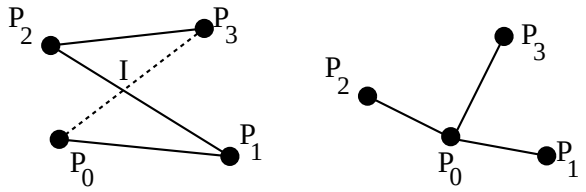

Figure 2: Definition of invariants from a Z (left) or Y (right) three-segment configuration.

For a Z configuration, such as that shown in FIG. 2, two invariants are defined.

$$\rho = \|\overrightarrow{P_3 I}\| / \|\overrightarrow{P_0 I}\| \qquad \text{and} \qquad \sigma = \|\overrightarrow{P_2 I}\| / \|\overrightarrow{P_1 I}\|$$

To reduce ambiguity, the configuration points are labeled so that ρ is greater than or equal to 1.

In the case of Y configurations, the invariants are the affine coordinates of P_0 with respect to the three other points P_1, P_2 and P_3. These coordinates are defined as follows:

$$\begin{cases} a_1 P_1 + a_2 P_2 + a_3 P_3 = P_0 \\ a_1 + a_2 + a_3 = 0 \end{cases}$$

The configuration points are labeled in such a way that a_1 is smaller than a_2 and a_2 is smaller than a_3. In both cases, the configuration is weighed by the sum of length of the three segments.

2.3.2 Stage 2: matching invariants

The affine matching process follows exactly the same algorithm as for similarities. Only the matching conditions (1) are different. For two Z configurations with invariants (ρ_1, σ_1) and (ρ_2, σ_2), the conditions are:

$$\rho_1 < k\rho_2 \qquad \rho_2 < k\rho_1 \qquad \sigma_1 < k\sigma_2 \qquad \sigma_2 < k\sigma_1$$

For two Y configurations with invariants (a_1, a_2, a_3) and (b_1, b_2, b_3), they are:

$$|a_1 - b_1| < \varepsilon \qquad |a_2 - b_2| < \varepsilon \qquad |a_3 - b_3| < \varepsilon$$

In our experiments, k was taken equal to 2.2 and ε to 1.5.

2.3.3 Stage 3: computing affine transformations

Each configuration match provides four point matches. From these, it is possible to compute the affine transformation T that transforms the first configuration into the second one, using a least square method.

Let P_i^j be the i-th point of the j-th configuration and $(x_i^j y_i^j 1)^t$ be its homogeneous coordinate vector. If $\begin{pmatrix} a_1 & a_2 & t_x \\ a_3 & a_4 & t_y \\ 0 & 0 & 1 \end{pmatrix}$ is the homogeneous matrix associated with T, we have the following equation:

$$\begin{pmatrix} x_1^2 & x_2^2 & x_3^2 & x_4^2 \\ y_1^2 & y_2^2 & y_3^2 & y_4^2 \\ 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} a_1 & a_2 & t_x \\ a_3 & a_4 & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1^1 & x_2^1 & x_3^1 & x_4^1 \\ y_1^1 & y_2^1 & y_3^1 & y_4^1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

which can also be be written in the following form:

$$\begin{pmatrix} x_1^2 \\ x_2^2 \\ x_3^2 \\ x_4^2 \end{pmatrix} = \begin{pmatrix} x_1^1 & y_1^1 & 1 \\ x_2^1 & y_2^1 & 1 \\ x_3^1 & y_3^1 & 1 \\ x_4^1 & y_4^1 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_1 \\ a_2 \\ t_x \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} y_1^2 \\ y_2^2 \\ y_3^2 \\ y_4^2 \end{pmatrix} = \begin{pmatrix} x_1^1 & y_1^1 & 1 \\ x_2^1 & y_2^1 & 1 \\ x_3^1 & y_3^1 & 1 \\ x_4^1 & y_4^1 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_3 \\ a_4 \\ t_y \end{pmatrix}$$

The least square resolution of these systems, for example using the Singular Value Decomposition [39], provides as a solution the transformation T that minimizes the expression $\sum_{i=0}^{3} \|P_i^2 - TP_i^1\|^2$. We chose this solution T as the best approximation to the apparent motion for the two considered configurations.

2.3.4 Last stages

Proceedings as for the similarities, affine transformations are represented as points in $\mathbb{R}^6$, and filtered in order to separate the correct matches from the false ones. Feature matches are then deduced from configuration matches and voted on. To compare affine transformations, we use the following distance:

$$\mathrm{d}\left(\begin{pmatrix} a_1 & a_2 & t_x \\ a_3 & a_4 & t_y \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} a'_1 & a'_2 & t'_x \\ a'_3 & a'_4 & t'_y \\ 0 & 0 & 1 \end{pmatrix}\right) = \sum_{i=1}^{4} (a_i - a'_i)^2 + \frac{(tx - t'x)^2}{\lambda_x^2} + \frac{(t_y - t'_y)^2}{\lambda_y^2}$$

and compute for each transformation the sum:

$$\sum_{(t_x, t_y, \theta, k)} \frac{1}{2.5 + \mathrm{d}\left((t_x, t_y, \theta, k), (t_x^0, t_y^0, \theta^0, k^0)\right)}$$

2.4 Algorithm evaluation

2.4.1 Complexity

The most time consuming operation in the algorithm consists of comparing tuples up to a threshold ε, and of doing an elementary operation for all the pairs of close elements. If n is the number of tuples, the complexity of the operation is $O(n \log n)$ when $\varepsilon = 0$ and $O(n^2)$ when $\varepsilon \to \infty$. Let us denote $O\left(f_\varepsilon(n)\right)$ that complexity in the general case.

In order to compute the complexity of the algorithm, we can consider that there are approximately the same number of segments and extremities in both images. Let n be that number. As it is uncommon to have more than 4 segments with one common extremity, the computation of all invariants may be done with a $O(n)$ complexity in time. Their comparison may be done in $O\left(f_{\varepsilon_1}(n)\right)$.

At the next step, m transformations are computed. In the worst case, m is $O(f_{\varepsilon_1}(n))$. The comparison of the transformations can be done in $O\left(f_{\varepsilon_2}(m)\right)$. The last stage of the algorithm has a lower complexity.

Then, the global complexity is $O\left(f_{\varepsilon_2}(f_{\varepsilon_1}(n))\right)$, with:

$$O(n \log^2 n) \leqslant O\left(f_{\varepsilon_2}(f_{\varepsilon_1}(n))\right) \leqslant O(n^4)$$

The geometric hashing algorithm is based on the comparison of all the quadruples of points of two images, and have a complexity of $O(n^4)$. It should be noted that geometric hashing is an algorithm designed for indexing rather than matching. In the case of indexing, a complete match is not necessary and the complexity can be reduced.

	T	1	2	3	4	5	6	7	8	9
Image 1	S	33	29	53						
Image 2	S	34	27	62	402	91	403	12	18	17
Image 3	S	136	142	173						
Image 4	S	102	106	131	2648	487	1223	13	21	25
Image 5	S	306	266	485						
Image 6	S	312	276	491	26241	3960	17988	33	62	78
Image 1	A	33	29	101						
Image 2	A	34	27	128	5119	1292	9894	30	21	20
Image 3	A	136	142	264						
Image 4	A	102	106	199	16677	2998	8790	26	22	22
Image 5	A	306	266	906						
Image 6	A	312	276	920	31276	5625	20499	5	15	20

Table 1: Experimental evaluation of the matching algorithm complexity.

Results of an experimental evaluation of the algorithm complexity are shown on table 1. For each experiment, the columns shows the number of: segments in the image (1), extremities in the image (2), invariants in each image (3), comparisons done between invariants (4), transformations computed

(5), distances computed between transformations (6), transformations considered as correct (7), segments matched (8) and extremities matched (9). The column T indicates the type of transformations used: S for similarities, A for affine transformations. These results show clearly that the complexity remains much smaller from $O(n^4)$!

2.4.2 Experimental results

In this section, our matching algorithm is illustrated with some results obtained using real images: this allows the assumptions and approximations made to be validated.

