Pose Determination using Octrees: A Comparative Survey

Régis Houde[1], Jacques Tremblay[2], Frédéric Auclair[2] and Denis Laurendeau[2]

[1]Division robotique
Institut de recherche d'Hydro-Québec
Varennes (Québec), Canada, J3X 1S1

[2]Laboratoire de vision et systèmes numériques
Département de génie électrique
Université Laval, Québec, Canada, G1K 7P4

Abstract

This paper presents a comparative survey of some pose determination techniques with Octrees as the volumetric representation of data. The approaches compared are: the Projection onto the Principal Planes method; the Potential Field method; the Geometric Hashing method and the Iterative Closest Point method. Particular attention is given to the 3-D Potential Field approach since it is presented here for the first time. Pointers to other pose determination related papers are also given.

1. Introduction

In many robotic applications we encounter the problem of finding the position and orientation of objects that a robot might grasp, avoid or interact with. This process is called pose determination and consists of finding the rigid transformation between a canonical position of an object and its actual position given a 2-D or 3-D representation of the scene. Pose determination is a sub-problem of the more general problem of object recognition which is not covered in this paper although some of the techniques presented could be used to do so. The interested reader is referred to [6] and [14] for references on object recognition.

Previously, a number of different approaches have been proposed to solve the pose determination problem and with the emergence of new 3-D sensors, there are still numerous research activities going on. Model based approaches, which are very popular, try to match features detected in the image with corresponding features on a model. These models are typically simple and do not encode a high-level representation of the objects. The major challenge in the model-based pose estimation is the matching process. An error at this stage can lead to poor or erroneous pose estimate.

Grimson and Lozano-Perez have introduced the interpretation tree paradigm for object recognition and localization [19]. Indeed, many approaches use an interpretation graph to discard inconsistent pairings ([1], [21], [31], [34]). One serious limitation with this approach is that the objects are assumed to be isolated [41]. The interpretation tree might also becomes huge if many models are to be taken into account. Methods based on the generalized Hough Transform ([4] and [16]) are also popular for object recognition and pose determination ([5], [8], [37], [29]).

The problem of achieving high speed pose determination is of interest to many researchers ([33], [40], [30], [38]). Due to the lack of high speed range sensors, most of the work has been done with 2-D data (except for [38]). Two-dimensional data are less sensitive to motion along the optical axis thus accurate pose estimation is not always possible with conventional 2-D cameras.

This paper deals with the problem of pose determination when the volumetric representation used for the objects is an Octree. In order to compare different approaches, one must consider the context in which they will be used. In this paper, we are mostly interested in pose determination of known objects in cluttered scenes. We assume that an operator, say in a telerobotic application, could suggest a fairly good initial guess. We are therefore more interested in pose refinement and speed than coarse matching and object recognition *per se*.

Section 2 describes the building and merging of Octrees and some of their interesting properties. A survey of 4 different pose determination methods is made in Section 3 while the comparison of these methods is presented in Section 4.

2. Volumetric Representation

In this paper, we suppose that the information about the environment comes from a 3-D range finder. Because of its compactness and its hierarchical structure, the Octree scheme was chosen to gather data acquired from different points of view. This volumetric representation was first proposed by Jackins and Tanimoto [27] as the three-dimensional analog of Quadtrees. The compact tree-like structure permits an efficient traversal algorithm to be applied for fast pose determination. The Octree inherent hierarchy is ideal

for robotic tasks where different resolutions are often needed. For example, unlike object grasping, collision avoidance algorithms do not require huge quantities of data.

2.1 Building the Octree

In order to build an Octree, we demarcate a first cube, called the *root*, around the scene of interest. If the space occupancy is uniform throughout the cube, the construction stops and an identification tag is assigned to this voxel. If it is not, the cube is divided into eight smaller voxels of same size called *branches*. Each new voxel can also be divided into branches or set to a tag. The Octree generation can stop at any desired resolution but it is pointless to go beyond the sensor accuracy or the task requirement. Figure 1 illustrates the hierarchical structure of an Octree. The example shown comes from a typical electricity distribution structure.

