Model Based Object Recognition using Stereo Vision and Geometric Hashing

H.A.L. van Dijck F. van der Heijden M.J. Korsten

University of Twente, Department of Electrical Engineering
P.O. Box 217, 7500 AE Enschede, The Netherlands, e-mail: harrie@mi.el.utwente.nl

1 Introduction

Stereo vision enables efficient object recognition by providing 3 dimensional information about the scene under consideration. The use of stereo vision for object recognition, however, requires a solution to the correspondence problem, i.e. points in the left and right image must be paired in such a way that they are the respective projections of the same 3-D points in the world coordinate system. This problem can be partially resolved by defining special points related to local features which may be found in both images independently. Imperfect detection of these features, noise and the abundance of such features, however, may still prevent a unique and correct stereo match.

We propose a technique that combines stereo vision with geometric hashing to deal with these problems. We do not try to solve the correspondence problem immediately. Instead we feed the 3-D points generated by all pairs of points in respectively the left and right image that meet the geometrical constraints to form a stereo pair, to a geometric hashing algorithm that performs model-based recognition. This set of 3-D points not only contains the correct points but also a lot of incorrect, spurious points. In this paper we will show that the inherent robustness of geometric hashing to spurious data can be used to overcome the problems in stereo vision.

The organisation of this paper is as follows. Section 2 discusses the geometric hashing technique and some previous work in this area. In section 3 we describe the stereo vision technique used. Section 4 shows some experimental results that prove the applicability of our method with real images and in section 5 we draw some overall conclusions and we discuss our future research interests.

2 Geometric hashing

2.1 The basic technique

Geometric hashing is a technique to generate candidate matches between a model database and a set of measurement features. It was first employed by Lamdan *et al.* [5, 6] and has since then been used and developed by many other authors [2, 3, 7, 8].

Geometric hashing performs recognition with a set of points. In our case these points will be 3 dimensional points. Each of these points may have an attribute list to accommodate features that carry more information than just a position (e.g. a line). The algorithm works in 2 stages. In the first stage, which can be performed off-line with just the model database, a hash table is generated from the set of object models. In the second stage, the actual recognition, this hash table is used to perform recognition on a set of measurement features.

For the first stage we take a model of an object that consists of say k points. In order to find an object with arbitrary position and orientation, we want to express the position of the points in a way that is invariant to rotations and translations. This can be done by expressing these positions in a coordinate system which is fixed with respect to the object and defined by a subset of the model points. In our case of 3 dimensional points, we need 3 points to define such a coordinate system. The first of these basis points will become the new origin. Thus we translate the set of model points such that this points coincides with origin of the coordinate system. Next we rotate the set of points about the y-axis and z-axis until the second basis point falls onto the x-axis. Finally we rotate the set of points about the x-axis until the third basis point falls into the xy-plane. If any of the attributes of a point contains geometrical information, this information should also be updated in accordance with these transformations. The resulting set of points

is invariant to translations and rotations because any translation or rotation would also affect the basis points leaving the transformed point set untouched.

In our case the points will carry no further attributes except their geometrical position. Therefore the first basis point contains no longer any information in the transformed point set as it will always be located at the origin of the coordinate system. The remaining k-1 points are stored in a 3 dimensional hash table which in fact is just a 3 dimensional array that quantifies the 3-D space into bins. Each of the k-1 points is stored in the appropriate bin tagged with the model-basis combination that generated it. The process of defining a new coordinate system is repeated for each subset of 3 non-collinear model points. This results in approximately k*(k-1)*(k-2) different sets of basis points, each generating k-1 entries in the hash table. Due to symmetries in the object model, different sets of basis points may generate exactly the same set of entries in the hash table. Therefore, our software keeps track of the transformed point sets already stored in the hash table, thereby preventing the same set from being stored twice with different bases. In this manner all available object models are stored in the hash table.

During the second stage we do basically the same with a set of measurement features (points). Assume that we have a set of n 3-D measurement points. From these n points, we select at random 3 points to form a basis and we generate a set of n-1 transformed points. For each of these transformed points we access the hash table at the appropriate bin and all bins surrounding it (27 in total). For each model-basis combination whose entries we find in these bins we cast a vote. After all n-1 points have been processed in this way, we collect the votes for all model-basis combinations. If the vote count of any of the model-basis combinations exceeds a certain threshold we accept a match of the model of the model-basis combination with a pose defined by the transformation between the basis points from the measurement set and the basis points from the model-basis combination. If none of the vote counts exceeds the threshold a new subset of measurement points is selected to form the basis points and the process is repeated until all possible bases in the set of measurement points have been tried.

