RSD-TR-3-87

An Algorithm for Model Based Recognition of Objects Using Range Maps*

by

Paul G. Gottschalk

Trevor N. Mudge

Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, Michigan 48109-1109

February 1987

Center for Research on Integrated Manufacturing

Robot Systems Division
College of Engineering
The University of Michigan
Ann Arbor, Michigan 48109-1109

*This work was supported in part by AFOSR grant 080012 and ARO grant DAAG29-84-k-0070

Енди
ИМК
1669

This algorithm is a synthesis of work we have done as well as the work of Grimson e al [1]. In this document we describe an algorithm for recognizing 3-d objects from range images that we feel would be efficient, accurate, and robust.

Assumptions:

1) A dense range map is available.

2) 3-d models of the objects to be recognized are available in a form that allow synthetic range maps to be generated at any position and orientation.

Object Models:

The models of the objects are two part. First there is a 3-d model that permits the construction of a sythetic range image with an instance of the object occurring any position and orientation (pose), and a precomputed set of all possible features that car occur on the object. The features are described in detail below.

Features:

The features that the algorithm uses are illustrated in Figure 1. The features consist o two normal vectors and the vector joining them. For generality in the figure, the topmost normal vector is considered to be the i^{th} such vector and the bottom most vector is the j^{th} such normal vector. The normal vectors are meant to approximate the normal to a surface at the point in the range image where they are located. They could be calculated in a number of ways, such as finding the best fit plane to a local neighborhood in the range image, from which the normals are easily found.

These pairs of normals have four parameters that describe them independent of the 3-d pose at which they may appear in the image: d_{ij}, α_{ij}, α_{ji}, and β_{ij}. The fourtuple (d_{ij}, α_{ij}, α_{ji}, β_{ij}) can be viewed as forming a 4-d feature vector. The precise definitior of of the parameters is as follows:

d_{ij} -- The distance between the ith and the jth normal vectors.
α_{ij} -- The angle between the vector joining the ith and jth normals and the ith normal.
α_{ji} -- The angle between the vector joining the jth and ith normals and the jth normal.
β_{ij} -- The angle between the ith and the jth normal vectors.

A Proposed Recognition Algorithm:

Preprocessing:

For each object that may appear in a range map, a set of features must be computed. Roughly speaking, this set of features consists of all possible pairs of normals computed from surface patches on the object. There are two parameters of interest: the patch size and the density of the covering of the patches. The best values for these parameters can not be ascertained *a priori;* experiments must be performed. Some considerations:

A. Patch Size -- 1) Patches cannot be made too small or they will be sensitive to noise.

2) Making the patches to large is wasteful of computation since doing this will not necessarily reduce the density of the covering needed. In addition, large patches will cause smoothing of features on the object.

B. Density of Covering -- 1) The covering must be dense enough that when a feature is randomly chosen from the image, there is a preprecessed model feature vector near to it so that some features will match. In addition, the positional accuracy of the algorthm is directly proportional to the density of the covering.

2) A too dense covering will cause too many hypotheses to be generated by the matching algorithm. This can be dealt with to some degree in the matching algorithm.

Once the patch size and covering density has been found, the normals are computed and then d_{ij}, α_{ij}, α_{ji}, and β_{ij} are computed for all patches i and j. Records using the feature vectors (d_{ij}, α_{ij}, α_{ji}, β_{ij}) as keys are stored in a k-d tree. The records contain, in addition to d_{ij}, α_{ij}, α_{ji}, and β_{ij}, information that will allow the model to be transformed to the pose that is indicated by a feature in the image that matches a model feature (all that is necessary is the location and value of the normal vectors in the model's coordinate system).

Online Recognition Processing: The matching algorithm

The algorithm proceeds as follows, given a dense range image:

1) Segment the image into regions which probably belong to one object.
   One can concieve of many possible ways of doing this. Perhaps the simplest approch would be to use a standard intensity image edge detector to locate range discontinuities that would indicate a probable boundery of an object.
2) Compute two normals from surface patches that, from step 1, are likely to be from the same object. Try to choose them to be as far apart as possible on the same object. (See the second comment below).
3) Compute the feature vector (d_{ij}, α_{ij}, α_{ji}, β_{ij}).
4) Do a range (neighborhood) search for those model features that are near (d_{ij}, α_{ij}, α_{ji}, β_{ij}) using a k-d tree. For more information concerning the use of k-d trees for effecient feature matching, see [2]. The size and shape of the neighborhood should be chosen large enough so that the correct feature will be retrieved (in most cases) if the patches comprising the feature are indeed from the same object. (ie, the ranges for the range search must be chosen)
5) If there are too many matching features (hypotheses) go back to step 2.
6) For each retrieved feature, use the information in its record to create a

hypothesis and check it against what is actually in the range map.