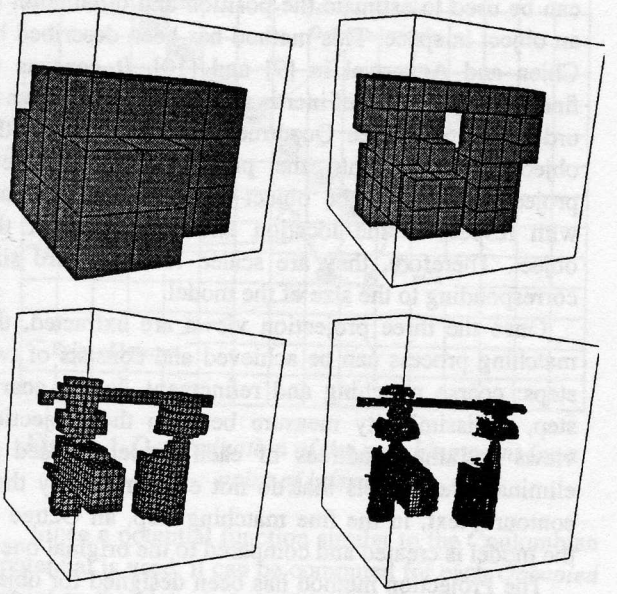


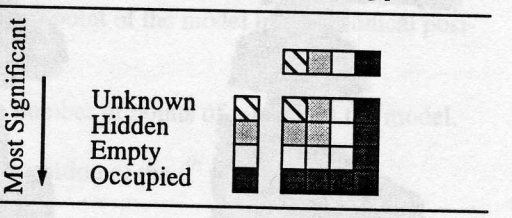
Figure 1. Hierarchical structure of Octrees

2.2 View Integration

During the construction of an Octree, four different tags can be assigned to the voxels. Those tags are: *Unknown*, *Empty*, *Occupied* and *Hidden*. An *Unknown* tag is assigned to a voxel which is out of the field of view of the sensor while the *Empty* tag is given to a voxel located between the sensor and a measured point (*Occupied* tag). Finally, voxels that are located behind an *Occupied* voxel get the *Hidden* tag.

In the integration of the data when two Octrees are merged, corresponding voxels do not have necessarily

Table 1. Rule of precedence in the merging process



the same label. In such case, we apply a precedence rule based on safety considerations and logic. The *Occupied* tag is the most significant since unsetting this tag could result in a collision. The *Unknown* label must be the least significant, since any information is better than none. Finally, the *Empty* label has precedence on the *Hidden* one because this last tag is not really measured but inferred. Table 1 summarizes the precedence rule while Figure 2 gives an example of the merging process for two different points of view.

Figure 3 shows an Octree obtained from real data. The scene is a typical electricity distribution structure. It is formed of different objects such as insulators, cables, wooden poles and cross-arms connected together. All the *Unknown* tags have disappeared while the inside of objects are filled with *Hidden* voxels. This information could be used in the pose determination process.

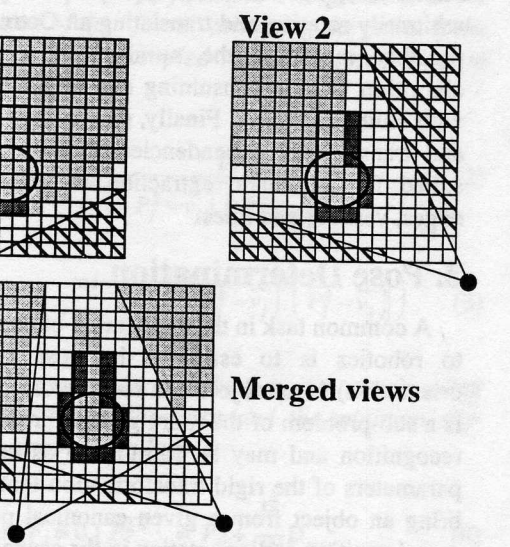


Figure 2. Merging of 2 different views

2.3 Characteristics of Octrees

Octrees have some advantages over other volumetric representations. The memory requirement is low and the location of voxels can be found rapidly. Octrees are convenient for the representation of irregular objects, even those with cavities. They permit efficient