Although most of the work in geometric hashing considered 2-D models and images, there have been some examples of 3-D object recognition in recent literature. Lamdan and Wolfson [6] discuss the recognition of 3-D objects from a single 2-D image. They propose a number of approaches but only discuss one method in detail in which 2-D aspect models are generated from the 3-D object models for a discrete set of viewing angles. A drawback of this approach is that it increases the number of models in the hash table, although this may be compensated by the fact that usually less features are necessary than for 3-D models to form a basis, so that the total number of model-basis combinations doesn't necessarily increase. Furthermore, the approach introduces errors due to the fact that only a discrete number of viewing angles can be modelled and the approach discards part of the information that is available in the 3-D models. Gavrila and Groen [3] use a similar approach in which they adapt the set of viewing angles for each of the 3-D models based on the errors caused by the discretization. Flynn and Jain [2] use an indexing technique similar to geometric hashing, for invariant features based on 3 dimensional surfaces. Their approach, however, seems most suitable for range images.

2.2 Weighted voting

The crude voting strategy used by Lamdan does not allow a graceful degradation of the vote count for a model-basis combination as the amount of noise in the feature positions increases. Rigoutsos and Hummel [7, 8], therefore, proposed a weighted voting system. In fact they developed a Bayesian approach to geometric hashing to derive an expression for weighted voting. We applied a simplified form of their result in our application.

The weighted vote is given by

$$z(\vec{b}_{l,i}, \vec{d}_j) = \exp(-\frac{\|\vec{b}_{l,i} - \vec{d}_j\|^2}{\sigma_m^2}), \tag{1}$$

where $\vec{b}_{l,i}$ indicates the ith hash entry from model-basis combination l, $\vec{d}_j$ indicates the jth transformed point from the measurement set and σ_m is a parameter of the weighing

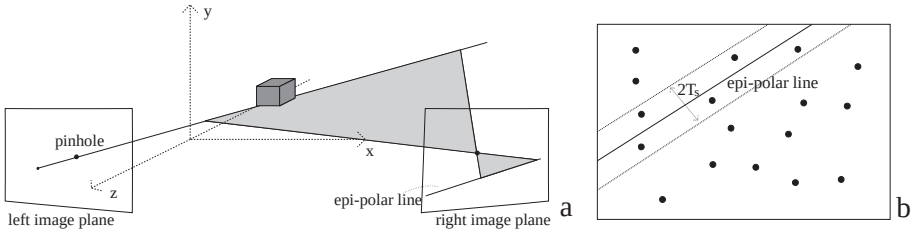


Figure 1: Stereo matching, (a) reconstruction of the epi-polar line for a point in the left image; (b) candidate stereo matches in right image.

function. The parameter σ_m indicates the width of the weighing function and should be fixed according to the expected error in the transformed measurement points. During the recognition phase, instead of just casting a vote, we will now calculate a weighted vote for each entry that we find in one of the 27 bins that we access for a certain transformed measurement point. It is assumed that the weighted vote for a hash entry outside these bins would have been negligible anyway, which is a reasonable assumption if the linear bin size is at least equal to σ_m.

The total vote for a certain model-basis combination is now collected according to

$$Z(\vec{b}_l) = \sum_{i=1}^{k-1} \max_{j=1..(n-1)} \{z(\vec{b}_{l,i}, \vec{d}_j)\}. \tag{2}$$

Here the summation sums over the different hash entries due to one model-basis combination and the maximisation ensures that only one measurement point will vote for a certain model point.

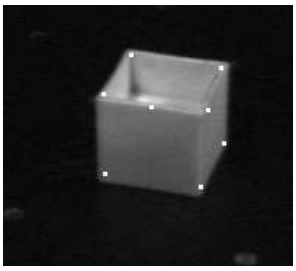
3 Stereo matching

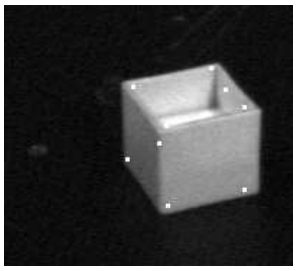
We use a calibrated stereo setup in which we know the transformations from the 3 dimensional world coordinates to the 2 dimensional coordinates in both image planes. These transformations are based on a pinhole camera model and involve the position and orientation of the camera (external parameters), a perspective projection and the position and orientation of the image plane (internal parameters). Except for the perspective transformation, the transformations are linear. If a subset of the internal parameters is given (e.g. by the manufacturer of the camera and the lens), the other parameters may be estimated using a calibration object with a known 3-D position and orientation.

