OFF-LINE PLANNING FOR ON-LINE OBJECT LOCALIZATION

Tomás Lozano-Pérez
W. Eric L. Grimson

MIT Artificial Intelligence Laboratory
Cambridge, MA 02139

Abstract. Many robot applications require using sensors to locate objects whose initial pose is constrained but not exactly known. Most techniques for object localization assume that the object's pose is completely unknown. This paper describes a simple method for localizing known objects in a scene. We describe how an off-line computation that exploits constraints on the object's expected pose can be used to reduce the expected time for the on-line computation to localize the object. The objects treated here are modeled as polyhedra that, in principle, can have up to six degrees of positional freedom relative to the sensors.

0. Introduction

The problems of object recognition and localization have received a great deal of attention (see [Jain 86, Grimson and Lozano-Pérez 84,85] for reviews of the literature). Most approaches to recognition assume that the object's pose is entirely unconstrained. In most practical robotics applications, however, the uncertainty in part location is bounded to relatively small ranges. These constraints may come from knowledge of the feeding mechanisms or the physics of part stability. In most recognition systems, it is difficult to incorporate these type of constraints on the initial object pose. There have been a few systems where such information is readily incorporated, but the methods themselves have tended to be fairly complex [Bolles 76, Brooks 81, Goad 83, Baird 85, Faugeras and Hebert 83]. Nevertheless, the approach described here was significantly influenced by these previous methods, especially Goad's excellent paper. The localization algorithm described here is quite simple as is the mechanism for incorporating any available constraints on object pose. In the absence of any global constraints, the algorithm will still work, albeit more slowly.

The specific problem considered in this paper is how to locate a known object in a cluttered scene using sensors that provide dense position information. We assume that worst-case bounds on the pose of the object are available, as well as bounds on sensor measurement error. Our goal is to exploit the known bounds on object pose so as to reduce the amount of on-line computation required to localize the object. An important subgoal is that both the on-line and off-line methods should be simple enough to be easily implemented.

The method described here can be applied to both two-dimensional and three-dimensional sensing situations. In the two-dimensional case, objects have only three degrees of positional freedom relative to the sensor (two translational and one rotational). In this case, the sensors (and their pre-processors) are assumed to compute edges, that is, line segments in the scene. In the three-dimensional case, objects have up to three translational and three rotational degrees of freedom. In this case, the sensors (and their pre-processors) are assumed to compute planar patches in the scene. We do *not* deal with the general case in which only two-dimensional data is available but the object has more than three degrees of freedom. For the sake of brevity, we limit our discussion to the three-dimensional case; the specialization to two-dimensions is straightforward.

We assume that the objects of interest can be modeled as sets of planar faces. Only the individual plane equations and dimensions of the model faces are needed. No face, edge, or vertex connectivity information is required; the model faces do not even have to be connected. Because of this, the method can be applied to curved objects that are readily approximated by planar patches. Of course, such planar approximations are not adequate for all cases, for example, objects of high curvature or multiply curved surfaces.

We assume the availability of a sensor and pre-processor that can compute the planar patches present within some given rectangular sub-window of the scene. A great deal of work in computer vision has been dedicated to solving this problem of obtaining depth from two-dimensional visual data (see [Horn 86] for a representative sample). Other less computationally-intensive methods exist for obtaining the required patches, notably range sensing (see [Jarvis 83] for a review). We will not address this problem further.

In section 1, we present the basic on-line localization method. In section 2, we describe the off-line computations required for the on-line method. In section 3, we discuss the method and point out areas for further work.

CH2345-7/86/0000/0138$01.00

1. A simple on-line localization algorithm

The process of localization is carried out in three steps:

- The first step is to identify possible assignments of sensed data to model faces consistent with a set of measurements derived from the model. This is the crucial step.
- The second step is to identify the pose of the object from each of these assignments.
- The third step is to pick the solution that best matches all the available data.

At this level of description, the method is similar to the *interpretation tree* method described in [Grimson and Lozano-Pérez 84, 85] and draws results from that earlier method. The method described in this paper differs in the first of these steps, while the earlier method does not do any hypothesis verification, the method described here goes very early into a hypothesize/verify cycle.