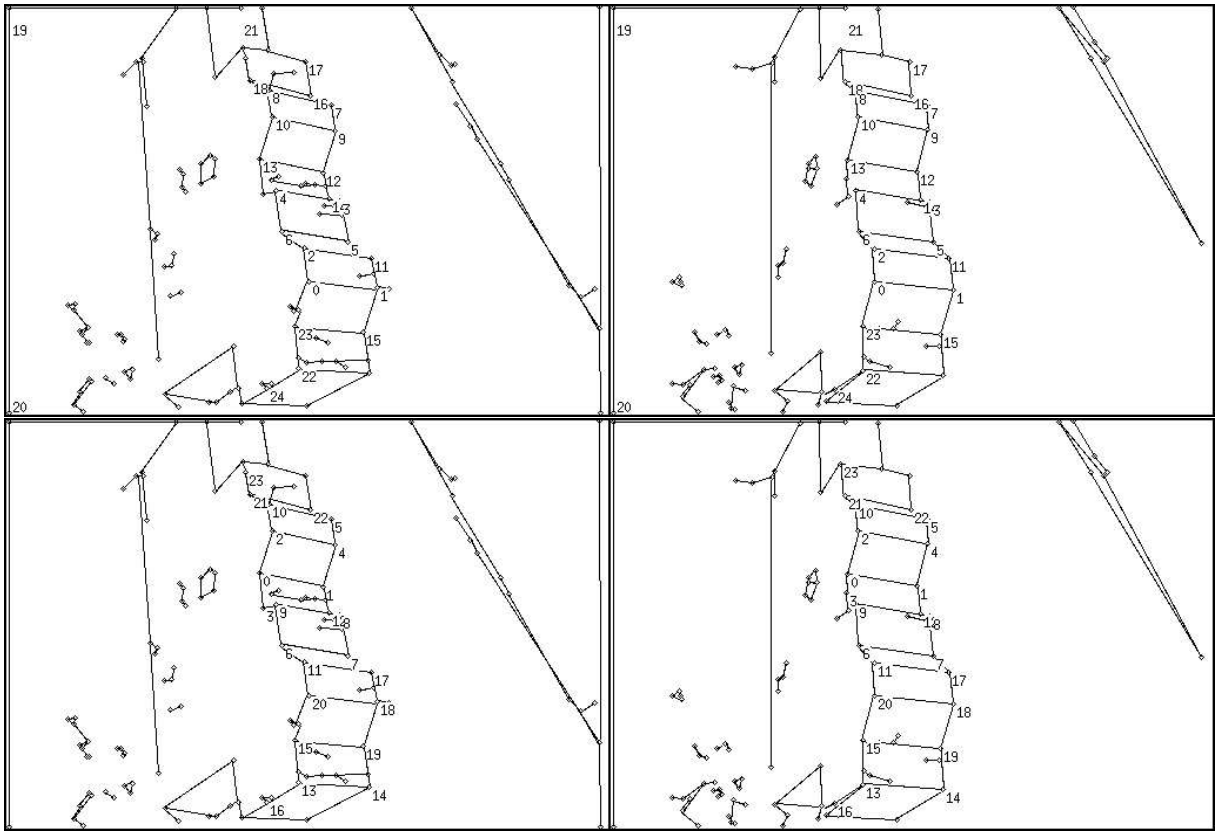

Figure 3: A first example of matching, using similarities (top), and affine transformations (bottom).

FIG. 3 shows the result of matching two images. Matched points have the same number. The similarity-based method was used for the two upper images, and the affine method for the two lower images. (There is no correspondence between the numbers in the upper images and the numbers in the lower ones).

The left image contains 133 points, the right one 106. The main object contains 67 points in the former, 53 in the latter. The others are due to noise, background or texture. 25 matches are obtained using similarities , and 24 using affine transformations. All of the matches are correct, although, in the case of similarities, two of them correspond to the image border. On the other

hand, some points clearly belonging to the object are not matched. This is due to imperfect segmentation (for example segments split in two by a vertex), or to regions where the apparent motion is not well approximated by a similarity or even an affine transformation (due to perspective effects for example).

Clearly, planar local invariants are sufficiently robust for reliable matching, even when the apparent motion is visibly far from an exact similarity or an affine transformation.

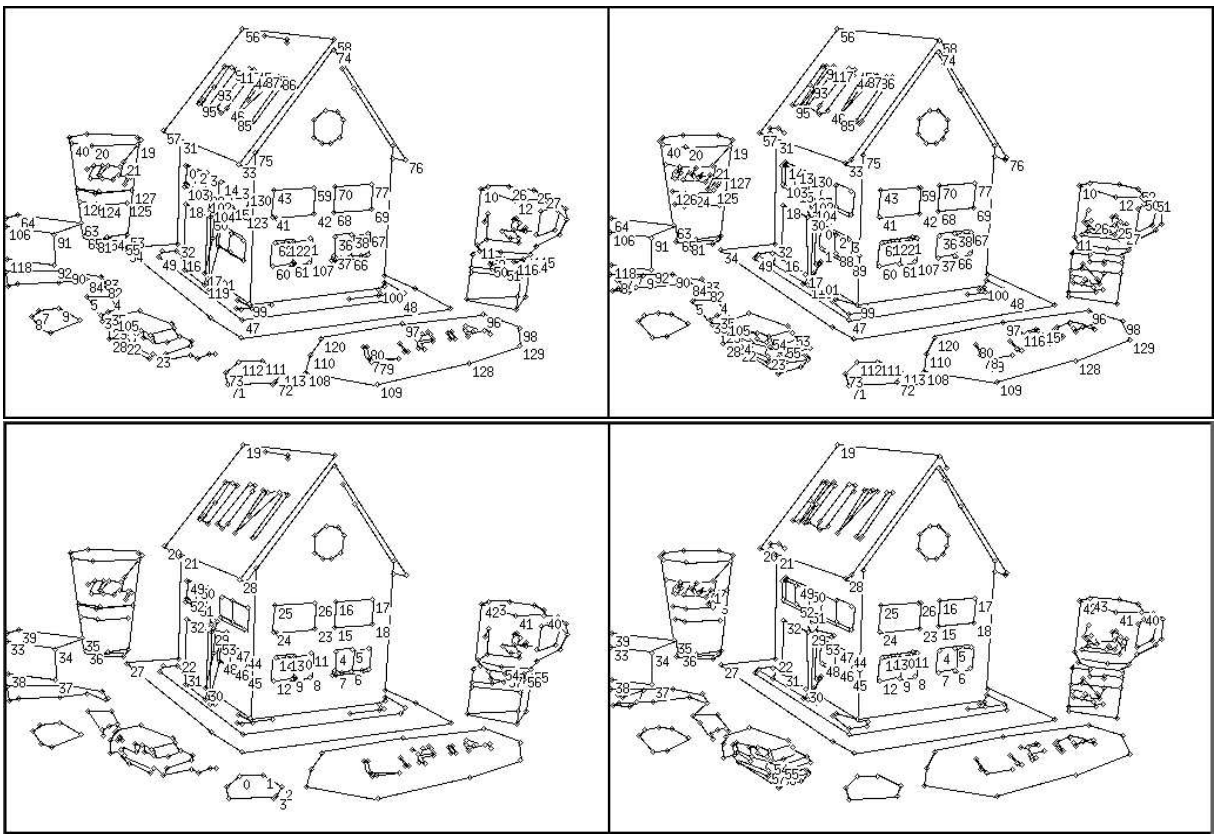

Figure 4: Another example of matching using similarities (top) and affine transformations (bottom).

FIG. 4 shows similar results for another pair of images These images contain non polyhedral objects, for which the approximation of edges by segments is unstable and very noisy. The first image contains 322 points, the second 352. About 80 points belong to the house. With similarities, 131 points were matched, with 15 outliers, and 15 errors at the windows: the windows are represented by repetitive rectangles and errors often occur in such situations. With affine transformations, 58 points are matched, with 8 outliers and 4 errors due to the windows.

2.5 Conclusion

The results obtained show the validity of the algorithm, especially its ability to deal with real noisy images containing polyhedral and even curved objects.

Of course, the matching obtained is not complete, and the next section shows how these initial results can be improved.

The method appears to be general and not specific to the particular kinds of local invariants used. To demonstrate this, we used other kinds of invariants, such as Euclidean invariants computed from three segments. The results are similar.

Another idea would be to use projective invariants, computed using cross ratios. In this case, the problem is that the computation of such invariants and of the corresponding homographies between configurations is very sensitive to noise. This makes the search for an accumulation point in $I\!R^8$ extremely difficult. Our conclusion was that projective invariants are only useful for matching when the data is very precise.

As a conclusion, we can say that our algorithm is robust, and that its robustness comes from the use of simple local planar invariants. They appear to be a powerful tool to deal with images where noise forbids to use exact and more sensitive invariants. They make a big part of the originality and of the success of our method.

3 Improving image matching

The algorithm described in the previous section provides an initial match between two images without any prior information. The results may contain a few incorrect matches and miss correct ones: apparent motion approximation by a similarity or an affine transformation is too restrictive for some image regions.

However, once an initial match is found, other finer tools can be used to evaluate the validity of computed matches, and to find new ones. A first tool is the approximation of apparent motion by a 2D homography. When the observed object is planar, this is not an approximation, but an exact computation. A second tool is epipolar geometry. This only gives point to line correspondences rather than point to point ones, however it is exact for both planar and non planar objects. The computation and use of these two tools are described in this section.

These tools are particularly interesting when the scene contains an object moving against a fixed background, as the mobile object will fail to define the same epipolar geometry or homography as the rest of the scene. This opens new directions for applications of the matching method.

3.1 Projective approximation of apparent motion

As similarities or of an affine transformations may not approximate the apparent motion well over large image regions, a first idea is to use a transformation with more parameters. In the Klein hierarchy [40], the next class after the affine transformations is the projective ones (homographies). At least four points are necessary to define a homography between two images. The matching algorithm described in the previous section provides, in general, more than

four correspondences, and this redundancy allows the computation instability described in the conclusion 2.5 to be suppressed.

Several methods are available to compute a homography between two images from at least four matches.

3.1.1 Linear least squares method

The first and simplest method is linear least squares as in stage 3 of paragraph 2.3. Let (P_i^1, P_i^2) be the matches obtained previously, and T be the homography which minimizes $\sum_i \|P_i^2 \wedge (T.P_i^1)\|^2$, with homogeneous matrix $\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$. If the equations $P_i^2 \propto T.P_i^1$ are written as it was done in paragraph 2.3, T can be obtained by solving the system of equations by singular value decomposition.

3.1.2 Robust least median squares method

Real images are always noisy and some false matches always fall within the thresholds of the matching algorithm. The least squares method uses all the matches, including the errors. A few errors are sufficient to cause inaccuracy of the computed transformation, even at the points given to compute it. A least median squares method can correct this drawback by detecting outliers. The algorithm is the following:

1. Four point matches are chosen randomly.
2. The homography T defined by these four matches is computed using the linear method.
3. The errors $\|P_i^2 \wedge (T.P_i^1)\|^2$ are computed for all of the remaining matches, and their median ε is found and associated with T (more generally, the x%-ian error can be used for $x \neq 50$).

This process is repeated sufficiently many times that the probability of choosing at least one set of four correct matches is very high. The final estimation of the homography is the transformation T whose associated value ε is the smallest.