If we know the 2-D positions of both projections of the same 3-D point (a stereo pair), we can reconstruct the position of the 3-D point using these positions and our knowledge of the camera transformations. The difficulty, however, is finding these stereo pairs. This problem is called the *correspondence problem* in stereo vision. The problem can be reduced by defining special points, local features, such that the number of candidate stereo pairs strongly decreases. The number of candidate stereo pairs can be further reduced by using the epi-polar line constraint [1]. Given a point in one of the images, we can calculate its projection line, the line in the 3-D world that projects onto this particular point, see figure (1). This line is given by the line that goes through the image point on the image plane and the pinhole of the camera. Next we project this line onto the image plane of the other camera, generating the so called epi-polar line. Because the 3-D point that generated the first point must be somewhere on the projection line, the projection of this point onto the second image plane must be somewhere along the epi-polar line.

Due to noise, discretization errors and limitations of the camera model, however, we can not expect this second point to be exactly on the epi-polar line. Therefore we take a strip along the epi-polar line of width $2T_s$, T_s being the maximum distance of a candidate point to the epi-polar line. All points that fall within this strip will form a candidate stereo pair with the point that generated the epi-polar line. This means that in general one image point will form multiple candidate stereo pairs which are mutually exclusive.

Once a candidate stereo pair has been established, the 3-D point can be reconstructed by calculating the projection lines of both image points. In general these lines will not


a


b

x(mm)	y(mm)	z(mm)
150	28	-224
150	49	-171
172	27	-208
132	27	-206
154	0	-252
152	16	-214
154	30	-184
151	28	-167
174	2	-205
132	3	-203
155	-29	-251
155	2	-185

c

Figure 2: Stereo matching results with a cube, (a) left image showing the cube and the detected corner points; (b) right image; (c) coordinates of 3-D points calculated from these images in the coordinate system shown in figure (1a).

intersect. Therefore we calculate the mid point between the points on the lines where these lines draw nearest to each other. This is the 3-D point that we offer to the recognition phase.

4 Experiments

4.1 Experimental setup

We have applied our method in an experiment involving simple geometric objects described earlier by Hoogeveen and Korsten [4]. We want to test the behaviour of our method for different values of the parameters and we want to test its ability to deal with complex scenes in the presence of occlusion. Our model database contains models for a cube, a hexagon and a wedge. These objects are modelled by their corner points. Furthermore, we consider all possible bases in the set of measurement points. We do not stop if we find a probable match because this would restrict the number of solutions which is undesirable during testing. The corner points are detected by local maxima in the isophote curvature multiplied by the cube of the gradient, κ. This κ is given by [9]

$$\kappa(\vec{x}) = |I_y^2(\vec{x})I_{xx}(\vec{x}) - 2I_x(\vec{x})I_y(\vec{x})I_{xy}(\vec{x}) + I_x^2(\vec{x})I_{yy}(\vec{x})|, \tag{3}$$

where $I_y(\vec{x})$ etc. are the local partial derivatives of the gray value image $I(\vec{x})$. These derivatives are estimated by convolving the image with the appropriate derivatives of the Gaussian function. We calculate the local maxima of $\kappa(\vec{x})$ and we select those maxima that exceed a certain threshold. Our corner detector, therefore, exhibits 2 parameters, σ_c for the width of Gaussian function and T_c for the threshold.

Figures (2a,b) show the left and right images of a scene containing a cube with the corner points that have been detected, the table in figure (2c) gives the 3-D corner positions that have been calculated from these images. As can be seen from these images, the detection of the corner points works fairly good but there are some spurious points due to shadows and highlights.