The current method is geared to situations where the set of possible matches of data patches to model faces can be constrained a priori. The goal is to reduce the combinatorics of the matching process in the earlier methods by exploiting the global position and orientation constraints.

1.1 Sensor Input

The on-line algorithm uses as input a list of the planar patches present in each of a set of windows specified by the off-line planner (see section 1.2). The patch must be completely within the window to be elegible. Each patch obtained by the sensor is characterized by a list of points and a plane equation in the form $\mathbf{n} \cdot \mathbf{x} = c$, where $\mathbf{n}$ is the unit normal to the plane. All the points are required to be inside the same face of the object. We represent patches by a set of points, instead of polygons, so as to simplify the processing and to accommodate a wide variety of sensors, including sparse sensors such as tactile sensors.

It is possible to use information about any patch feature, such as, size, color, reflectivity, and texture, to limit the possible model faces that a patch could match. The availability of these measurements can significantly reduce the combinatorics of the matching process. This is straightforward extension and we do not discuss it further.

1.2 Constraints from the off-line planner

The information from the off-line planner is used to reduce the combinatorics of the matching process. There are three sorts of constraints that are useful for this purpose: (1) restrictions on the data patches that can be involved in match a given model face, (2) restrictions on the model faces that can be involved in a solution, and (3) a ranked list of legal initial hypotheses. In particular, the off-line planner provides the following information for the on-line matching algorithm:

- A list of windows — each data patch is assigned to one or more rectangular windows in the scene.
- A list of matching constraints for each model face — each face requires that the patch that is matched to it be drawn from a specific window, furthermore there are constraints on the normal of the matching data patch.
- Ranges of possible measurements between pairs of model faces — these are used to check the consistency of assignments of patches to model faces.
- A list of visible face lists — these lists are indexed by the face pairs in the initial hypotheses and describe the faces visible if a particular hypothesis is correct.
- An ordered list of face pairs — these pairs are used to form initial hypotheses as to the object pose; they are ordered by size of the faces.

The windows could, in principle, be used to restrict the application of the sensor pre-processing to subsets of the scene. The windows, however, may overlap arbitrarily. In the general case, it is preferable to apply the sensor pre-processing to the whole scene and then assign the data patches to the corresponding windows. That is the strategy we have used. In simple situations where only a few windows are needed, it makes sense to limit the pre-processing to these windows. In any case, the main purpose of the windows is not to reduce the sensory pre-processing but to reduce the combinatorics of the matching process.

The key information computed by the off-line planner are the constraints on face assignments. Each face is restricted to matching a patch from a specific window. In fact, the windows are actually computed as the loci of particular model faces. The windows enforce the position constraints on faces derived from the global pose constraints. Associated with each face are constraints on the data patch normal that can match that face. This constraint is expressed as a list of cones, that is, center vectors and a minimum value for the cosine of the angle between the patch vector and the specified vector (see section 2). Also associated with each face is a window where the matching data patch must appear.

The off-line planner identifies those pairs of model faces that can appear in some view of the object consistent with the pose constraints. Of these pairs, the ones with significantly different orientation can be used to obtain an initial solution for the object's orientation. It is these pairs that drive the initial phase of the matching algorithm described in section 1.3.

The off-line planner also computes all the sets of model

faces that can be simultaneously visible given the known pose constraints. These are used during the verification phase of the matching algorithm.

The object model provided by the user is described by a set of planar faces. Each face is specified by a polygon, which may be non-convex, and a plane equation. This form of the model is not particularly useful to the matcher. The off-line planner computes from this model three tables described below. Let $\mathbf{p}_i$, $\mathbf{p}_j$ and $\mathbf{n}_i$, $\mathbf{n}_j$ be respectively points on and normals to faces i and j.

- Distance — For each pair of model faces, i and j, the upper and lower bounds on distances between all possible pairs of points $\mathbf{p}_i$ and $\mathbf{p}_j$.
- Angle — For each pair of model surfaces, the angle between the face normals $\mathbf{n}_i$ and $\mathbf{n}_j$.
- Distance vector — For each pair of model surfaces, the upper and lower bound on the values $(\mathbf{p}_j - \mathbf{p}_i) \cdot \mathbf{n}_i$ and $(\mathbf{p}_j - \mathbf{p}_i) \cdot \mathbf{n}_j$, that is, the component along the face normals of vectors connecting the faces.