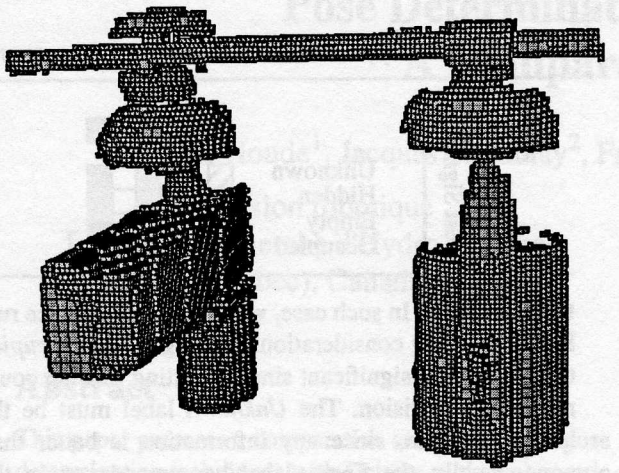


Figure 3. Octree of a typical electricity distribution structure

execution of boolean operations such as union which are useful for view integration. Furthermore, due to their hierachical structures, Octrees are suitable for recursive and tree-traversal algorithms as well as for tasks that might require different levels of precision.

On the other hand, Octrees do not provide an exact representation of objects (see [36], [32] and [13] for ways to overcome this problem) and their sizes depend on the position and orientation of the object in the scene. Hong and Shneier [22] proposed an algorithm for arbitrarily rotating and translating an Octree in order to possibly minimize the number of nodes but this algorithm is time consuming and might not be useful with cluttered scenes. Finally, due to their orientational and translational dependencies, Octrees are not well suited for automatic extraction of features such as edges, vertices and lines.

3. Pose Determination

A common task in the application of computer vision to robotics is to estimate the pose (position and orientation) of an object in a scene. Pose determination is a sub-problem of the more general problem of object recognition and may be formulated as computing the parameters of the rigid transformation that is needed to bring an object from a given canonical position to its actual position and orientation in the scene.

Pose determination is different from the problem that consists of only finding the transformation parameters between two set of points already matched. This problem is sometimes called absolute orientation [39] and many solutions have already been proposed. Iterative [26] and non-iterative techniques using quaternions ([17] and [24]) have been presented. Other solutions based on singular value decomposition of a covariance matrix of the data have been developed in [2], [23] and [20].

Previously, a number of different approaches have been proposed to solve the pose determination problem. These approaches differ in the number and type of assumptions made to reach a solution. Though many approaches could be adapted to Octrees, few researchers have proposed methods designed to work specifically with that representation. Chien and Aggarwal have made the most significant work in that field and the next section relates part of their work. Section 3.2 describes a method first proposed by Houde *et al.* [25] for Quadtrees and adapted to Octrees. Section 3.3 presents an approach based on Geometric Hashing while Section 3.4 describes the Iterative Closest Point algorithm.

3.1 Projection onto the Principal Planes

The projection of an object onto its principal planes can be used to estimate the position and orientation of an object in space. This method has been described by Chien and Aggarwal in [9] and [10]. It consists of finding the moments of inertia matrix from the Octree in order to create three Quadtree representations of the object projected onto the principal planes. These projections need to be object centered and invariant with respect to the location and orientation of the object. Therefore, they are scaled to a standard size corresponding to the size of the model.

Once the three projection views are extracted, the matching process can be achieved and consists of two steps: coarse matching and refinement. In the coarse step, a dissimilarity measure between the projection views and the quadtrees of each model is used to eliminate the models that do not even match by their contour. Next, in the fine matching step, an Octree of the model is created and compared to the original one.

The Projection method has been designed for object recognition and not pose determination. A boolean operation (X-OR or AND) is used for the refinement stage to reject models but not to increase the accuracy of the transformation matrix obtained in the coarse stage.

3.2 Potential Fields

The Potential Field (PF) method was first proposed by Houde *et al.* [25] for 2-D pose determination. This section presents the 3-D extension of the technique. As for the original method, a positive charge is associated to each *Occupied* voxel of the Octree while a negative charge is given to every point in the model. The operator brings the model towards the object in the

scene and the artificial potential field surrounding the voxels exert a force on the model that brings it over its actual position.

The implementation consists of performing a gradient descent on the potential function to move the model towards the minimum energy position of the scene. To achieve this goal, for each point of the model, the program finds a 3-D point located at a distance D_{min} in the direction of the gradient of the potential function where D_{min} is the distance of the closest point of the scene involved in the computation of the force. These points are called «matching points» and do not necessarily exist in the scene. This concept is shown on Figure 4. Then we find, using the method described in [20], the best rigid transformation that will bring the points of interest in the model as close as possible to their matching points.