7) If any of the hypothses checked in step 6 match well enough, pick the best matching one as the algorithm's guess for the location of that object in the image. Go back to step 2, but do not take any more features from the identified regions of the range image. If there are no more likely objects, then quit.

Some Comments on the Algorithm:

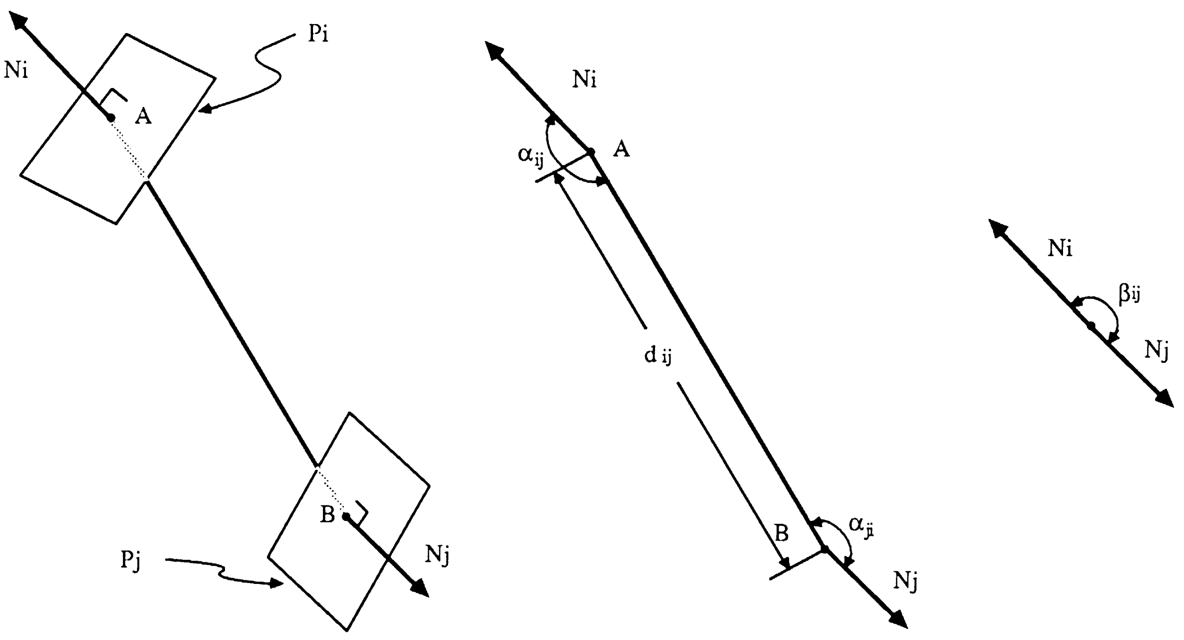
Based on some work that we did on a model of a coke bottle that was of intermediate complexity, we found that there were roughly 40,000 features. The model was created with a geometric modeler using measurements of the coke bottle. Although the density of coverage was probably higher than was necessary in this case, nevertheless the density was not too large by more than a factor of two. Thus there still would have been 10,000 pairs per object (the number of pairs is roughly the number of patches squared). This may be cut down further by considering patches that are a great distance apart as these will likely generate less hypotheses to be tested However, with all that could be done to reduce the number of pairs needed to adequately recognize an object, several thousand pair will be needed. With a large set of objects, the number of pairs may approach millions. Thus the matching strategy must be extremely efficient. The k-d tree, which has O(log N) lookup time, where N is the number of pairs, is the fastest algorithm we know of to perform this task, and is more than fast enough to do it, we believe, in time that is practically useable, whereas a linear algorithm is essentially unusable. See [2] for more information.

Finally, in regard to step 2 of the algorithm outlined above, it is good to choose the feature (ie. the pair of patches from the image) so that the patches are as far apart as possible yet still are likely to lie on a single object and neither of the patches will fall on an object boundery. The reason for this is that a feature whose patches are far apart will be less likely to be similar to another feature. The reason is simply that, because of the assumed continuity of the objects in the scenes, patches that are near eachother (on the average) cannot have widely differing normal vectors. Thus, such features will densly populate certain regions of the 4-d feature space. If (again, on the average) a feature can be chosen from the image to be in a less densely populated portion of the feature space, then the matching process will deliver fewer possible hypotheses. This will reduce the total time complxity of the algorithm since less time will need to be spent on hypothesis verification.

References

1 Grimson, W. L. and Lozano-Pérez, T.; "Recognition and Localization of Overlapping Parts from Sparse Data in Two and Three Dimensions", IEEE Conference Proceedings, CH2152-7/85, 1985.

2 Gottschalk, P. G., Turney, J. L., and Mudge, T. N.; "Two-Dimensional Partially Visible Object Recognition Using Efficient Multidimensional Range Queries", Accepted for publication in Proceedings of the 1987 IEEE Conference on Robotics and Automation.



Pi and Pj are best fit planes to the range data.

Figure 1