In each case, the values in these tables take into account the error bounds on measuring both the patch normals the positions of points. Algorithms for computing these tables are found in [Grimson and Lozano-Pérez 84]. The combination of these three types of measurements has been shown to be quite powerful in discarding invalid matches even in the presence of significant measurement error [Grimson and Lozano-Pérez 84, 85].

1.3 The matching algorithm

The matching algorithm is very simple and is based on the assumption that the set of data patches that can match a given model face is relatively small (this is up to the off-line planner to guarantee). The method works by considering each of the possible model face pairs determined by the off-line planner as suitable for constructing an initial hypothesis. The first step is to find all the pairs of data patches that can be matched to the face pair under consideration. The data patch pair must satisfy the following conditions:

- Each data patch must come from the window specified by the off-line planner for the corresponding face.
- The measured normals for the data patches must be within one of the cones associated with the corresponding face.
- The three sets of measurements (distance, angle, and distance vector) for the data patch pair must be consistent with those of the model face pair.

We could find such data pairs by looking at all combinations of patches, but that would be time consuming. It is straightforward to improve the expected performance by using hashing. Simply pre-process all pairwise combinations of data patches and place them into buckets based on the angle between their normals. Then, given a candidate pair of model faces, one can in constant time limit the data patch combinations to those whose angle is within the measurement error of the angle between the model faces. Only these data pairs need to be subjected to further testing.

Having obtained the feasible matches to a model face pair, the next stage is to use them to obtain supporting evidence for particular combinations of visible faces. Note that the match of two data patch normals $(\mathbf{n}_i, \mathbf{n}_j)$ to two independent model face normals $(\mathbf{m}_i, \mathbf{m}_j)$ determines uniquely (up to a sign) the rotation matrix R, where $R\mathbf{x} + \mathbf{p}$ is the transformation that maps points, $\mathbf{x}$, in the model coordinate system to vectors in the sensor coordinate system.

The rotation matrix R can be easily computed in the form $Rot(\mathbf{r}, \theta)$, where $\mathbf{r}$ is the axis of rotation and θ is the rotation angle. The axis of rotation is the unit vector in the direction

$$(\mathbf{m}_i - \mathbf{n}_i) \times (\mathbf{m}_j - \mathbf{n}_j)$$

and the angle of rotation can be obtained from these two relationships

$$\cos\theta = 1 - \frac{1 - (\mathbf{n}_i \cdot \mathbf{m}_i)}{1 - (\mathbf{r} \cdot \mathbf{n}_i)(\mathbf{r} \cdot \mathbf{m}_i)}$$

$$\sin\theta = \frac{(\mathbf{r} \times \mathbf{n}_i) \cdot \mathbf{m}_i}{1 - (\mathbf{r} \cdot \mathbf{n}_i)(\mathbf{r} \cdot \mathbf{m}_i)}$$

For a detailed derivation see [Grimson and Lozano-Pérez 84].

Using R we can compute for each potentially visible face on the object, the nominal data patch normal that can match that face. We must take into account measurement errors when matching the predicted normal to actual measured normals.

The match also constrains the translation vector, $\mathbf{p}$, to be on a line determined by the planes of the two measured patches. The range of values of $\mathbf{p}$ can be determined as follows: Let the equation of the patch planes be of the form $\mathbf{n}_i \cdot \mathbf{x} = c_i$ and the equation of the model planes be of the form $\mathbf{m}_i \cdot \mathbf{x} = d_i$. Then we can write $\mathbf{p}$ as a one parameter family of vectors:

$$\mathbf{p} = \alpha\mathbf{n}_i + \beta\mathbf{n}_j + \gamma(\mathbf{n}_i \times \mathbf{n}_j)$$

where γ is the free parameter and

$$\alpha = \frac{(c_i - d_i) - (\mathbf{n}_i \cdot \mathbf{n}_j)(c_j - d_j)}{1 - (\mathbf{n}_i \cdot \mathbf{n}_j)^2}$$

$$\beta = \frac{(c_j - d_j) - (\mathbf{n}_i \cdot \mathbf{n}_j)(c_i - d_i)}{1 - (\mathbf{n}_i \cdot \mathbf{n}_j)^2}$$

The parameter γ can be constrained by requiring that all the points in the data patch be mapped by the resulting transformation to be *inside* the model face. In our implementation, we compute bounds on γ by considering the set of translations that map the center of area of the patch onto

each of the vertices of the face.

