Handey: A Robot System that Recognizes, Plans, and Manipulates

Tomás Lozano-Pérez, Joseph L. Jones, Emmanuel Mazer*, Patrick A. O'Donnell, W. Eric L. Grimson
MIT Artificial Intelligence Laboratory, USA

Pierre Tournassoud
INRIA, France

Alain Lanusse
E.T.C.A., Arcueil, France

* On leave from LIFIA, Grenoble, France.

Abstract. We describe a robot system capable of locating a part in an unstructured pile of objects, choose a grasp on the part, plan a motion to reach the part safely, and plan a motion to place the part at a commanded position. The system requires as input a polyhedral world model including models of the part to be manipulated, the robot arm, and any other fixed objects in the environment. In addition, the system builds a depth map, using structured light, of the area where the part is to be found initially. Any other objects present in that area do not have to be modeled.

1. Introduction

The word robot should conjure up the image of a system with (at least) three generic capabilities:

- The ability to perceive its environment and to locate objects of interest.
- The ability to act on its environment.
- The ability to plan actions to achieve its goals.

Surprisingly, very few systems in the, admittedly short, history of robotics have possessed all of these capabilities in non-trivial form. Most of the ones that have had this combination of capabilities have been mobile robots, for example, Shakey [Nilsson 69], and Hilare [Giralt et al 79]. In the area of manipulation, some early systems possessed these capabilities, for example, the Stanford Hand-Eye System [Feldman 69; Paul 72], MIT Copy Demo [Winston 72], Edinburgh's Freddy [Ambler et al 1975], and a very few recent systems, for example, [Ikeuchi, et al 86]. In all cases, these systems operated in a very restricted domain of objects and their component modules were tailored to specific tasks.

In this paper, we describe a new integrated robot system, called Handey, under development at MIT. Handey's domain is that of simple assembly of (mostly) planar-faced objects. The user starts by building accurate object models for all the parts to be manipulated. The user then specifies a sequence of MOVE commands, each of which specifies an object and its destination. Handey locates each part on its worktable, chooses how to grasp it, and takes it to the destination. The unique features of Handey are its ability to operate on a wide class of objects and to operate in a cluttered environment. Also, Handey's modules are designed to be reasonably general purpose; they are not tailored to a specific task. Figure 1 illustrates a sample plan found by Handey; the task consists of picking up one of the L-shaped objects and placing it on top of the other so that they form a block.

Handey consists of the following major modules:

- *Object modeling*: Handey can model a fairly general class of polyhedral objects, including an interactive facility for defining object models from depth data.
- *Range finder*: Handey uses a triangulation range-finder based on projected laser planes.
- *Model-based object localization*: Handey can locate known polyhedral objects in complex scenes that may involve obscuration of the target object.
- *Collision-free path planning*: Handey can plan motions for a six degree of freedom revolute manipulator in cluttered environments.
- *Grasping*: Handey can choose grasping positions on objects in cluttered environments and plan regrasping motions if necessary.
- *Robot Control*: Handey uses a traditional trajectory controller capable of joint-interpolated or cartesian motions.

Handey's current limitations are: the lack of any capability for planning or executing compliant motions, the inability to postpone decisions, and its limited ability to deal with errors. Eliminating these limitations is the the subject of ongoing work.

Before proceeding to a description of Handey, we must answer an important question: "What is to be learnt by building an integrated robot system such as Handey?" There are several good reasons. The first is to uncover problems in the interaction between the modules. The second is to stress-test the modules; the use of module within a system tends to uncover hidden assumptions in the design. The third is that the resulting system can be a tool in investigating high-level planning in Artificial Intelligence.

2. An overview of Handey

The basic command for Handey is of the form: MOVE *object* TO *destination*. In this section we describe briefly the steps Handey must go through to execute a single MOVE command. Subsequent sections will focus on the individual steps in more detail.

Handey requires that a metrically-accurate polyhedral model of the object be available. The object is assumed to be located in arbitrary pose within the field of view of the range sensor. The destination will typically be outside of the field of view.