This method allows outliers in the data to be detected. Any matches giving an error three times greater than the median (or twice greater than ε if x is greater than 50), may be considered to be outliers and eliminated. The best estimate of the homography is independent of these outliers, and turns out to be good enough for many applications. For a more accurate result, it is possible to refine the estimate using the non linear method below.

The number of times the three stages 1, 2, and 3 must be repeated and the percentage x depend on the the fraction of outliers in the data. x must be less than the estimate rate of outliers in the data. 50% may be taken as a default value, but the result is of course better with a better estimation of the real rate.

If y is the real fraction of false matches in the data, the probability of choosing at least one configuration of n correct matches in m trials is [41]:

$$P = 1 - [1 - (1 - y)^n]^m$$

If we assume that $y = 40\%$ and we want $P > 0.99$, 34 iterations are sufficient. As indicated in [41], this method can be improved by requiring that the chosen points are well distributed in the images. According to our experimental results, the additional time this requires is hardly justified by the improvement of the result.

3.1.3 Non linear optimization method

The two previous methods are not symmetric with respect to the images. Two images play different roles in the criterion $\sum_i \|P_i^2 \wedge (T.P_i^1)\|^2$ to be minimized, and this may have consequences on the result.

This expression may be symmetrized:

$$\sum_i \left(\|P_i^2 \wedge T.P_i^1\|^2 + \|P_i^1 \wedge T^{-1}.P_i^2\|^2 \right)$$

A linear resolution is no longer possible. The results of the least mean or median squares methods may be used as an initial estimate of the solution in a non linear optimization method, such as that of Levenberg-Marquardt [39].

The linear solution usually provides a good enough approximation to the real solution to insure convergence within a few iterations. On the other hand, the elimination of outliers by the least median squares method seems necessary to obtain accurate results in most cases. The non linear method is sensitive to the quality of the initial estimate of the solution: this is yet another argument for the use of the robust method rather than that of the linear one.

3.1.4 Conclusion

In our experiments, we used the least median squares method to remove outliers. It gives much better results than the simple linear method. On the other hand, our data are not precise enough to have a real improvement of the results when using the non linear optimization method. In all cases, homographies work best for scenes that are not too deep.

3.2 Epipolar geometry computation

Epipolar geometry is the geometric relation between two images of the same scene, and is summarized by the fundamental matrix F. This 3×3 rank 2 matrix verifies ${}^tP^2.F.P^1 = 0$ for every couple of matched points (P^1, P^2), represented by their homogeneous coordinates. Many methods of computation of this matrix from point matches have been proposed in the literature. They may be classified in three groups:

- linear methods, which allow epipolar geometry to be computed directly from point matches;

- robust methods, which allow outliers to be detected and eliminated.
- non linear methods, which provide very accurate results but require an initial estimate of the solution.

Here are short descriptions of the main methods:

Simple linear method: this considers the problem as a simple system of linear equations. The fundamental matrix F is defined up to a scale factor and one of its coefficient may be taken equal to 1. The matrix is then found by solving the equations ${}^tP^2.F.P^1 = 0$, for at least eight different point matches, using the SVD method. This method has a few drawbacks: the system to be solved is ill-conditioned and the rank the computed matrix is often equal to 3 as soon as the data are not exact.

Hartley's method: the previous method may be significantly improved in several ways [42]: first, changing the coordinate basis allows to correct the ill-conditioning of the initial system. Second, setting the smallest singular value of the linear solution to zero guarantees that the obtained matrix is singular.

Boufama's method: a new method was recently proposed [43]. Based on a different formulation of the problem, it allows a singular fundamental matrix to be found by a simple resolution. For planar scenes, the two previous methods provide meaningless results, while Boufama's method provides correct pencils of epipolar lines: every point of every epipolar line has its corresponding point on the corresponding epipolar line.

Least median squares method: as with the homography computation, it is possible to use a robust method based on the least median squares. This method may be improved in several ways, concerning the way the points are chosen, or the linear method used ... Such improvements are described in [41].

Non linear optimization method: the first method could provide a non singular matrix. To correct this, the criterion $\sum_i \| {}^tP_i^2.F.P_i^1 \|^2$ may be minimized under the constraint $\det F = 0$. This may be done by using a non linear optimization algorithm such as Levenberg-Marquardt [39].

The main drawback of this method is the instability of the function $F \mapsto \sum_i \| {}^tP_i^2.F.P_i^1 \|^2$, which admits many local minima; the correct solution, i.e. the one that corresponds to the true camera motion, is not always even the global minimum of the function as soon as the images are noisy. To use this method, one must eliminate the outliers with the robust method first, and give a very good first estimation of the result.

For our matching purpose, we generally use the least median squares method with the linear Boufama's method. They provide robust epipolar pencils which can be efficiently used for matching. On the other hand, the epipoles are not used and their position need not be determined precisely.

3.3 Corrections and improvements

The tools described above allow the results obtained with the matching algorithm presented in the first section of the paper to be significantly improved.

- Computing epipolar geometry or approximating the apparent motion with a homography using robust methods allows outliers to be detected. The mean-error found when computing the homography may be used to decide whether the scene is planar or not. In the former case, Boufama's method or the robust method associated with Boufama's one should be used to compute the epipolar geometry. In the latter case, the epipolar geometry should be used since the homography cannot accurately approximate the apparent motion.
- New matches may be deduced from those already found. Let s_1 be a segment whose endpoints are a_1 and a_2. If only one of these, say a_1, is matched to a point b_1 in the other image, we can search for another point b_2 in this image, such that there exists a segment between b_1 and b_2 and a_2 and b_2 are compatible with respect to epipolar geometry or the homography approximating the apparent motion. If neither a_1 nor a_2 are matched, it is possible to look for a segment s_2 whose extremities b_1 and b_2 respect the epipolar and homography constraints with a_1 and a_2.
- Finally, we can search among all the unmatched points in both images for all the couples (a, b) that satisfy the epipolar and homography constraints. However, this may give incoherent results with respect to image topology and should be used with considerable care.

3.4 Experimental results

3.4.1 Improved matches

FIG. 5 and 6 show results obtained from the matches shown in FIG. 3 and 4: the epipolar geometry and the homography were computed using a robust method, and the outliers were detected and eliminated. FIG. 7 and 8 show the results obtained after the improvement stage.

It is of course possible that new errors are introduced during this last stage. However they must be at least approximately correct from a geometrical viewpoint, because they must respect the two constraints.

On the other hand, some points are still not matched. This is mostly because the homography gives a good approximation to the apparent motion only in part of the image. In the remaining part, it suppresses any new matches.

The performance of these stages are summarized in Tab.2. The columns Object 1 and Object 2 concern the images of FIG. 3, while the columns House 1 and House 2 concern the images of FIG. 4. Each entry of the first line is the number of points in the corresponding image. The following lines provide the number of points matched using the matching algorithm described in section 2 and the number of false matches, the number of remaining matches and errors after the outliers detection stage, and the final number of matches and errors.

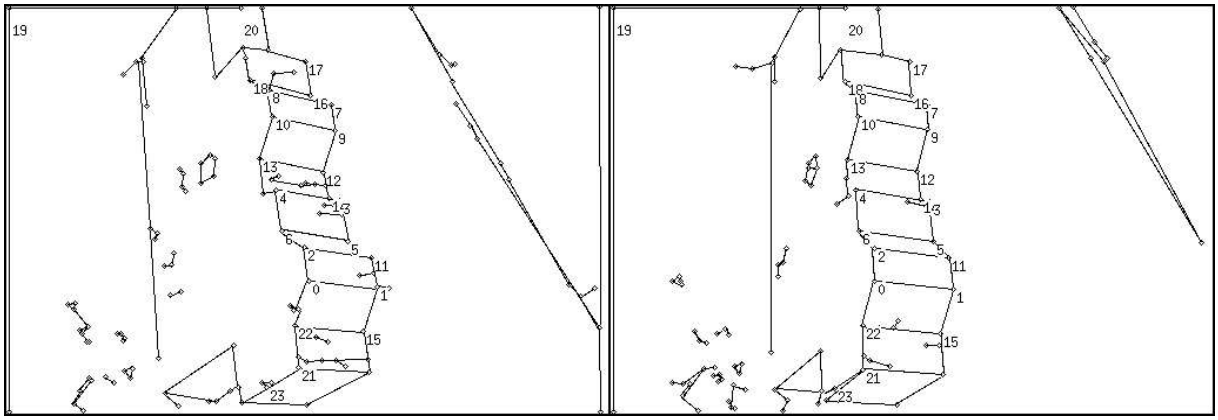

Figure 5: The matches of FIG. 3 after error suppression.

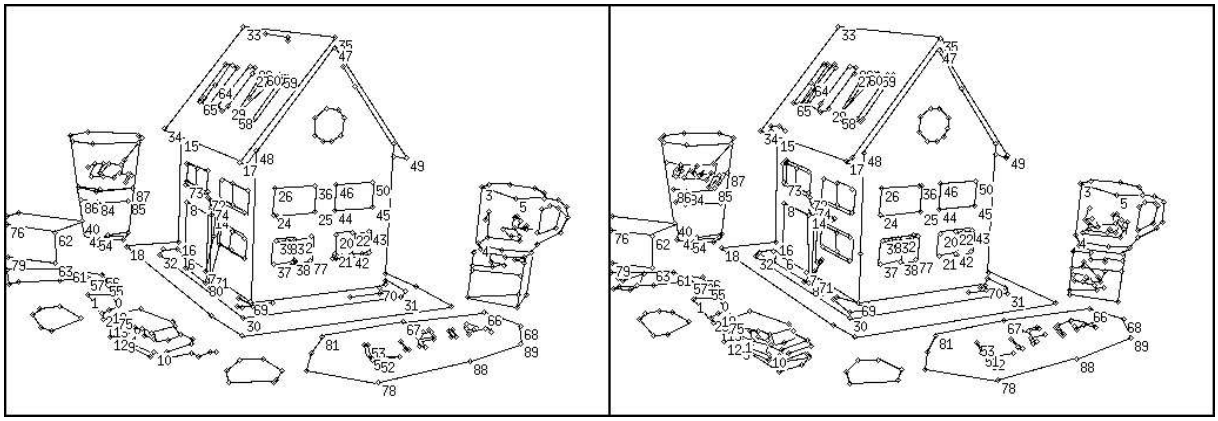

Figure 6: The matches of FIG. 4 after error suppression.