Let each potentially visible face have plane equation $\mathbf{m}_k \cdot \mathbf{x} = d_k$. Given the range of $\mathbf{p}$ we can compute not only the predicted normal, $R\mathbf{m}_k$, for a matching data patch but also the range of values of the offset c in the plane equation of the data patch: $c = d_k + (R\mathbf{m}_k) \cdot \mathbf{p}$, where $\mathbf{p}$ is parameterized by γ.

We also know, from the off-line computation, the window in which the data patch must appear. Thus, the initial verification process consists of checking the windows for patches satisfying the position and orientation constraints. Of course, once a third independent patch is found, the position and orientation can be determined uniquely up to the sensing error.

Once a feasible pairing of model faces and data patches, together with the corresponding object pose(s), is found there still remains a detailed verification process. This proceeds in two steps: (1) check that the patch points are all mapped to the inside of the face polygon by the computed transformation and (2) check that the patches measured in each of the windows are consistent with the computed model pose, that is, that no data patch has been measured to be *below* where the model predicts a surface to be. Note that if a data patch is above a predicted surface then this is neutral evidence, since this situation could arise from occlusion. Nevertheless, one wants to ensure that a hypothesis accounts for a sufficiently large percentage of the object's measured surface.

The matching process described above should be carried out in "depth-first" fashion, verifying each hypothesis as it is generated, rather than in "breadth-first" fashion, finding all the hypotheses and then verifying each one. Doing the matching depth-first allows us to terminate the matching once an adequate match is found. Observe that if the object were completely visible, each and every initial hypothesis should lead to a complete and correct interpretation. This points out the redundancy of examining all the initial hypotheses. In practice, face occlusion requires that we be ready to examine all the possible pairings even though we seldom will. Note that in the depth-first mode the ordering of the hypotheses by likelihood of locating the faces can significantly reduce the expected time to verify the hypothesis.

Let us quickly recap the flow of control in the on-line matcher. First, the sensing is done within a window guaranteed to contain all the faces. All the data patches are found and allocated to any windows that completely contain the patch. The matcher then proceeds through its list of legal face pairs attempting to form initial hypotheses and then verify them. Once an acceptable hypothesis is found, the matching stops. For each face pair, the matcher looks for a patch pair that has the appropriate angle between the normals. Each patch must be in the appropriate window for the matching face and satisfy the orientation constraint. Also, the assignment of the two faces to the two patches must pass all the pairwise constraints derived from the model. Only after all these test are satisfied does the matcher have a valid initial hypothesis. The next step is to verify the hypothesis. First, a potentially visible face is chosen, its orientation and range of displacements predicted and then its presence verified. If a matching patch is located, the pose of the object can be computed and all the other face predictions checked. A completeness score (based on correctly predicted area) is computed for the hypothesis.

1.4 Complexity

The matching method described above can be used with a single window and no off-line computation. In that case it has a worst-case complexity $O(n^3m^3)$ where n is the number of faces in the model and m is the number of data patches in the scene. This naive bound is simple to derive: The outer loop considers all n^2 permutations (with repetition) of the n model faces, for each such pair it tests for consistency all $m(m-1)/2$ pairwise patch combinations (without repetition). In the worst case, each pair of these $O(n^2m^2)$ combinations needs to be examined further. The further computation requires checking for each of the n faces of the model, each of the m data patches.