4.2 Parameter dependence

To determine the dependence of the success of our strategy on the specific parameter values, we consider the scene in figure (2). Figures (3a,b) show two matching results that are likely to occur with this scene. Figure (3a) shows the projection of the 3-D points of the modelled cube onto the left image. Note that the position of the cube model is non-optimal: one of the model points coincides almost exactly with an image point, namely the point that was selected as the origin of the coordinate system. Figure (3b) shows the hexagon model matched to an image of the cube. It turns out that 4 corners of the hexagon can be made to coincide almost exactly with corners of the cube. Together with the fact that no more than 6 corners of the cube will be visible in both the left- and right image, this makes the hexagon a very likely candidate match for scenes containing the cube. Therefore, as a measure for the reliability of the match we introduce the ratio of the highest vote (as given in equation (2)) for a model-basis combination of the cube over the highest vote for a model-basis combination of the hexagon.

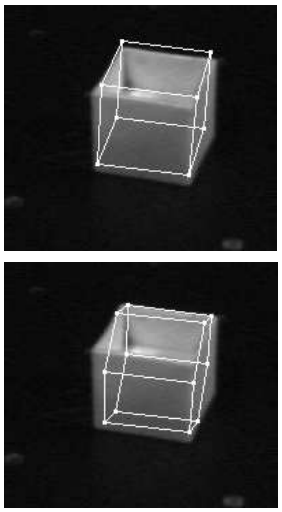


Figure 3: Model matching results with the cube, (a) left image showing the correct model; (b) image of cube matched to hexagon model.

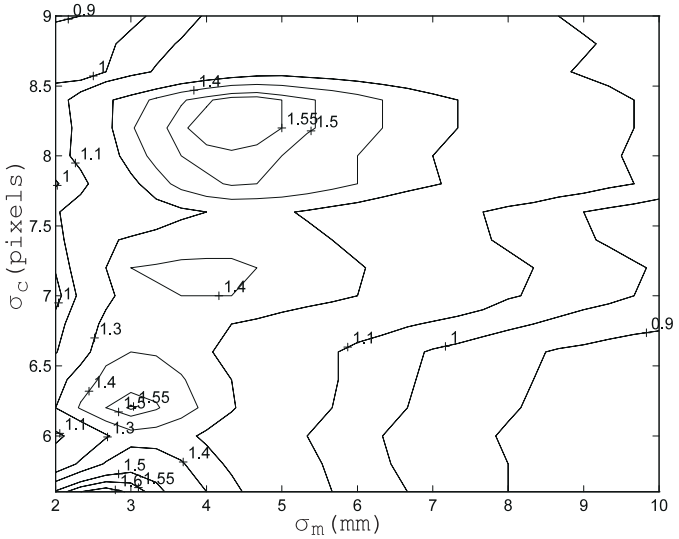


Figure 4: Contour plot of relative match value ($T_c = 0.6$, $T_s = 5$).

This relative match value is plotted in figure (4) as a function of the parameters σ_c and σ_m. This figure shows that to gain an optimal result with an increasing value of σ_c, σ_m has to increase as well. This is understandable because an increase in the value of σ_c means that corner points are detected more reliable but with less positional accuracy, so that the position of the 3-D points will be less accurate. Therefore to find a good match the width of the weighing function also has to increase.

As for the thresholds, it appears that the best results are obtained by choosing the threshold values such that a lot of 3-D points are generated. Which means choosing a low value for the curvature threshold T_c and a high value for the strip width T_s. This is in accordance with our assumption that geometric hashing is very robust to spurious data so that the incorrect points have little effect on the final result while these parameter settings ensure that all correct 3-D points are found. There is, however, a drawback to such settings. Because we consider all possible bases in the scene points, the amount of computing time necessary to evaluate a scene is of the order n^4 (n being the number of 3-D points calculated from the images). There are about n^3 different bases which each result in order n transformed 3-D points. This problem also limits the range of values for σ_c in figure (4). For $\sigma_c = 5.6$ pixels we find 46 3-D points which take about 40 minutes to evaluate on a SUN Sparc 20, while with $\sigma_c = 8$ pixels we find only 10 3-D points which take about 14 seconds to evaluate.

4.3 Multiple objects and occlusion

Because our model matching strategy uses only local features, it should be able to recognize objects that are partially occluded. In fact we already did that in the images with the cube, the cube partially occludes itself. But we have also analysed more complex images where multiple objects where visible, partially occluding each other. Figure (5a) shows such an scene. For these scenes we had to limit the value of T_s because for large values of the strip width T_s the close presence of multiple objects led to a large number of 3-D points that took long to evaluate and generated incorrect, random matches. Figure (5b) shows a match result for the occluded scene where the hexagon has been successfully detected.