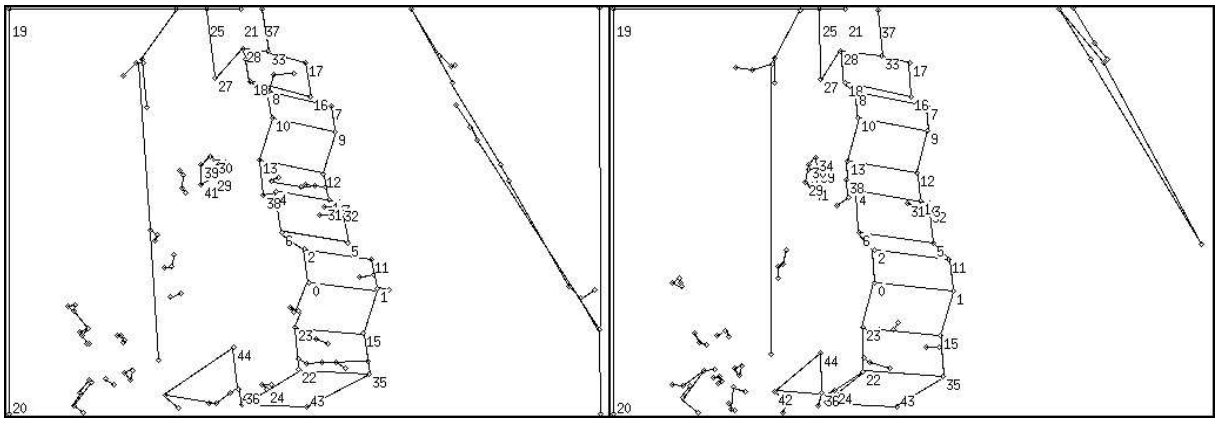

Figure 7: Improvement of the matches shown on FIG. 3

Images	Object 1	Object 2	House 1	House 2
Number of image points	133	106	322	352
First matches	25		131	
Number of errors	0		30	
Remaining matches	24		90	
Remaining errors	0		0	
Final matches	45		262	
Final errors	1		12	

Table 2: Results obtained with the improvement algorithms.

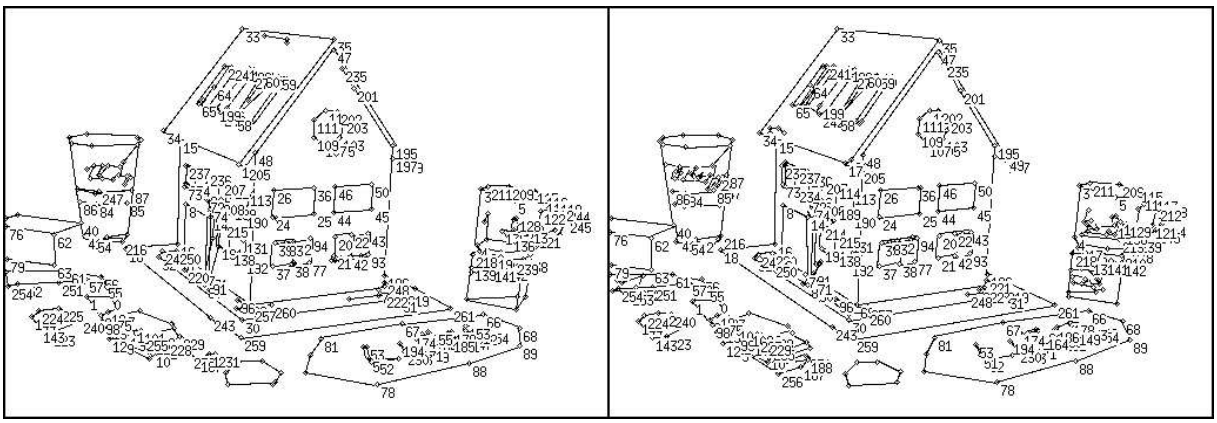

Figure 8: Improvement of the matches shown on FIG. 4

3.4.2 Limits of the algorithm

A good way to judge an algorithm is to study the cases where it just fails. An important point in our algorithm is the approximation of the apparent motion by a planar geometric transformation. The results presented here concern the limit of this approximation.

The image 0 of the sequence was matched with all the other images. Table 3 shows the number of point matches and the number of errors obtained with different methods: simple match using similarities (SS), improved math using similarities (IS), simple match using affine transformations (SA), and improved match using affine transformations (IA) respectively.

Image 16 and image 9 correspond to the limit of deformation acceptable for the algorithm when using similarities and affine transformations respectively.

Another limit of the algorithm is due to the numerous thresholds: these thresholds are necessary when comparing invariants or transformations because of the noise present in the image. They also allow to reduce the complexity. Their choice was done empirically, after a long use of the algorithm on many different images. A bad choice of the thresholds may explain bad results.

Image	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Nb Matches SS	27	40	24	27	19	20	20	21	14	15	10	9	7	11	6	7
Nb Errors SS	0	0	0	0	3	3	0	3	3	0	3	0	0	0	0	0
Nb Matches IS	105	96	99	89	102	99	68	72	47	52	55	82	49	46	56	51
Nb Errors IS	2	5	3	4	3	4	7	8	8	12	8	12	15	12	15	19
Nb Matches SA	31	26	21	20	24	14	14	13	14	10	8	4	6	9	8	8
Nb Errors SA	2	0	2	0	0	1	0	0	0	6	4	4	6	9	8	4
Nb Matches IA	100	102	91	67	82	64	59	56	51	14	8	51	0	15	1	3
Nb Errors IA	1	1	6	9	11	9	10	11	4	10	1	13	0	15	1	0

Table 3: Number of points matched and of errors when matching image 0 with the image of a sequence.

3.4.3 Images with moving objects

All of the above tools may be mixed and used with different aims. In the example presented here, two images are taken from slightly different viewpoints, but two objects have also moved between the two shots.

These images are matched and the match is improved using the tools presented above. The result obtained after the first stage is shown in FIG. 9.

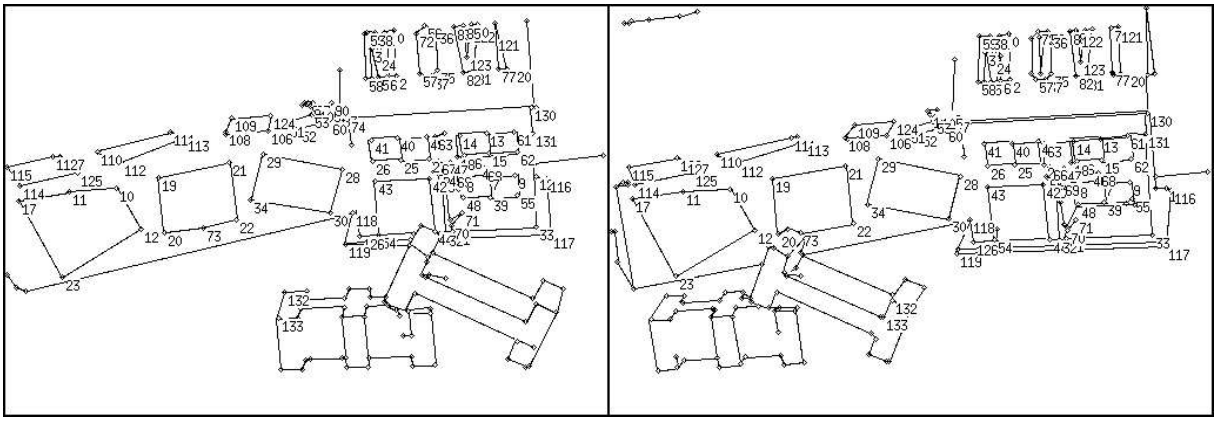

Figure 9: Result of the first matching.

In the second stage, we display only the segments that were not matched and their extremities. These remaining segments are matched, and the match is improved. The result is shown in FIG. 10.

This example shows the ability of our method to deal with mobile objects in a scene. In the first run, the objects in the background were matched. As a consequence of the use of a global constraint on the apparent motion, the mobile objects were not matched. But when the matched features are removed, i.e. most of the background, the mobile objects can be matched.