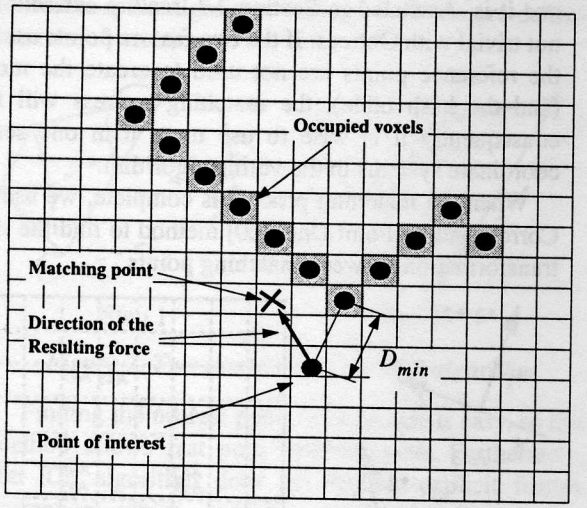


Figure 4. Determination of the matching point for a point of interest

Since a potential function similar to the Coulombian potential is used, it can be computed for each *Occupied* point in the Octree representation. In order to find the attraction force on a point in the model, the program just sums the contribution of each point in the scene. The directing force is given by the direction of the sum.

This method reaches convergence very fast as can be seen in the example of Figure 5 where stable state was almost reached in two iterations. Total convergence was reached in five iterations. The starting position of the descent has to be near the solution because the points of the model can be attracted by other objects in the scene.

3.2.1 Algorithm

Before describing the algorithm in detail, we define some notations used in this section.

- $\vec{P_i^k}$: The ith point of the model after the k^{th} iteration.
- $\vec{P_i^0}$ The ith point of the model in its canonical position.
- n: The number of points of interest of the model.
- $\vec{v_i}$: The position of the i^{th} *Occupied* voxel.
- N: The number of Occupied voxels.
- A^k: The rigid transformation after the k^{th} iteration.
- A^0: The rigid transformation corresponding to the initial estimate of the position.

Here is the algorithm in pseudo code:

Step 1. Bring the points of model close to the object in the scene.

$$P^k = A^k P^0 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 & x_1 & x_2 & \cdots & x_{n-1} \\ y_0 & y_1 & y_2 & \cdots & y_{n-1} \\ z_0 & z_1 & z_2 & \cdots & z_{n-1} \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix} \quad (1)$$

Step 2. For each point of interest compute the gradient of the potential function. The gradient is the direction of the "force" attracting the point. If the point is in an Occupied voxel, the force is set to zero.

$$\vec{F_i} = \sum_{j=0}^{N-1} \frac{1}{\left(\vec{P_i^k} - \vec{v_j}\right)^T \left(\vec{P_i^k} - \vec{v_j}\right)} \quad (2)$$

$$D_{min} = \min_{j \in \{1, \ldots, N\}} \left(\left| \left(\vec{P_i^k} - \vec{v_j}\right)^T \left(\vec{P_i^k} - \vec{v_j}\right) \right| \right) \quad (3)$$

Step 3. Compute the matching points matrix using the direction of the force and the minimum distance computed in Step 2.

$$\text{if } (\vec{F_i} \neq 0)\ \overrightarrow{P_i^{k+1}} = \vec{P_i^k} + \frac{\vec{F_i}}{\left|\vec{F_i}\right|} D_{min} \quad (4)$$

$$\text{else } \overrightarrow{P_i^{k+1}} = \vec{P_i^k} \quad (5)$$

Step 4. Find the best rigid transformation B:

$$B\vec{P^k} = \overrightarrow{P^{k+1}} \quad (6)$$

Step 5. Compute the new transformation matrix:

$$A^{k+1} = BA^{k} \tag{7}$$

Step 6. Increment k and go to Step 1 unless:

$$\sum_{j=0}^{3}\sum_{i=0}^{3}(B_{ij}-I_{ij})^{2} < \varepsilon \tag{8}$$

where I is the identity matrix and ε is the tolerance factor. As the algorithm reaches convergence, B becomes very close to an identity matrix.