We have also tried to find multiple objects in one scene. In order to do this, we remove the scene points belonging to the detected object after a match has been found and repeat the matching process until no more objects are found. We tested this approach on images containing multiple objects that did not occlude each other and it worked fine on these images. However, for the scene in figure (5a), we were not able to detect the cube after the hexagon had been found. This implies that the match value of the cube was not larger than the match value of the "best" incorrect match. It is probably the result of the fact

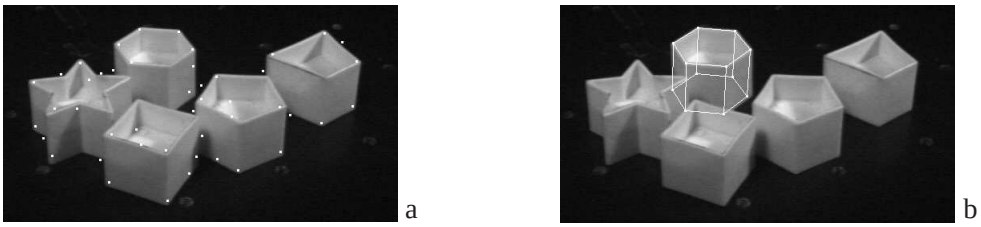


Figure 5: Model matching results with the multi objects, (a) left image the detected points; (b) matching result ($\sigma_c = 8, T_c = 0.6, T_s = 1, \sigma_m = 4$).

that many objects close together generate many 3-D points in a small volume. This makes it hard to find the cube which is modelled by only a few corner points. It is easier to find to hexagon because its model contains more points.

5 Conclusions and future research

Geometric hashing enables the recognition of 3-D objects from stereo images in a relatively simple manner. The proposed technique relies on local features which permits recognition even when the objects are partially occluded. The recognition phase, however, still uses global information in the form of the geometric positions of the features to perform reliable matching. The robustness of the geometric hashing technique to spurious data helps to solve the correspondence problem in stereo vision.

The experiments show that technique works well on simple scenes even with the rather limited set of features that we used. For more complex scenes it turns out that the limited set of model features and the computation time required for such scenes, are a problem. The computation time can be strongly reduced by not evaluating every possible basis but accept a match if its match value exceeds a certain threshold. In that case and if a modelled object is present in the scene, the computational complexity reduces from n^4 to n [5]. In order to be able to put a threshold on the match value, however, we should at least account for the variation in the number of features among the different models. To achieve this we are planning to extend the Bayesian approach developed by Rigoutsos [7] to our application. To increase the information contained in the models, we are planning to extend our set of features to include 3-D lines and curves. If we use a combination of line and point features, but only use the point features as possible elements for a basis, we may realize this extension at very low additional computational costs.

The authors wish to thank Dick Snippe for his contributions to the ideas presented in this paper.

References

[1] O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. The MIT Press, 1993.

[2] P. J. Flynn and A. K. Jain. 3D object recognition using invariant feature indexing of interpretation tables. *CVGIP: Image understanding*, 55(2):119–129, 1993.

[3] D. Gavrila and F. Groen. 3D object recognition from 2D images using geometric hashing. *Pattern recognition letters*, 13:263–278, 1992.

[4] R. M. Hoogeveen and M. J. Korsten. A fast recognition and pose estimation method for solid 3D industrial objects using stereo vision. In *Proceedings of the 9th Scandinavian Conference on Image Analysis*, pages 397–406, 1995. Uppsala.

[5] Y. Lamdan, J. T. Schwartz, and H. J. Wolfson. Affine invariant model-based object recognition. *IEEE transactions on robotics and automation*, 6(5):578–589, 1990.

[6] Y. Lamdan and H. J. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *Proceedings of the Second International Conference on Computer Vision*, pages 238–249, 1988. Tarpon Springs (Fl.).

[7] I. Rigoutsos. *Massively Parallel Bayesian Object Recognition*. PhD thesis, New York University, 1992.

[8] I. Rigoutsos and R. Hummel. A Bayesian approach to model matching with geometric hashing. *Computer vision and image understanding*, 62(1):11–26, 1995.

[9] B. M. ter Haar Romeny, editor. *Geometry-Driven Diffusion in Computer Vision*. Kluwer academic publishers, 1994.