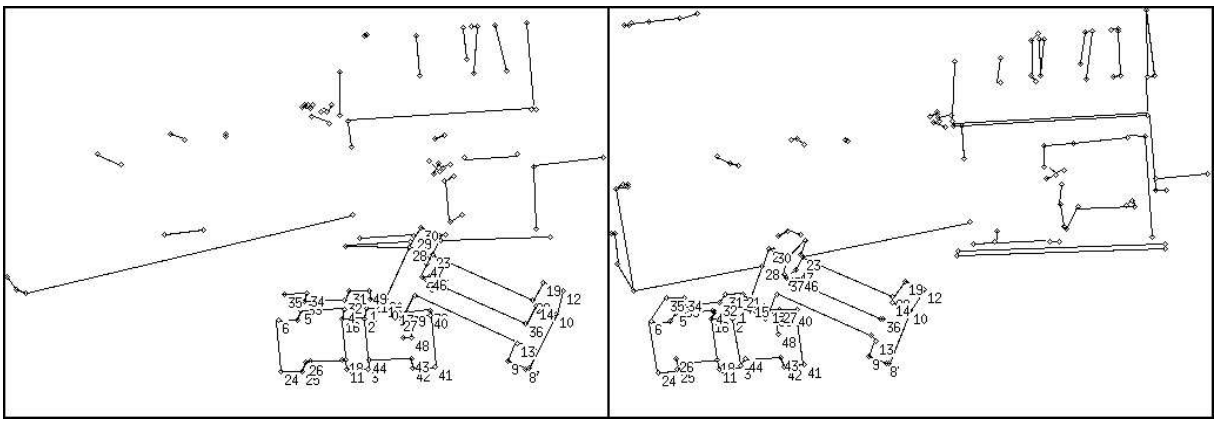

Figure 10: Result of the second matching.

4 Matching more than two images

The remainder of the paper is devoted to the case where more than two images must be matched. To accomplish this, we propose an algorithm in which the images are first matched pairwise using the methods described above, and then a global match is deduced from these partial ones. The algorithm and a few experimental results are presented in the first paragraph of this section.

4.1 Algorithm principle

Let us assume that we want to match n images, given in no particular order. In a first step, they are matched pairwise as above. The matches may be summarized as a "match-graph": the graph's vertices are the image features, and its edges are the feature matches.

From this graph, we want to compute a hyper-graph which has the same vertices and whose hyper-edges represent global matches. Another way to state this problem is to search for a partitioning of the match-graph, such that each of the obtained components is a global match.

4.1.1 A first simple method: connectedness

A first simple idea consists of finding the connected components of the match-graph. By itself, this method gives bad results: any error links two components that should have remained separate and thus has a catastrophic effect. If more than ten images are involved, the biggest components typically contain at least ten to twenty features of each image.

FIG. 11 shows a part of the match-graph of a set of images. The features of all images were ordered such that features having the same number should go in the same global match. Correct matches are shown with continuous lines and false matches with broken ones. The three errors in the figure are enough to merge the first four global matches, and it is clear that every error can merge two such components and thus has a catastrophic effect.

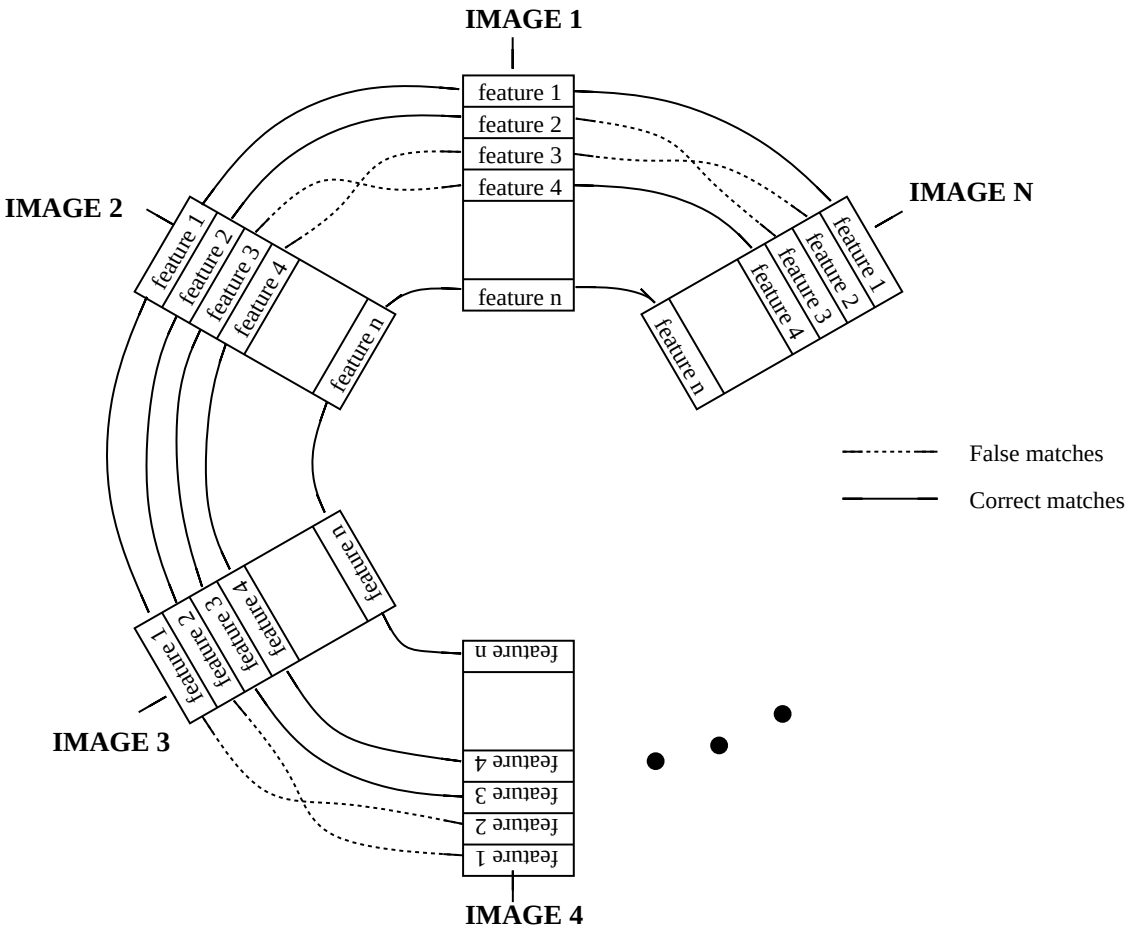

Figure 11: When considering connected components, every error has a catastrophic effect.

The drawbacks of this method are threefold:

- the existence of a single path between two features is too weak a relation, and does not take the available redundancy into account. Consider two cases: in the first one, there exists only one path between two features f_1 and f_2, and the length of this path is 10; in the second case, there exist 10 paths of length 2 between these two features. It is clear that the probability that f_1 and f_2 are in the same global match is much bigger in the second case than it is in the first one. We would like the algorithm to take such an information into account.

- the constraints of the problem are neglected: it is not common for two features of the same image to belong to the same global match (although it does happen when a corner gives rise to two points rather than one). The case of 3 features of the same image may be excluded a priori. Ideal components should contain, in the best case, one feature of every image.

- as stated before, every error has a catastrophic effect.

4.1.2 Second method: strong connectedness

We want to eliminate these drawbacks, especially the last one, so that one error is not sufficient to merge two components. We define the notion of *strong connectedness*. Two features are said to be *k-strongly connected*, if there exist at least k paths of length less than or equal to 2 between the two features in the matches graph.

It is then possible to search for the equivalence classes of the transitive closure of this relation for a given k. This method provides much better results than the first one, but raises the problem of choosing the threshold k.

4.1.3 Third method: strong connectedness and threshold adaptation

To deal with this last problem, we propose a method to automatically adapt the threshold k. Let $C_k = \{c_k^i\}$ be the set of all components obtained with the relation of strong connectedness for a given threshold k. It is clear that the components of C_{k+1} form a partition of those of C_k. Each component c_{k+1}^i is included in a particular component c_k^j. All these components may then be organized in a tree: the root is the set of all image features; at the first level, the tree nodes are the components c_1^i; at level two are the components c_2^i... The edges between the tree nodes represent the relation of inclusion between the components of two consecutive levels.

The FIG. 12 presents such a tree. On the first line is c_0^1 which regroups all the components. The components obtained which k equal to 1, 2 and 3 are respectively on the second, third and fourth lines.

We use the following algorithm, where n is the number of images and s_{top} et s_{down} are two thresholds, taken respectively equal to 1.2 and 0.8 in practice, for all examples. The algorithm is not very sensitive to these two thresholds. They just constrain every component to have almost one feature by image.

1. The tree is examined from its root.

2. If an examined component c_k^i contains more than $n.s_{top}$ features, its sons are respectively examined. On FIG. 12, this is the case for c_1^1, c_1^2 and c_1^3 on the first line.

   If none of them may form a final component, c_k^i is a final component. This case is illustrated by c_2^1 on FIG. 12.

3. If c_k^i contains less than $n.s_{down}$ features, it is not a final component. This is the case for c_3^1, c_3^2 or c_3^3 on FIG. 12.

4. If c_k^i contains between $n.s_{down}$ and $n.s_{top}$ features, it is a final component. This case is illustrated by c_1^4 or c_2^3 on FIG. 12.

Two post-treatments are added to this search method.