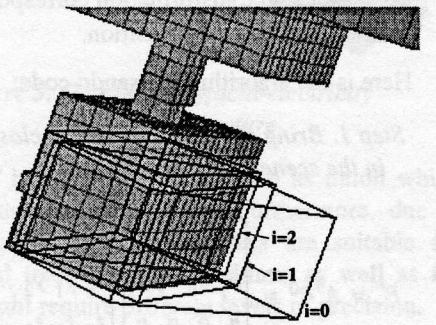


Figure 5. Two iterations of the pose refinement approach using PF

As described here, the algorithm does not take into account the hierarchical structure of the Octree. Gain in speed could be made by refining the pose at a lower resolution then increasing the resolution until the best resolution is reached. When working at a lower resolution, the charge of each voxel is computed by taking the sum of all *Occupied* voxels located below the node. The attractive force is thus weighted by the density of *Occupied* voxel within a region.

3.3 Geometric Hashing

The Geometric Hashing method first proposed by Lamdan, Schwartz and Wolfson [28] is based on intensive off-line model preprocessing that allows fast point-to-point matching. This method uses a pre-computed hash-table constructed from all possible models and built using minimal transformation invariant features. Geometric Hashing can be used for both object recognition and pose determination.

Figure 6 gives an example of the construction of a simple hash-table in 2-D space. Given a model A, we can arbitrarily select two points and express all the other points according to a coordinate system based on those two points. For example, if points 4 and 5 are used as the coordinate system, then the model can be rotated and scaled as to make those two points fit in squares C6 and F6 (which are randomly chosen to contain the base points). Point 1 will be registered in square D5 which will get the label A45 indicating that a point of the model A corresponds to that square when expressed upon the Base 4 and 5. In the same way, Points 2 and 3 will set the label A45 to squares E3 and F3 respectively. This exercise is repeated for all possible sets of two points for each model. Figure 6 gives the example for Bases A45 and A15. Though this computation is complex, it is only done once and off-line for each model.

Once the hash table has been created, the matching process is achieved by choosing two arbitrary feature points in the scene and expressing all the other feature points according to that base. This new coordinate system should correspond to a base already registered in the hash table and a voting algorithm determines which one it is. As stated in Section 2.3, feature extraction is not trivial with Octrees. If the two feature points used as the reference points are not used to create the model (and the hash table), the matching process will fail, consequently it is wise to use more than one set of coordinate systems in the voting algorithm.

When the matching process is complete, we use the Corresponding Point Data [20] method to find the rigid transformation between matching points.

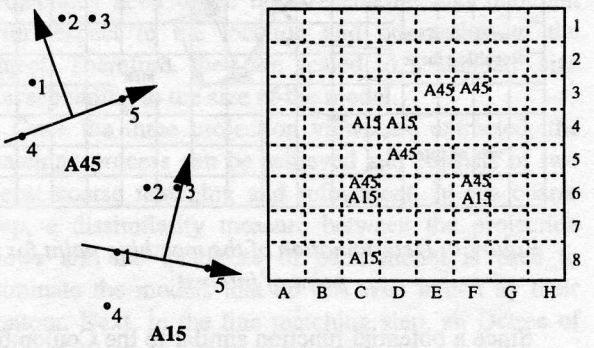


Figure 6. Hash table creation

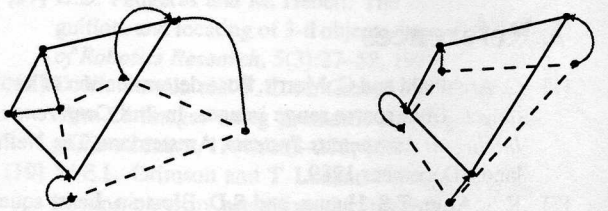
For more details about the Geometric Hashing technique, the reader is referred to [28], [35] and [18].

3.4 Iterative Closest Point

The Iterative Closest Point (ICP) algorithm was first introduced by Besl and McKay [7] while Simon *et al.* [38] proposed some speed enhancements allowing real time pose determination. They highlighted the fact that the most expensive part of the original ICP algorithm is searching for the closest point sets. In order to speed up the algorithm, a tree-like (kd-tree) representation of the data was used. The use of the ICP algorithm, as will be shown, is thus well indicated with Octrees.

The method consists of matching each point of the model with the closest point in the real data set. Then, a 3-D transformation minimizing a distance measure between the two points set is applied to the model. Again, different methods may be used (see Section 3 for a few pointers). The first two steps are iteratively repeated until a steady state is reached. Figure 7 shows a graphical 2-D example of the ICP algorithm. The dashed polygon represents the model.