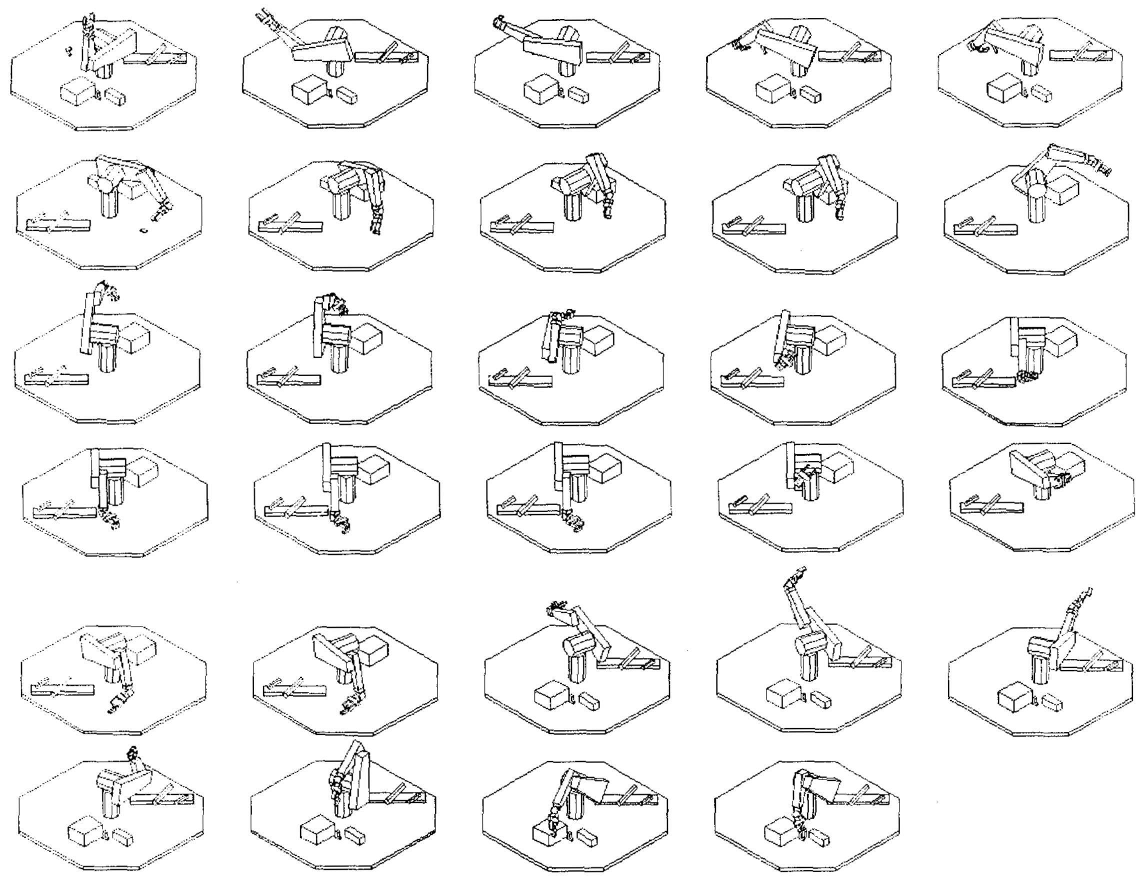

Figure 1. A plan found by Handey. The first image is on the top left; it shows the initial position of the part identified by the edge matcher. The final position of the part, specified by the user, is on the bottom right.

Handey executes the following steps in fixed order:

- *Locate the object*: A depth map is built of the area under the range sensor's view. The recognition module locates the object within the scene and returns a transformation that describes the object's position relative to robot's coordinate system.
- *Choose a grasp on the object*: The grasping module searches for a grasp on the object that is both stable and can be reached by the robot at both the origin and destination. Choosing a grasp will require taking into account the obstacles present in the depth map and in the world model. If such a grasp is not possible, then choose a legal grasp for the object's original position; regrasping will be necessary.
- *Approach the grasp location and grasp the part*: Plan a collision-free path from the robot's current position to the chosen grasp position. Grasp the part.
- *Regrasp the object*: If regrasping is necessary, plan a sequence of regrasping motions that will enable the robot to reach a grasp that is legal at the destination. An empty area of the worktable is used to perform these motions.
- *Plan an approach path to the destination*: Plan a collision-free path from the robot's current position to a point near the destination.
- *Place the object at the destination*: Generate a force-guarded motion to place the object at the destination. In a future version of the system, a compliant motion strategy should be used.