Dilatation: let a final component have no final component as brother in the tree: if it does not contain a feature from an image, and if its father

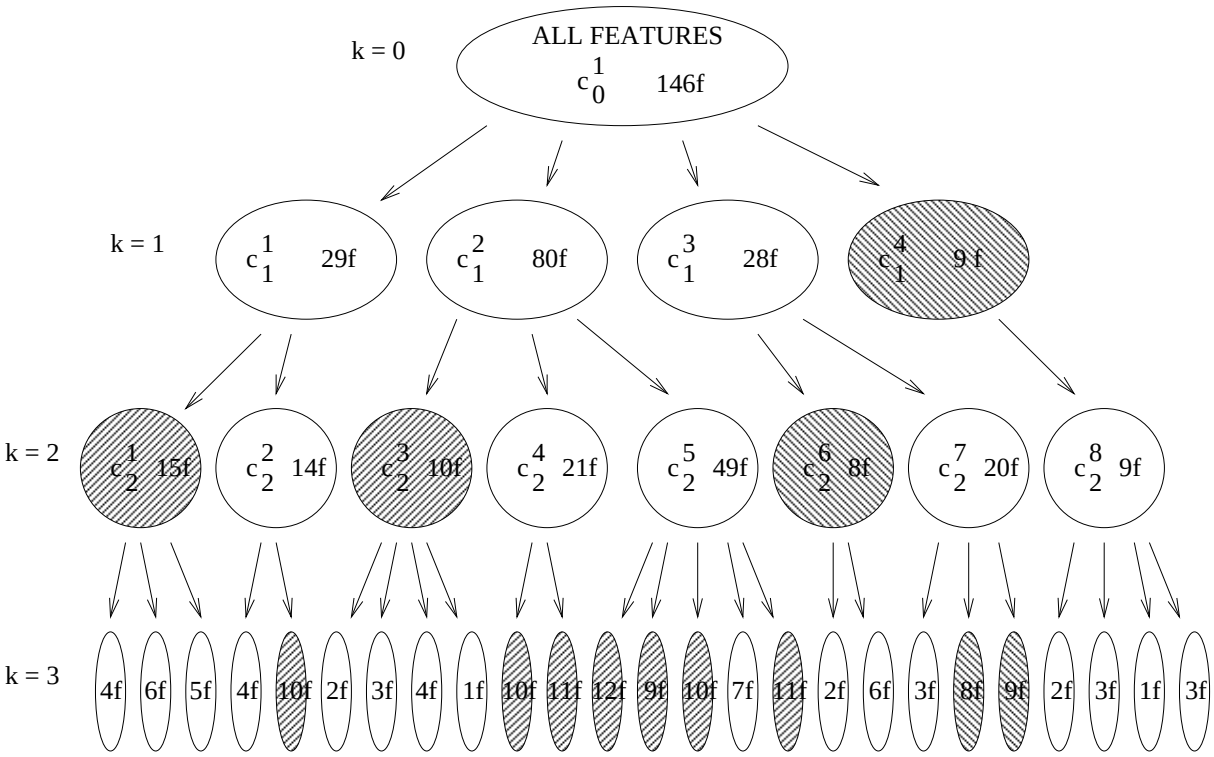

Figure 12: An illustration of threshold adaptation. The component name is written in the first components, and the number of contained features in written in every component. The final components are hatched.

contains a feature from this missing image, this feature is added to the final component.

Reduction: let a final component contain two or more features of the same image: only the feature the most connected with the rest of the component remains in this component. Such a reduction will be applied to c_2^1 on FIG. 12.

4.2 Experimental results

We applied the previous algorithm, with strong connectedness, automatic threshold adaptation and the two post-treatments, to the graphs formed by point matches only. Segments matches were deduced from point matches afterwards.

Ten images of the object shown in FIG. 3 and 7 were matched. FIG. 13 shows the result of the global match on image numbers 1, 4, 7, and 10 of the sequence. Matching points have the same number.

During the computation, 68 global matches were found: 50 contain a feature from every image, 5 contain a feature from only 9 images, and 4 a feature from 8 images. The average number of errors is 5 per image. These errors are mainly due to noisy segments of the backgroung, or to the hole in the side of the object.

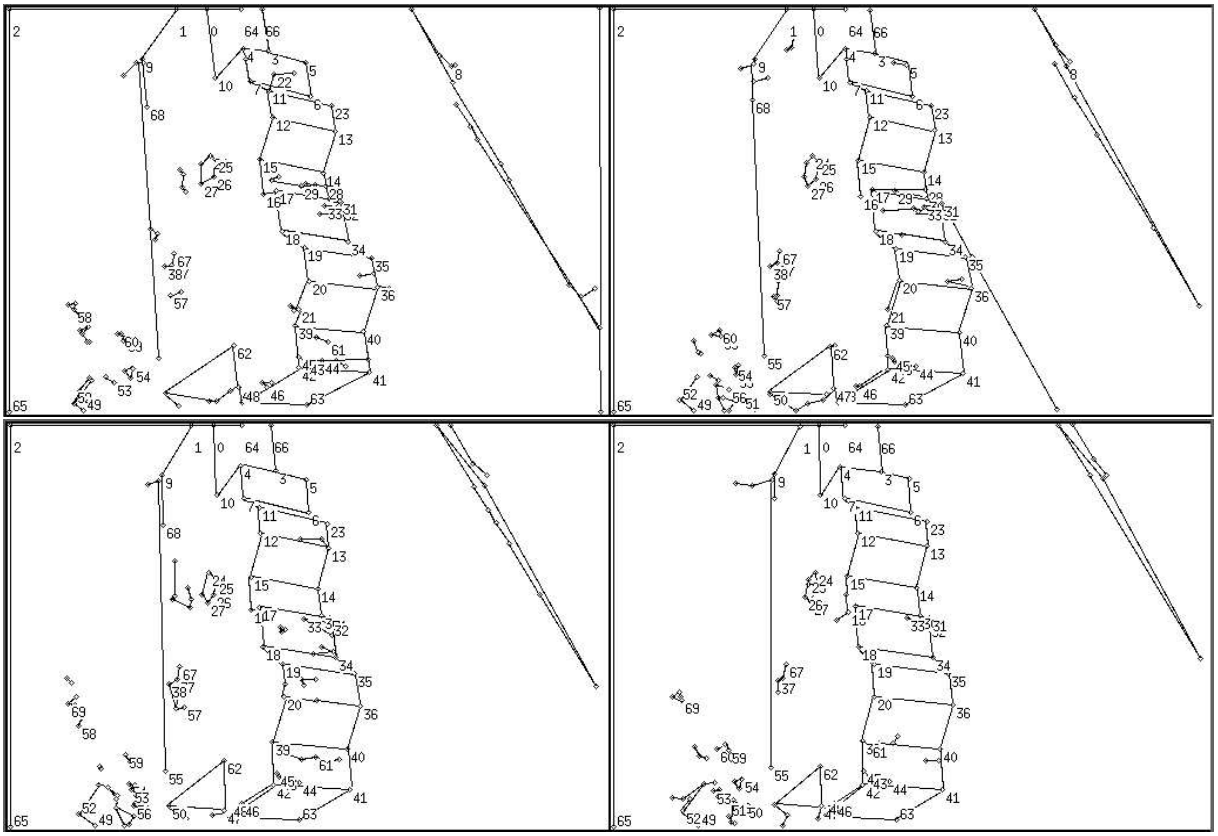

Figure 13: Global match of a set of ten images (the images presented here have the number 1, 4, 7, and 10 in the sequence).

5 Aspect modeling

Finally, we illustrate the use of the previous algorithms to model the different aspects of an object. The aim of this modeling process is to find which aspects of an object are present in a set of images, and to provide a "model", or "generic image", for each aspect of the object.

Many methods have been proposed in the literature to compute the different aspects of an object from its 3D CAD model[44]. This problem is difficult, because CAD data does not contain exactly the information needed to describe real visual appearance, and computationally expansive, because there are many more topological aspects than visual ones [45].

We thus take a pragmatic approach, using kind of "learning from example" paradigm. The aspects are deduced from what is actually seen in real images. Of course our models do not have the accuracy of CAD models, but the complexity of the computation has no comparison also.

5.1 Modeling principle

From several images of an object, we want to automatically establish a model of all aspects of the object that are represented in the image set. With this

aim in view, we use the following algorithm.

1. The images are matched pairwise.
2. A dissimilarity measurement is evaluated for each pair of images. The results are put in a dissimilarity matrix.
3. A hierarchical clustering algorithm is used to gather images representing the same or nearby aspects of the object into components.
4. In each components, a global match is made using the partial matches computed at stage 1.
5. A model of each represented aspect is computed from these global matches.

Stages 1 and 4 have been already described. The three other stages are described below.

5.1.1 Image dissimilarity

We need to evaluate the dissimilarity of the contents of two images. This can be done by computing the fraction of the image tokens that were matched between the two images. We use the following formula for two images I_1 and I_2:

$$d(I_1, I_2) = \frac{2}{3}\frac{N_v(I_1) \times N_v(I_2)}{N_v(I_1, I_2)^2} + \frac{1}{3}\frac{N_e(I_1) \times N_e(I_2)}{N_e(I_1, I_2)^2}$$

where $N_v(I)$ and $N_e(I)$ are respectively the number of vertices and the number of edges of image I, and $N_v(I_1, I_2)$ and $N_e(I_1, I_2)$ are respectively the number of vertex matches and of edge matches between the two images.

The value of the dissimilarity is thus 1 when the images are totally matched, and ∞ when the matching algorithm has failed totally. It is not a distance in a strict mathematical sense, but it is sufficient for a clustering algorithm.

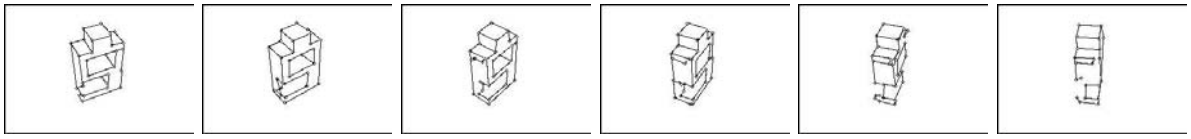

Figure 14: Six views of an object.

For example, consider the six images shown in FIG. 14. They were matched using Euclidean invariants, and their distance matrix is:

$$\begin{pmatrix} 1 & 1.82859 & 11.8551 & 13.6304 & 27.2987 & 61.0833 \\ 1.82859 & 1 & 2.35642 & 24.6308 & 23.7482 & 15.5712 \\ 11.8551 & 2.35642 & 1 & 2.42964 & 18.4236 & 38.6125 \\ 13.6304 & 24.6308 & 2.42964 & 1 & 2.78519 & 41.875 \\ 27.2987 & 23.7482 & 18.4236 & 2.78519 & 1 & 2.67569 \\ 61.0833 & 15.5712 & 38.6125 & 41.875 & 2.67569 & 1 \end{pmatrix}$$

5.1.2 Hierarchical clustering

Numerous clustering methods are proposed in the literature [46, 47]. They usually require either a threshold or the final number of clusters to be given as input to the algorithm.