Despite the fact that one point is mismatched in Step 1, the model moves towards the object in the scene and the matching process succeeds in Step 2. To check that the pose determination is over, or caught in a local minima, all that is required is to measure the translation and rotation component of the transformation matrix and stop if they are under a set threshold. To avoid being trapped in an endless loop around local minimums, the algorithm also stops after a pre-defined number of iterations.



Step 1 Step 2

Figure 7. Two iterations of the ICP algorithm

Finding the nearest point in an Octree is easy, so this method allows fast pose determination. Furthermore, the ICP algorithm does not require explicit feature extraction which reduces the level of operator's intervention.

It is clear that the initial guess has to be fairly good especially when there is more than one object in the scene. This will be discussed more in Section 4.

4. Comparison of Methods

Before making any comparisons between each of the approaches described in this paper, let us restate our goals and assumptions. First, we are interested in finding the location of known objects in cluttered scenes. We assume that an operator could suggest a fairly good initial estimate. Therefore we are more concerned by pose refinement and speed rather than object recognition *per se*. The goal in terms of speed is not exactly «real time» but something reasonable for a operator to wait for like few seconds on a standard workstation (no specialized hardware).

The first method described in the paper (Projection onto the Principal Planes) was rejected and not tested because it was clear that it could not be applied when multiple objects would be present in the scene. Although Chien and Aggarwal proposed a variant to this method [12] allowing recognition from occluding contours, we suspect that the enhanced method would still fail with more complex scenes than those presented in their paper. We also suspect their approach to be quite slow even with their algorithm that speeds up the computation of the moment of inertia matrix by taking advantage of some of the inherent characteristics of Octrees [11].

The three other methods were implemented and tested with simulated and real data. An example of the kind of scenes that were used for testing the algorithm is shown in Figure 3. It is an Octree mapping the space occupancy of a typical electricity distribution structure. For all the methods, the same models were used. Those models were composed of points that could be located anywhere on the objects' surfaces, edges or along the vertices. For all the techniques studied, the extraction of features was done by the operator. Using Octrees, automatic feature extraction on real data was shown to be unreliable and difficult to achieve.

For the Geometric Hashing technique, the operator was asked to select at least four easily identifiable points in the scene. Three of these points were used to create the reference base and the other points were used in the voting algorithm. The points that form the coordinate system will not necessarily lead to an orthonormal base, thus a space scaling is required. The performance of the matching step will be affected by this non-orthonormality, so in the practical implementation, an algorithm determines the three points that are the most «orthonormal to each other». This was not always possible especially for long objects such as cables. To circumvent this problem these kinds of object were treated with a particular algorithm but this model dependent solution decreases the generality of the method. See [3] for more details on the implementation.

Another problem with the Geometric Hashing approach is the difficulty for an operator to select points in the scene that correspond to those in the model, especially with real data. Choosing a reference base that does not exist in the hash table leads to ludicrous solutions.

In conclusion, despite the fact that Geometric Hashing was the fastest technique we tested, it appeared unpractical in our context.

Table 2: Speed comparison between methods

Method	1752 Occupied voxels	3127 Occupied voxels	15847 Occupied voxels
PF	5 seconds	14 seconds	31 seconds
ICP	2 seconds	7 seconds	9 seconds

The ICP algorithm and the Potential Field approaches showed to be the most convenient for our needs. Both methods do not need exact feature extraction and give good results even with noisy data. The ICP algorithm was faster than the PF method as seen in Table 2. These algorithms were implemented on a Sun workstation (Sparc 20). For small Octrees, these methods were fast enough to be useful but as the size of data grew, the PF method became unpractical. This phenomenon is related to the fact that a fast tree-traversal algorithm was used to find the nearest point in the ICP algorithm.

On the other hand, the ICP algorithm was more prone to get trapped in local minima. Figure 8 shows two examples of situations where the ICP algorithm fails because of its shortsighted behavior. The algorithm stops after Step 2 since the matching has not changed even if there exists a better solution. The PF approach will succeed because it takes into account more points thereby eliminating small local minima.