Subsequent sections explain the operation of the different modules and how they are used to carry out these steps.

3. Locating the object

The first step in executing a MOVE command is localizing the object using depth information obtained from a range sensor. The range sensor is a light-striping triangulation sensor. It is used to produce a depth map of a small region on the worktable. The object to be MOVE'd is assumed to be present in this area.

After the depth map is constructed (Figure 2), the map is processed as if it were an image, except that "brightness" corresponds directly to elevation above the worktable. A standard "edge" operator [Canny 86] is run over the image and extended linear segments are identified in the resulting array. Note that this process identifies 3D edge segments, not just their projection in an image.

The method used for object localization is a simple hypothesize-verify algorithm based on matching linear segments in the depth map to edges in the polyhedral model of the part. This method is a variation of the method described in [Lozano-Pérez and Grimson 86], using edge data instead of face data. The basic step in the matcher is to take two edges in the model and consider all possible assignments of these edges to pairs of data

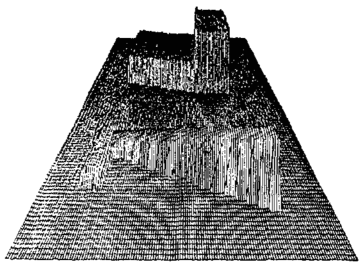
Figure 2. A very simple depth map

edges. If the pairs of model edges are non-collinear, such an assignment of data edges to model edges is sufficient to solve for at most four transformations that map the model coordinate system into the data coordinate system. The ambiguity in the transformation arises due to the possible assignment of direction vectors to the edges. Given such a transformation, we can predict the location of other model edges in the data and verify their presence. The assignment that predicts the location of the most data edges is chosen.

The matcher attempts to consider only a few assignments of data edge pairs to model edge pairs. To reduce the set of such assignments it exploits two basic geometric constraints. First, a data edge should match a model edge only if their lengths are compatible, that is, if the length of the data edge is less than or equal to that of the model edge. Second, a pair of data edges can be matched to a pair of model edges if the parameters describing the relative pose of the edges in the pairs are consistent, taking into account the measurement error. Exploiting these constraints significantly reduces the number of matches that need to be considered.

We can describe the relative pose of two 3D edges using the following parameters (Figure 3): α — the angle between the lines supporting the edge segments; d — the length between the lines measured along the common perpendicular to the lines; a_1 and a_2 — the distance from the base of the common perpendicular to the nearest end of the edge. Note that we only know the *line* on which a data edge lies; to obtain a *vector* we would need to know an additional sign. Therefore, there is a potential ambiguity in the parameters arising from the missing sign. This has to be treated carefully when testing whether two pairs are consistent.

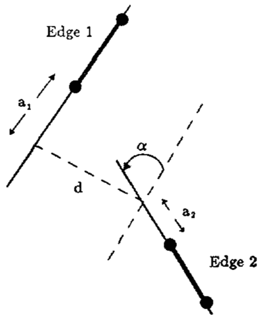

Figure 3. Definition of the four parameters that define the relationship between two edge segments.

Figure 4 shows examples of the matcher in operation, using edges extracted from depth maps similar to that in Figure 2.

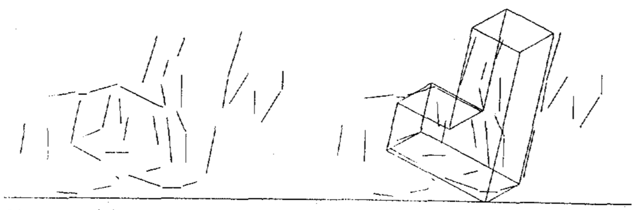
Figure 4. (a) The x, y projection of edge fragments obtained by running the Canny edge detector on a depth map. (b) The projected object model located by the edge matcher superimposed on the edge fragments. This example corresponds to Figure 1.

4. Planning collision-free motions

At a number of points in the operation of the system, a collision-free path is required from one specified location to another. Handey uses a simplified version of the path planner described in [Lozano-Pérez 86]. This path planner uses the robot's joint space as the configuration space.