This bound for matcher performance is for the very worst case. The angle bucketing described above should improve the expected performance. In this version of the algorithm, one still has the $O(n^2)$ outer loop, but one expects significantly fewer than $m(m-1)/2$ data patches will have to be considered. Having found a pair of model faces and data patches, they can be tested for consistency with the distance, angle, and distance vector constraints. This will further reduce the combinatorics of the method. Of course, the actual performance will depend on the object model and the measured data. The worst performance will be for a very symmetric object such as a cube.

Thus far, we have not considered the effect of having a priori bounds on the object pose. In the absence of such bounds we are limited to using coordinate-frame independent constraints, such as angles between pairs of faces. Once we know bounds on the pose of the object, we can constrain individual matches. For example, suppose we knew that the object was in some particular stable pose on a horizontal table, then the z component of each face normal is known within the measurement error. Given a candidate model face, only data patches whose normals have z components in the appropriate range need be considered as matches. This reduces the effective number of data patches to be the expected number of data patches with the same z component.

The matching algorithm described in section 1.3 is aimed at exploiting this type of constraint. The windows enforce global position constraints and the angle cones associated with each model face enforce global orientation constraints. The purpose of these constraints is to minimize the number of data patches that can match a model face. In the ideal case, only a single patch can match each face and the whole algorithm boils down to finding three visible data patches.

These constraints, therefore, affect only the inner loop of the matcher; there still remains a potentially $O(n^2)$ outer loop. The algorithm attempts to reduce the expected number of initial hypotheses that need to be verified in two ways. The pre-processing of the pairs of model faces serves to capture other global constraints such as the fact that parallel faces cannot be used for the initial hypothesis and that not all pairs of faces can be visible simultaneously. These constraint tend to reduce the number of initial pairs well below the n^2 value. The constraints derived from the model (distance, angle, etc.) are used to prune out those initial hypotheses that remain before any detailed prediction and verification is done.

1.5 Example

Here, we consider a simple example of the matching algorithm; figure 1 outlines the stages of processing. Figure 1A shows three-dimensional depth data of a very cluttered scene obtained with a structured light range sensor, figure 1B shows the model of the target object given by the user, figure 1C shows the result of pre-processing the depth values to obtain planar patches, figure 1D schematically suggests the matching process of patches to faces, and figure 1E shows the resulting localized object superimposed on the original scene.

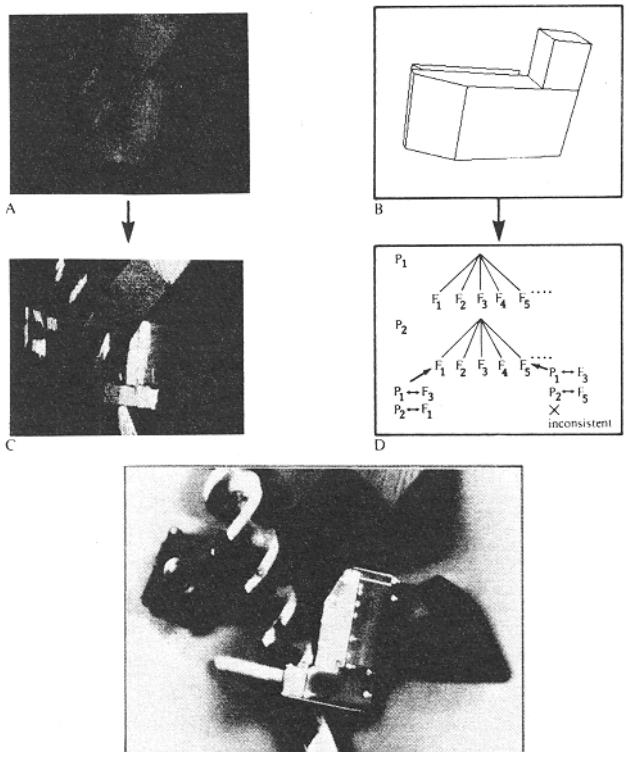

Figure 1.

Note that the matching algorithm can operate with no bounds on the pose of the input object. In that case, there is a single window within which all data patches can be found, also, there are no global constraints on the orientation of matching patches. The constraints that remain in effect are the pairwise matching constraints derived from the model, the pairwise visibility constraints, and the constraint that face pairs for the initial hypothesis have independent normals. Under those circumstances, the algorithm finds 865 legal initial hypotheses in the scene shown in figure 1. Recall that an initial hypothesis is an assignment of a pair of model faces to data patches that satisfy all the constraints. Note that for this example $n = 12$ and $m = 30$, so the worst case is 62,640 potential hypotheses. This illustrates that even in the absence of global pose constraints, the coordinate-frame independent constraints are quite powerful.