As we do not have any idea of the number of clusters that should be obtained in our case, we preferred to use a hierarchical merging method based on a threshold to partition the initial image set.

1. Each image is put in a different set.
2. The two nearest sets (the least dissimilar ones) are merged if their dissimilarity is smaller than a given threshold.
3. The dissimilarity matrix is updated. The dissimilarity between two image sets is, by definition, the mean dissimilarity between the images of the first set and the images of the second one.
4. The stages 2 and 3 are repeated until no more sets can be merged.

Updating the matrix is easy. If I, J and K are three image sets, and if $|L|$ denotes the number of elements of a set L:

$$\begin{aligned} d(I \cup J, K) &= \frac{1}{|I \cup J|.|K|} \sum_{i \in I \cup J, k \in K} d(i,k) \\ &= \frac{|I|}{|I \cup J|} d(I,K) + \frac{|J|}{|I \cup J|} d(J,K) \end{aligned}$$

The choice of threshold is critical. In our experiments, we had to adapt the threshold according to the image quality. Noisy images need large thresholds (between 10 and 20), while exact data, coming from CAD models for example, needs smaller ones (around 2 or 3).

5.1.3 Modeling

In the last stage of the process, the images have been grouped into components that partition the initial image set, and in each component, the images have been globally matched. The aim of the last stage is to establish a "model image" for each component.

For each component, a key image is chosen. All of the other images in the component are transformed by a homography to be as close as possible to the chosen image (this homography was already computed during the matching improvement stage).

The model is a new image which contains the vertices and edges present in at least 60% of the component images. For features that are chosen to be part of the model, their positions are computed as the mean position of the corresponding features in the transformed images.

The rate of 60% is smaller than the s_{down} threshold previously used, because the reduction post-treatment may reduce the number of features in the global matches.

5.2 Experimental results

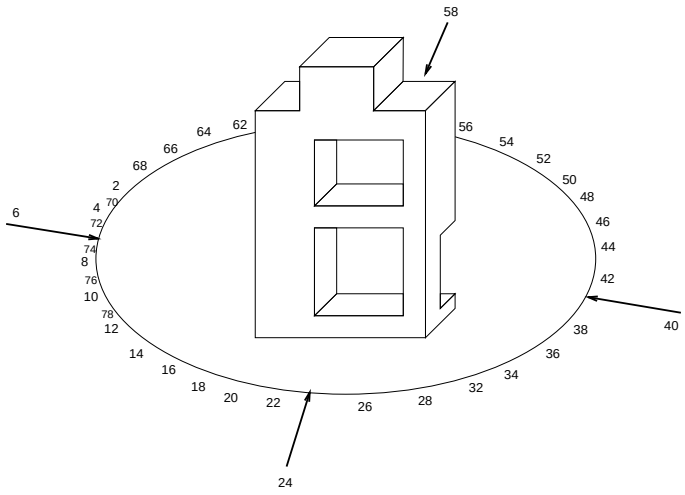

Figure 15: 78 images of an object.

To test the modeling algorithm, we used the object shown in FIG. 15. Seventy eight images of this object were taken by turning the camera around it. With a dissimilarity threshold equal to 10 in the clustering algorithm, we obtained 10 image sets S_i: FIG. 16 shows the features extracted from four images of each set, and the model computed for this set.

Here are the images contained in each set:

$S_1 = \{1, 2, 3, 33, 34, 35, 36, 37, 67, 68, 69, 70\}$
$S_2 = \{4, 5, 6, 7, 38, 39, 40, 41, 71, 72, 73, 74\}$
$S_3 = \{8, 9, 42, 43, 44, 45, 75, 76\}$
$S_4 = \{10, 11, 12, 77, 78\}$
$S_5 = \{13, 14\}$
$S_6 = \{46, 47\}$
$S_7 = \{15, 16, 17, 18, 48, 49, 50, 51, 52\}$
$S_8 = \{19, 20, 21, 22, 23, 24, 53, 54, 55, 56\}$
$S_9 = \{25, 26, 27, 57, 58, 59, 60, 61, 63\}$
$S_{10} = \{28, 29, 30, 31, 32, 62, 64, 65, 66\}$

It should be noted that the object is almost symmetric with respect to a vertical plane and each set contains some images taken from both sides of the object.

To illustrate the modeling algorithm, we also used the images shown in the previous sections, which have more numerous features. The model shown on the left of FIG. 17 was computed from the ten images of the object shown on FIG. 3. This model contains 70 points.

The model shown on the right of FIG. 17 was computed from five images of the house shown on FIG. 4. This model contains 181 points.

6 Conclusion

This paper has presented several algorithms for image matching and object modeling, in the case where the images are formed of line segments and vertices

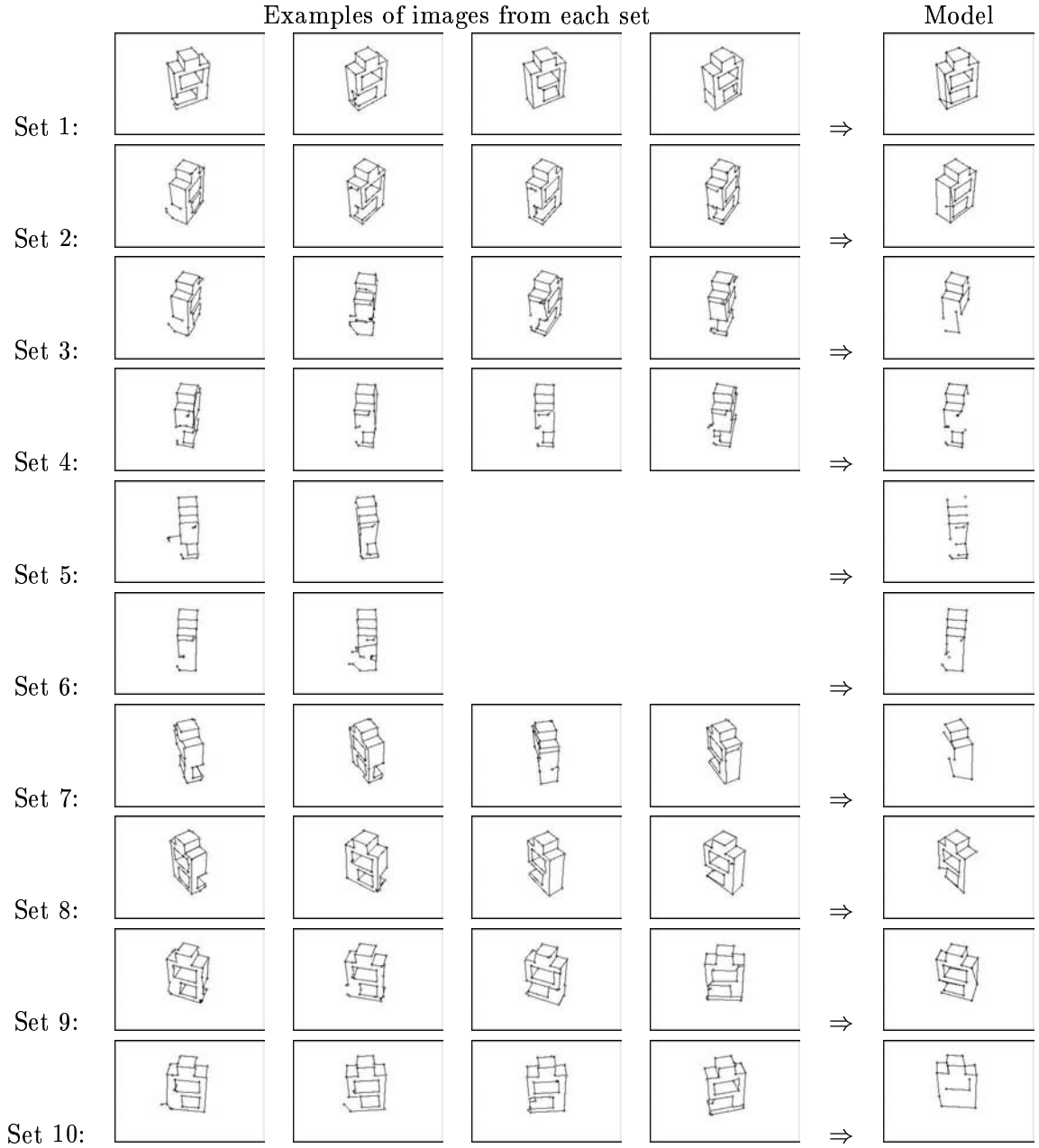

Figure 16: Four images and the model of each set.

or points between these segments. These allow two or more images to be matched, and the different aspects of an object to be modeled. All these algorithms were illustrated with results obtained from real images.

These algorithms are based on the use of local planar invariants and a global constraint (all of the matches have to be consistent with the same apparent motion).