The obstacles are mapped into a quantized version of this configuration space by a simple numerical method illustrated in Figure 5 for a two-link manipulator. Given a value for θ_1, we can compute the range of forbidden values of θ_2 due to each of the obstacles. The set of forbidden ranges of θ_2 for each value of θ_1 comprise an approximation to the exact configuration space. This space can then be searched for a path. For a manipulator with three joints, the process described above (applied to joints two and three) is repeated for all possible (quantized) values of the first joint angle.

The version of the path planner used by Handey never computes configuration spaces of dimension greater than three, but it allows motions requiring six degrees of freedom. Essentially, we assume that a path from the start to the goal exists such

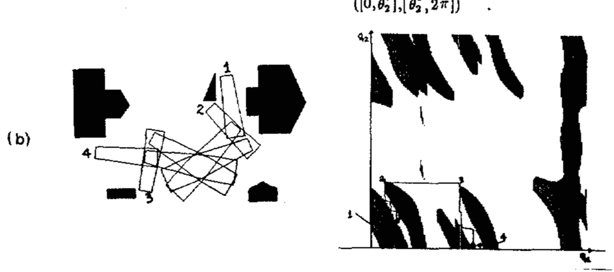

Figure 5. (a) The configuration space obstacles for a two-link manipulator are built by quantizing θ_1 and finding forbidden ranges of θ_2. (b) A sample configuration space with a path obtained by the path-planning algorithm.

that the last three joints of the arm retain their starting values until some intermediate point where they are changed to their values at the goal and never changed after that. It is easy to construct cases where this assumption will fail, but it works in a large percentage of actual cases.

The actual planning proceeds as follows: An approximate arm model is built in which the last three links are replaced by a box. This box must be large enough to enclose the last three links, the hand, and any object in the hand, not only at their start and goal positions but also at intermediate positions between the two. The three-dimensional configuration space for this model can then be built. We then find the closest free points in this configuration space to both the start and goal positions. A path is found between these two free points. Note that the complete robot is guaranteed to be safe along this path, for the whole range of values of the last three joints between the start and the goal. Therefore, we can simply interpolate the values of the last three joints between the start and goal values. Then, we plan a path using the original model of the robot between the free point closest to the start and the start itself. We also plan a path from the free point closest to the goal to the goal itself. In these two paths, the values of the last three joints are fixed. The concatenation of these three paths form the desired path.

5. Grasping

The most intensive interaction between perception, planning and action in Handey happens in grasping. After the target object's location has been determined, Handey must choose a pair of features, such as a pair of parallel faces, for grasping. Then, it must choose a path to reach those features that avoids any nearby objects. In addition to any known objects in the model, the hand must avoid colliding with any obstacles detected in the depth map, even if their identity is unknown.

A grasp that is suitable for picking up the object may not be suitable for placing it at the destination. Handey attempts to find consistent with both states. If there is not a single grasp suitable for both, then a sequence of grasps and intermediate motions may be necessary.

We model the hand by two opposing rectangular fingers which can close on an object by sliding along a crosspiece. A legal grasp has each finger in contact with a grasp feature on the object. The feasible grasp feature pairs are: two parallel faces of the object, or a face and a parallel edge, or a face and a vertex. The current version of Handey limits its attention to pairs of parallel faces.

The criteria for a legal grasp are as follows:

- The grasp must be stable. That is, the location and magnitude of the forces and torques exerted by the fingers must produce a balanced system of forces and be sufficient to overcome the effect of gravitational forces on the grasped object.
- The grasp must be reachable, both at the pickup point and the putdown point. That is, there must be a clear path to achieve contact with the chosen grasp features at the original location of the grasped object. Also, the grasp must not produce a collision when the object is placed at its destination.

In the current implementation of Handey we use very simple heuristics to guarantee stability. Our rationale is that for small objects and strong fingers, almost all grasps are stable. Since we limit ourselves to pairs of parallel faces, the forces are automatically balanced. Furthermore, we require that there be a user-specified minimum area of contact between the fingers and one of the grasped faces.