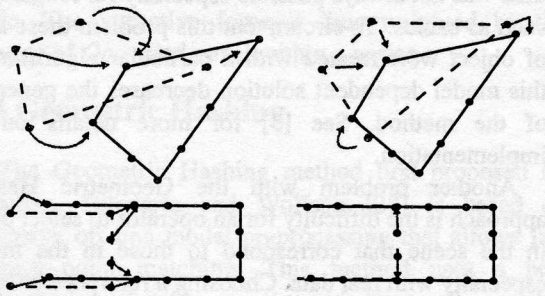


Figure 8. Two examples of failure of the ICP method over the potential field method

5. Conclusion

This paper described four different methods that can be used to find the pose of a model in an Octree. The Projection onto the Principle Planes Method and the Geometric Hashing Method reveal to be inadequate to fulfill our needs. The Potential Field approach and the ICP algorithm showed good results. The ICP algorithm was faster than the PF method but was more subject to get trapped in local minima. This problem can be circumvented by choosing a better initial guess.

6. Acknowledgments

The authors would like to thank Alain Croteau for the implementation of the Octree creation and manipulation library. We are also grateful to Jean Côté who implemented the algorithm for solving the absolute orientation problem. J. Tremblay was supported in part by a NSERC University Undergraduate Student Research Award. D. Laurendeau is a member of the Institute for Robotics and Intelligent Systems (IRIS) and wishes to acknowledge the support of the Network of Centres of Excellence Program of the Government of Canada, the Natural Sciences and Engineering Research Council, and the participation of PRECARN Associates Inc.

7. References

[1] C. Archibald and C. Merrit. Pose determination of known objects from sparse range images. In *2nd Conference on Intelligent Autnomous Systems*, Amsterdam, The Netherlands, December 1989.

[2] K.S. Arun, T.S. Huang, and S.D. Blostein. Least squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:698–700, 1987.

[3] Frédéric Auclair. Modélisation de l'encombrement de l'espace de travail d'un robot 'a partir de données 3d obtenues d'un capteur. Master's thesis, Université Laval, Faculté des sciences et de génie, Sainte-Foy, Québec, G1K 7P4, May 1993.

[4] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13:111–222, 1981.

[5] Dana H. Ballard and Christopher M. Brown. *Computer Vision*. Prentice Hall, Englewood Cliffs, New Jersey, 1982.

[6] P.J. Besl and R.C. Jain. Three-dimensional object recognition. *Computing Surveys*, 17:75–145, 1985.

[7] P.J. Besl and N.D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and machine Intelligence*, 14(2):239–256, February 1992.

[8] B. A. Boyter and J.K. Aggarwal. Recognition of polyhedra from range data. *IEEE Expert*, pages 47–59, spring 1986.

[9] C.H. Chien and J.K. Aggarwal. Reconstruction and matching of 3-d objects using quadtrees/octrees. In *Proceedings of the Third Workshop on Computer Vision: Representation and Control (Bellaire, MI, October 13–16, 1985)*, IEEE Publ. 85CH2248-3, pages 49–54. IEEE, IEEE, 1985.

[10] C.H. Chien and J.K. Aggarwal. Identification of 3d objects from multiple silhouettes using quadtrees/octrees. *Computer Vision, Graphics and Image Processing*,

36:256–273, 1986.

[11] C.H. Chien and J.K. Aggarwal. Volume/surface octrees for the representation of three-dimensional objects. *Computer Vision, Graphics and Image Processing*, 36:100–113, 1986.

[12] C.H. Chien and J.K. Aggarwal. Model construction and shape recognition from occluding contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:372–389, 1989.

[13] C.H. Chien, Y.B. Sim, and J.K. Aggarwal. Generation of volume/surface octree from range data. In CVPR1988 [15], pages 254–260.

[14] R.T. Chin and C.R. Dyer. Model-based recognition in robot vision. *Computing Surveys*, 18:67–108, 1986.

[15] *CVPR'88 (IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Ann Arbor, MI, June 5–9, 1988)*, Washington, DC., June 1988. Computer Society Press.

[16] L. S. Davis. Hierarchical generalized hough transforms and line-segment based generalized hough transforms. *Pattern Recognition*, 15:277–285, 1982.

[17] O.D. Faugeras and M. Hebert. The representation, recognition, and locating of 3-d objects. *International Journal of Robotics Research*, 5(3):27–52, 1986.