Continuing the example, assume that the target object's pose is constrained so that all the patches are within a smaller window (say with $m = 15$). This constraint reduces the number of potential initial hypotheses by roughly a factor of four (15,120). Furthermore, assume two specific faces can be constrained to smaller sub-windows (see figure 2), each with at most five patches in it ($m = 5$). Then,

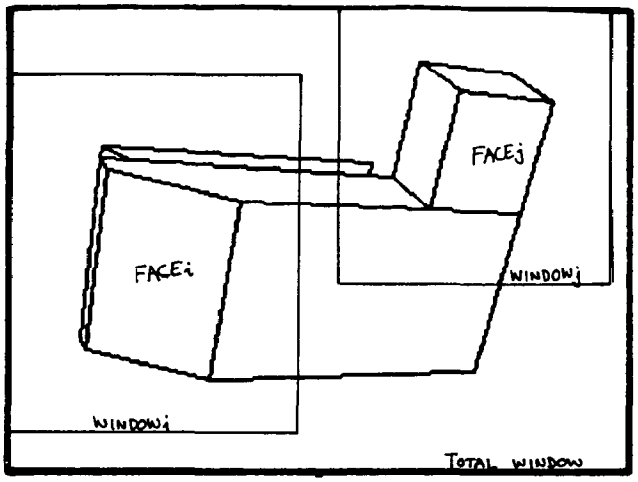

Figure 2.

the upper bound on the legal number of initial hypotheses, even without considering the other geometric constraints is cut by approximately a third (10,540). We can also exploit the fact that the two front faces and two back faces can never be simultaneously visible, similarly the two top faces and the bottom face. This eliminates an additional

1260 possible hypotheses. All of these numbers address the worst case bound; as shown above the actual number of hypotheses is typically substantially less. In the example shown above, of the 865 legal pairwise interpretations, 642 satisfy the window constraints described above.

In addition to the window constraints, there are global face orientation constraints that reduce the expected number of patches that can be assigned to the faces. These constraints will effectively reduce the value of m within the windows. As we saw above, reducing m produces large reductions in the possible number of hypotheses. For example, if the expected value of m can be uniformly reduced to 5, then the upper bound on the hypotheses (without the visibility constraint) is only 1040. The visibility constraint prunes an additional 120 potential hypotheses. The actual implementation actually had to consider 266 initial hypotheses.

2. The off-line planner

The efficiency of the off-line planner is not as crucial as that of the matching algorithm. Accuracy is not essential either; the only requirement is that the bounds be conservative. Therefore, we have adopted essentially brute-force sampling techniques for all the computations of the off-line planner.

2.1 Windows for the faces

Windows can be computed as rectangular bounds on the loci of the face's vertices over the range of legal poses. That is, the legal orientations of the object are sampled and the position of each face's vertices are computed. An enclosing rectangle is computed for the vertices of each face and updated so as to include all the point positions. This rectangle is then swept over the range of legal x, y translations of the object (also a rectangular range).

This computation generates a rectangular window for each face. Windows that overlap by some large fraction of their area are merged to form a single window. This is done to reduce the amount of computation that the sensory pre-processor needs to do when it assigns patches to windows. A window that contains all the other windows is also computed; this window bounds the area where sensor processing needs to be done.

2.2 Face orientation constraints

The global orientation constraints on the faces are represented as a set of cones for ease of testing. This can also be computed while sampling the range of object orientations. We proceed by tesselating the Gaussian sphere into the faces of an icosahedron. This gives us twenty uniformly distributed orientation buckets, represented as cones whose center vectors are the normals of the iscosahedron's faces. As we sample the orientations of the object, we keep track of which of these buckets the normal of each face falls in. This list of buckets is a conservative representation of the constraint on the orientation of the data patch that can match a face.