In general, to guarantee the stability of a grasp one must compute the forces and torques generated by the fingers, including the frictional forces and torques, and compare them to the gravity forces and torques acting on the object. Only if the applied forces and torques can balance the gravitational forces and torques will the grasp be stable. A procedure for computing the forces and torques for a particular grasp is given in [Barber et al 86.]. Future versions of Handey will incorporate a more careful stability test.

We have placed more emphasis on guaranteeing reachability. There are three phases in computing a reachable grasp. The first phase is choosing the grasp features and a gross orientation for the hand. The second is planning the detailed grasping motion. The third is regrasping, when necessary. The following sections examine these phases in turn.

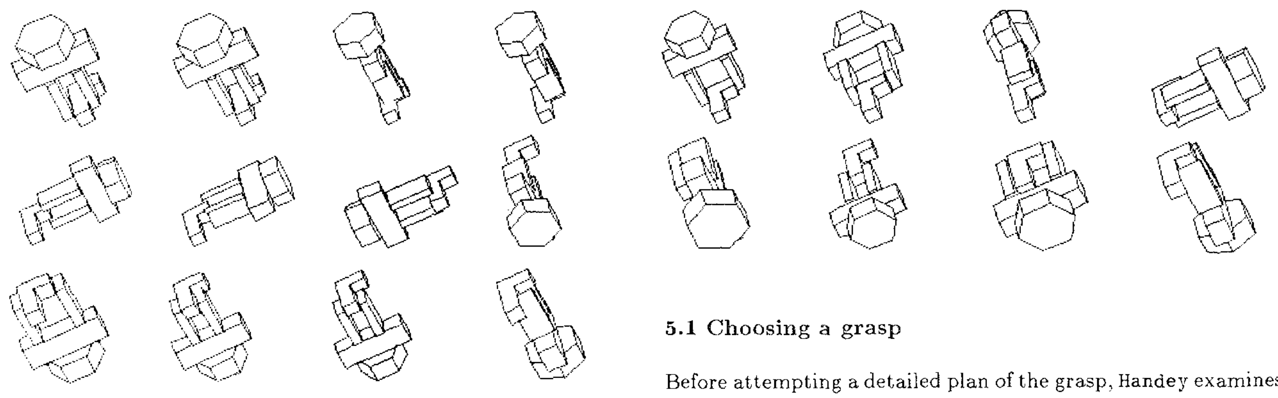
Figure 6. The different groups of approach directions and grasp classes for a particular orientation of an L-shaped object, heuristically ranked by desirability.

5.1 Choosing a grasp

Before attempting a detailed plan of the grasp, Handey examines different classes of candidate grasps and evaluates their feasibility both at the pickup point and the putdown point. A grasp class is characterized by a choice of object surfaces. Within a grasp class, there are qualitatively different ways of approaching the grasp. Handey splits the approach directions into groups determined by which edge of the grasp face is crossed first when approaching the face. Figure 6 shows the different groups of approach directions and grasp classes for a particular orientation of an L-shaped object. The grasps are sorted by a measure of how vertical the fingers are. Note that the grasps shown here are representative elements of the range of approach directions that Handey considers; they are not an exhaustive list of approach directions.

The feasibility of one of these grasp groups is investigated as follows:

- An inverse kinematic solution for the center of the grasp group is performed. If all the feasible solutions are too close to a joint limit, then the grasp group is discarded.
- Assuming that the fingers have penetrated into the face by the minimum amount to guarantee a stable grasp, check the range of feasible rotations of the hand about that point. The two endpoints of such an angular range are shown in Figure 7. When computing this range one must consider potential contacts between the hand and the object to be grasped and the table, both at the pickup and putdown position. These two sets of constraints can be treated simultaneously by transforming the obstacles at the putdown position into the pickup position using the inverse of the transformation relating the putdown to the pickup pose (see Figure 8). The range of legal angles can be computed using a submodule of the path planner [Lozano-Pérez 86].

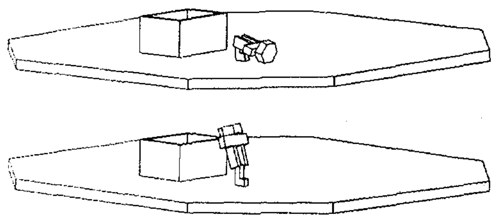
Figure 7. The extrema of the legal range of hand orientations at a particular grasp point.