[18] D. M. Gavrila and F. C. A. Groen. 3d object recognition from 2d images using geometric hashing. *Pattern Recognition Letters*, 13(4):263–278, 1992.

[19] W.E.L. Grimson and T. Lozano-Perez. Localizing overlapping parts by searching the interpretation tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:469–482, 1987.

[20] R.M. Haralick, C.N. Lee, X. Zhuang, V.G. Vaidya, and M.B. Kim. Pose estimation from corresponding point data. In *Workshop on Computer Vision, (Miami Beach, FL, November 30 – December 2, 1987)*, pages 258–263, Washington, DC., 1987. IEEE Computer Society Press.

[21] Ki Sang Hong and Kyung Nam Kim. Recognition strategy generation for pose estimation of multiple 3-dimensional objects. In IEEE Computer Society Press, editor, *IEEE International Conference on Pattern Recognition*, volume 1, pages 612–615, The Hague, The Netherlands, August 1992.

[22] Tsai-Hong Hong and Michael O. Shneier. Rotation and translation of objects represented by octrees. In *IEEE International Conference on Robotics and Automation*, pages 947–952, Raleigh, NC, USA, March 1987.

[23] B. K. P. Horn. Closed-form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society of America A*, 5(7):1127–1135, 1987.

[24] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629:642, April 1987.

[25] Régis Houde, Jacques Tremblay, Denis Laurendeau, and Michel Pelletier. Estimating the pose of 2-d objects using potential functions and quadtrees. *Vision Interface '93*, pages 15–20, May 1993.

[26] T.S. Huang, S. D. Blostein, and E. A. Margerum. Least-squares estimation of motion parameters from 3-d point correspondences. In *IEEE Conference in Computer Vision and Pattern Recognition*, pages 24–26, Miami Beach, Florida, 1986.

[27] Chris L. Jackins and Steven L. Tanimoto. Oct-trees and their use in representing three-dimensional objects. *Computer Graphics and Image Processing*, 14:249–270, 1980.

[28] Y. Lamdan, J.T. Schwartz, and H.J. Wolfson. Object recognition by affine invariant matching. In CVPR1988 [15], pages 335–344.

[29] Seppo Linnainmaa, David Harwood, and Larry S. Davis. Pose determination of a three-dimensional object using triangle pairs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):634–647, September 1988.

[30] David G. Lowe. Real-time systems for tracking articulated three-dimensional objects. In *Vision Interface 91*, pages 42–48, Calgary, Alberta, June 1991.

[31] D.W. Murray. Model-based recognition using 3d structure from motion. *Image and Vision Computing*, 5:85–90, 1987.

[32] I. Navazo, D. Ayala, and P. Brunet. A geometric modeller based on the exact octtree representation of ployhedra. *Computer Graphics Forum*, 5(2):91–104, June 1986.

[33] N. P. Papanikolopoulos, B. Nelson, and P. K. Khosla. Full 3-d tracking using the controlled active vision paradigm. In *Proceedings of the IEEE International Symposium on Intelligent Control*, Glasgow, Scotland, UK, August 1992.

[34] S.B. Pollard, J. Porrill, J.E.W. Mayhew, and J.P. Frisby. Matching geometrical descriptions in three-space. *Image and Vision Computing*, 5:73–78, 1987.

[35] Isidore Rogoutsos and Robert Hummel. Massively parallel model matching - geometric hashing on the connection machine. *Computer*, 25(2):33–42, February 1992.

[36] W. J. Schroeder and M. S. Shephard. A combined octree/delaunay method for fully automatic 3-d mesh generation. *International Journal for Numerical Methods in Engineering*, 29(1):37–55, January 1990.

[37] T. M. Silberberg, D. A. Harwood, and L. S. Davis. Object recognition using oriented model points. *Computer Vision Graphics and Image Processing*, 35:47–71, 1985.

[38] David A. Simon, Martial Hebert, and Takeo Kanade. Real-time 3-d pose estimation using a high-speed range sensor. *IEEE International Conference on Robotics and Automation*, 3:2235–2241, May 1994.

[39] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transaction on Pattern Analysis and machine Intelligence*, 13(4):376–380, April 1991.

[40] J. Wang and W.J. Wilson. 3d relative position and orientation estimation using kalman filter for robot control. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2638–2645, Nice, France, May 1992.

[41] Harry Wechsler. *Computational Vision*. Academic Press, 1990.