These tools allow us to:

- use uncalibrated cameras or stereo rigs,

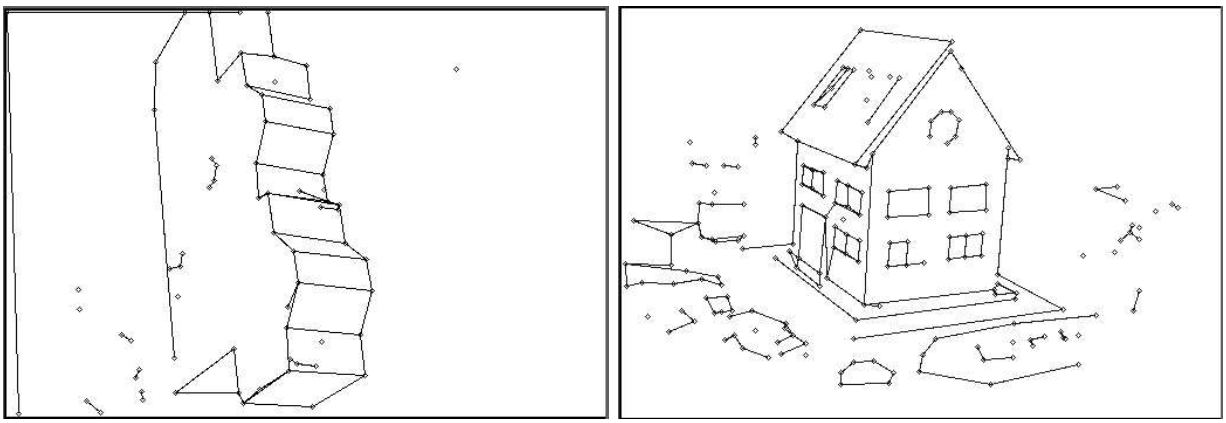

Figure 17: The two obtained models.

- manage the case where there are mobile objects in the scene,
- work with noisy images and occluded objects,
- keep a low complexity.

These advantages allow us to deal with problems for which no other existing method is applicable, and the results clearly show the robustness of the methods.

The work may be extended in three directions.

1. The basic algorithm, that of two image matching, is based on the existence of local quasi-invariants. It ought to work with other features, such as curves for example.
2. The models we obtain summarize the object features that are stable and characteristic. These may be used to recognize the object in other images. The problem is to be able to index such models in a data base, and to have an efficient search algorithm.
3. The modeling process is quite close to the "learning from example" paradigm. The problem is to show that our modeling technique really is a learning process, i.e. that the models that are built allow a better and faster recognition than the initial images.

References

[1] O. Faugeras. *Three-Dimensional Computer Vision - A Geometric Viewpoint.* Artificial intelligence. M.I.T. Press, Cambridge, MA, 1993.

[2] R.E. Kelly, P.R.H. McConnel, and S.J. Mildenberger. The Gestalt photomapper. *Photogrammetric Engineering and Remote Sensing*, 43:1407–1417, 1977.

[3] W. Förstner and A. Pertl. Photogrammetric standard methods and digital image matching techniques for high precision surface measurements. In E.S. Gelsema and L.N. Kanal, editors, *Pattern Recognition in Practice II*, pages 57–72. Elsevier Science Publishers, 1986.

[4] D.B. Gennery. *Modelling the Environment of An Exploring Vehicle by Means of Stereo Vision.* Ph.d. thesis, Stanford University, June 1980.

[5] H.K. Nishihara. PRISM, a pratical real-time imaging stereo matcher. Technical Report Technical Report A.I. Memo 780, Massachusetts Institute of Technology, 1984.

[6] M. Kass. A computational framework for the visual correspondence problem. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence, Karlsruhe, Germany*, pages 1043–1045, August 1983.

[7] M. Kass. Linear image features in stereopsis. *International Journal of Computer Vision*, 1(4):357–368, January 1988.

[8] A.W. Gruen. Adaptative least squares correlation: a powerful image matching technique. *S. Afr. Journal of Photogrammetry, Remote Sensing and Cartography*, 14(3):175–187, 1985.

[9] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2:283–310, 1989.

[10] P. Fua. A parallel stereo algorithm that produces dense depth maps and preserves image features. *Machine Vision and Applications*, 1990.

[11] D. Marr and T. Poggio. Cooperative computation of stereo disparity. *Science*, 194:283–287, 1976.

[12] D. Marr and T. Poggio. A computational theory of human stereo vision. *Proceedings of the Royal Society of London*, B 204:301–328, 1979.

[13] W.E.L. Grimson. A computer implementation of the theory of human stereo vision. *Philosophical Tansactions of the Royal Society of London*, B292(1058):217–253, 1981.

[14] W.E.L. Grimson. Computational experiments with a feature based stereo algorithm. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 7(1):17–34, 1985.

[15] S.B. Pollard. *Identifying Correspondences in Binocular Stereo.* PhD thesis, University of Sheffield, 1985.

[16] 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.

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

[18] Y. Ohta and T. Kanade. Stereo by intra and inter-scanline search using dynamic programming. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 7(2):139–154, 1985.

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

[20] N. Ayache and B. Faverjon. Efficient registration of stereo images by matching graph descriptions of edge segments. *International Journal of Computer Vision*, 1(2):107–132, April 1987.

[21] T. Skordas. *Mise en correspondance et reconstruction stéréo utilisant une description structurelle des images.* Thèse de doctorat, Institut National Polytechnique de Grenoble, France, 1988.

[22] L. Hérault. *Réseaux de neurones récursifs pour l'optimisation combinatoire.* Thèse de doctorat, Institut National Polytechnique de Grenoble, France, 1991.

[23] H.J. Wolfson. Model-based object recognition by geometric hashing. In O. Faugeras, editor, *Proceedings of the 1st European Conference on Computer Vision, Antibes, France*, pages 526–536. Springer-Verlag, April 1990.

[24] C.A. Rothwell. Hierarchical object descriptions using invariants. In *Proceeding of the* DARPA–ESPRIT *workshop on Applications of Invariants in Computer Vision, Azores, Portugal*, pages 287–303, October 1993.

[25] D. Forsyth, J. Mundy, and A. Zisserman. Transformationnal invariance - a primer. In *Proceedings of the British Machine Vision Conference, Oxford, England*, pages 1–6, September 1990.

[26] D. Forsyth, J.L. Mundy, A. Zisserman, and C. Rothwell. Invariant descriptors for 3D object recognition and pose. In *Proceeding of the* DARPA–ESPRIT *workshop on Applications of Invariants in Computer Vision, Reykjavik, Iceland*, pages 171–208, March 1991.

[27] J.L. Mundy and A. Zisserman, editors. *Geometric Invariance in Computer Vision.* The MIT Press, Cambridge, MA, USA, 1992.

[28] J.L. Mundy and A. Zisserman, editors. *Proceedings of the Second* ESPRIT - ARPA *Workshop on Applications of Invariance on Computer Vision, Ponta Delagada, Azores, Portugal*, October 1993.

[29] T.O. Binford and T.S. Levitt. Quasi-invariants: Theory and exploitation. In *Proceedings of* DARPA *Image Understanding Workshop*, pages 819–829, 1993.

[30] J. Ben-Arie. The probabilistic peaking effect of viewed angles and distances with application to 3D object recognition. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 12(8):760–774, August 1990.

[31] C.F. Olson. Probabilistic indexing for object recognition. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 17(5):518–522, May 1995.

[32] I. Shimshoni and J. Ponce. Probabilistic 3D object recognition. In *Proceedings of the 5th International Conference on Computer Vision, Cambridge, Massachusetts, USA*, pages 488–493, 1995.

[33] Y. Lamdan and H.J. Wolfson. Geometric hashing: a general and efficient model-based recognition scheme. In *Proceedings of the 2nd International Conference on Computer Vision, Tampa, Florida, USA*, pages 238–249, 1988.

[34] Y. Lamdan, J.T. Schwartz, and H.J. Wolfson. Affine invariant model-based object recognition. IEEE *Journal of Robotics and Automation*, 6:578–589, 1990.

[35] I. Weiss. Geometric invariants and object recognition. *International Journal of Computer Vision*, 10(3):207–231, 1993.

[36] J. Canny. A computational approach to edge detection. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[37] R. Deriche. Using Canny's criteria to derive a recursively implemented optimal edge detector. *International Journal of Computer Vision*, 1(2):167–187, 1987.

[38] B. Lamiroy and P. Gros. Rapid object indexing and recognition using enhanced geometric hashing. In *Proceedings of the 4th European Conference on Computer Vision, Cambridge, England*, volume 1, pages 59–70, April 1996. Postscript version available at ftp://ftp.imag.fr/pub/MOVI/publications/Lamiroy_eccv96.ps.gz.

[39] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, 1988.

[40] F. Klein. *Le programme d'Erlangen*. Collection "Discours de la méthode". Gauthier-Villars, Paris, 1974.

[41] Z. Zhang, R. Deriche, O. Faugeras, and Q.T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. Rapport de recherche 2273, INRIA, May 1994.

[42] R. Hartley. In defence of the 8-point algorithm. In *Proceedings of the 5th International Conference on Computer Vision, Cambridge, Massachusetts, USA*, pages 1064–1070, June 1995.

[43] B. Boufama and R. Mohr. Epipole and fundamental matrix estimation using the virtual parallax property. In *Proceedings of the 5th International Conference on Computer Vision, Cambridge, Massachusetts, USA*, pages 1030–1036, June 1995.

[44] L. Shapiro and K. Bowyer, editors. IEEE *Workshop on Directions in Automated CAD-Based Vision*, Maui, Hawai, 1991. IEEE Computer Society Press.

[45] K. Bowyer. Why aspect graphs are not (yet) practical for computer vision. In *Proceedings of the* IEEE *workshop on Direction on automated* CAD-*based Vision, Maui, Hawaii, USA*, pages 97–104, 1991.

[46] E. Diday and J.C. Simon. Clustering analysis. In K.S. Fu, editor, *Communication and Cybernetics*. Springer-Verlag, 1976.

[47] G.L. Scott and H.C. Longuet-Higgins. Feature grouping by "relocalisation" of eigenvectors of the proximity matrix. In *Proceedings of the British Machine Vision Conference, Oxford, England*, pages 103–108, September 1990.