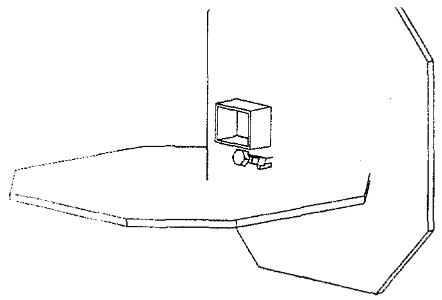
Figure 8. The obstacles at the putdown pose are transformed into pickup pose. A legal grasp in this environment is legal for both poses.

The highest ranked grasp group that passes these tests is used to compute a grasping motion that considers obstacles in the depth map. If a grasping motion cannot be found, a different grasp group is tried.

5.2 Planning the grasping motion

When choosing a grasping motion, we must take into account the presence of nearby objects as reflected in the depth map. For this purpose we use a planner specialized for planning the motion of the hand in the grasp plane. The *grasp plane* is a plane parallel to the faces being grasped and midway between them. When approaching a grasp the fingers remain parallel to the grasp plane and centered about it but are otherwise free to rotate and translate in the plane. The restriction to motion in the grasp plane is intended to minimize the risk of collision with the object to be grasped.

The planner uses a method loosely modeled on the potential field method for obstacle avoidance [Khatib 85]. The straightforward potential field method, although applicable, is not convenient in this situation because the obstacles are a large number of points, rather than a few extended obstacles. Also, the method described here is less likely to get stuck in local minima of the potential field.

The grasp plane is bounded by a rectangle whose size determines the range of motion allowed the hand during grasping. The rectangular volume the hand can sweep out while constrained to move in the grasp plane is the grasp volume. In fact we must consider separately three grasp volumes – one for each finger and one for the crosspiece. Only these volumes need to be investigated for potential collisions.

Consider a point specified by the depth map which lies within or above a grasp volume. The point can be thought of as the origin of a ray which extends downward through the table. That portion of the ray which lies within the grasp-volume is projected onto the grasp plane. Following this procedure for all such points marks the portion of the grasp plane where objects intrude into a grasp volume, that is, places where the hand cannot go. For ease of computation the projected line segments are discretized – becoming filled cells in a grid imposed on the grasp 'ane.

We establish grasp points on the face of the object to be grasped and on the finger which will contact it. The goal of the grasp motion planner is to bring these two points as close together as possible without causing a collision between the hand and other objects. In the absence of intervening filled grid cells in the grasp plane the motion of the hand would be a simple translation along the vector connecting the finger and object grasp points. We call the unit vector in this direction the free motion vector.

Surrounding the hand at some distance and moving with it are bump lines. A bump line is a line segment on the grasp plane which is checked each iteration to see if it crosses a filled grid cell. A bump vector is a unit vector perpendicular to a bump line pointing away from the hand (see Figure 9). Also associated with each bump line is a multiplier used for limiting motion along its bump vector. This multiplier is related to the distance between the hand and the bump line.

After investigating all the bump lines for collisions with filled cells we construct a unit circle and map onto it the bump vectors and free motion vector. In Figure 9, we show the product of the bump vectors and their multipliers. For non-colliding

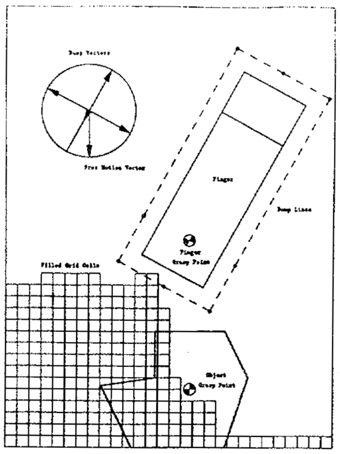

Figure 9. The grasp approach planner algorithm.

bump lines the multiplier is 1, for colliding bump lines it is 0. Not shown are bump lines which surround the hand at a greater distance; they may have collision multipliers greater than 1.

There are several possible ways to combine these vectors to pick a direction to move the hand. One way is simply to move along the non-zero bump vector closest in direction to the free motion vector. (This however leads to stairstep motion of the hand.) Another way is to move along the free motion vector as far as is allowed by the scaled bump vectors, choosing the former procedure only if latter computed no motion.