2.3 Visible faces

Potential visibility is determined by examining the sign of the component of the face normal along the sensing direction. This is what is known in graphics as "back face" elimination, rather than a full visible surface computation. The combinations of visible faces are also computed as the orientations of the object are sampled.

2.4 Face pairs

The planner constructs the list of all pairs of simultaneously visible faces whose normals make an angle greater than a predefined threshold ($\pi/6$ in our case). The list is ordered so that pairs involving large faces are at the front of the list. These are the most likely faces to be visible in spite of occlusion.

2.5 Pairwise geometric constraints

See [Grimson and Lozano-Pérez 84] for a description of how these constraints can be computed for polyhedral models.

3. Discussion

The algorithm presented in this paper is primarily based on a prediction/verification style: predicting the orientation and positions of faces in the model and verifying the presence of consistent data patches. The hypotheses are driven off of pairwise matches of faces to patches and it is the number of possible matches at this level that determines the performance of the method. In the unconstrained case, the algorithm needs to examine a large number of initial hypotheses. By considering some simple constraints arising from global constraints on the object pose, the number of legal initial hypothesis is significantly reduced to a manageable number. Recall that the actual prediction and verification process is already reasonably efficient so that hundreds of hypotheses can be evaluated in a matter of seconds.

Initial indications are that the method performs extremely well when the pose of the object is tightly constrained. As the range of possible poses grows, the performance reaches a plateau dictated by the performance of

the algorithm in the absence of constraints. Of course, this limiting performance is a non-linear function of the number of faces and number of data patches. Future work will attempt to derive better expected bounds on the performance of the algorithm.

The main advantage of the method described here is its simplicity. The major limitation of the algorithm is its reliance on planar face approximations; extending the approach to curved objects would not be straightforward. Another disadvantage is the relatively weak coupling between the sensor processing stage and the matching. In principle, a tighter coupling could be implemented so that the amount of sensor processing could be reduced. On the other hand, the simplicity of the algorithm is due largely to the fact that the pre-processing is simple and uniform.

The on-line matching algorithm described in section 1 has been implemented on a Symbolics Lisp Machine; the off-line planner is currently being implemented. The testing of the on-line method has been done with simple window and orientation constraints specified by the user.

Acknowledgments

We thank Philippe Brou for kindly providing the laser ranging system with which we obtained the data reported in figure 1. This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's Artificial Intelligence research is provided in part by a grant from the System Development Foundation, and in part by the Advanced Research Projects Agency under Office of Naval Research contracts N00014-80-C-0505 and N00014-82-K-0334.

Bibliography

Baird, H. 1986. *Model-based recognition.* MIT Press, Cambridge, Ma.

Bolles, R. C. 1976. Verification vision within a programmable assembly system. Stanford Artificial Intelligence Laboratory Memo 295.

Brooks, R. A. 1981. Symbolic reasoning among 3d models and 2d images. *Artificial Intelligence.* 17(1-3):285–348, August.

Faugeras, O. D. and Hebert, M. 1983. A 3D recognition and positioning algorithm using geometrical matching between between primitive surfaces. *Proc. Eigth Int. Joint Conf. on Artificial Intelligence,* Karlsruhe, W. Germany, 996–1002, August.

Goad, C. 1983. Special purpose automatic programming for 3d model-based vision. in *Proceedings of DARPA Image Understanding Workshop.*

Grimson, W. E. L., and Lozano-Pérez, T. 1984. Model-based recognition and localization from sparse range or tactile data. *Int. J. Robotics Res.* 3(3):3–35.

Grimson, W. E. L., and Lozano-Pérez, T. 1985. Recognition and localization of overlapping parts from sparse data in two and three dimensions. *Proc. IEEE Conf. on Robotics and Automation,* St. Louis, Mo., 61–66, March.

Horn, B. K. P. 1986. *Machine Vision.* MIT Press, Cambridge, Ma.

Jarvis, R. A. 1983. A perspective on range finding techniques for computer vision. *IEEE Trans. on Pattern Analysis and Machine Intelligence.* 5(2):122–139, March.

Jain, R. 1986, Three-dimensional object recognition. *Computing Surveys.*