The bump lines also provide a convenient way of computing a "torque" to rotate the hand. Any colliding bump line produces a torque whose magnitude is proportional to the cross product of the bump vector and a vector connecting the finger grasp point and the center of the bump line. The total torque on the hand is just the sum of torques generated by each colliding bump line.

In each iteration the distance between the finger and the object grasp points is checked. If this distance drops below a preselected threshold the grasp motion planner stops and returns the path it has discovered. However, as this method may fail to find a suitable grasp, the planner must be terminated after a certain number of iterations in any case. Even after such a termination a grasp is legal only if the finger and the grasp object face overlap sufficiently.

An example of planner output is shown in Figure 10.

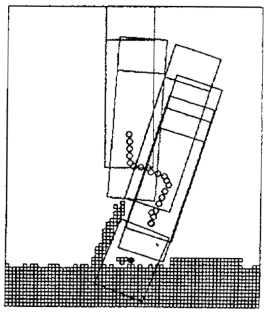

Figure 10. The grasp approach planner in operation.

5.3 Regrasping

In regrasping, one wants to find a sequence of pickup and putdown motions that will reorient the object so that it can be placed at the destination without collision. The regrasping algorithm used in Handey is described in [Tournassoud, Lozano-Pérez, and Mazer 87].

6. Calibration

The operation of Handey requires having accurate calibrations between several coordinate frames. The key systems that maintain their own coordinate frame are the range sensor and the modeling system. We are interested in the mapping between these coordinate frames and the cartesian frame supported by the robot controller. This section describes the calibration of the range sensor frame relative to the robot frame.

The calibration procedure assumes that the z axis of the range sensor's frame is parallel to the z axis of the robot's frame and that the height of the table is known in both reference systems. These assumptions are enforced by the mechanical construction of the range sensor. So the calibration problem is a two dimensional problem consisting of computing a rotation about the z axis and a linear offset in the x-y plane.

Phase one: Computing the rotation. A cube is placed in the robot's gripper so that the cube is aligned with the gripper, but its position relative to the gripper is arbitrary. The robot is commanded to move the cube in the field of view of the range sensor. Let (xw_1, yw_1) be the coordinates of the robot wrist in the x-y plane of the robot frame and (xv_1, yv_1) be the coordinates of the centroid of the single face of the cube visible to the range sensor (the z value is not used).

Then, the robots moves to $(xr_1 + dx, yr_1)$ without changing orientation. dx is chosen so that the cube remains in the field of view of the range sensor, (xv_2, yv_2) are the new coordinates of the face centroid given by the range sensor. The vector $(xv_2 - xv_1, yv_2 - yv_1)$ is parallel to the x axis of the robot frame, this allows us to compute the rotation between the sensor frame and the robot frame as: $R = Rot(\hat{\mathbf{z}}, atan2(xv_2 - xv_1, yv_2 - yv_1))$, where $Rot(\mathbf{v}, \theta)$ is the homogeneous transformation representing a rotation by θ radians about the vector $\mathbf{v}$.

Phase two: Computing the offset. Since nothing is assumed concerning the relative location of the wrist and of the centroid of the cube face, let (fx, fy) be the projection of this vector in the x-y plane of the robot frame. The goal of the next motion is to compute fx and fy. The robot is commanded to rotate the hand about the robot frame's z axis by an angle of π, while keeping the wrist position fixed. Therefore, the cube face's centroid moves to the opposite end of a circle centered on the projection of the wrist. Call the new centroid location (xv_3, yv_3). We can now write: $(fx, fy) = 0.5R^{-1}(xv_2 - xv_3, yv_2 - yv_3)$. The offset between the origin of the robot frame and the origin of the sensor frame is given by: $(o_x, o_y) = (xr_2, yr_2) + (fx, fy) + R^{-1}(xv_2, yv_2))$. If h_r is the height of the table in the robot frame and h_v and the height in the sensor frame then $o_z = h_r - h_v$ is the z offset between the two frames.

Finally the transform that maps points in the sensor frame to points in the robot frame can be written as: $Trans(o_x, o_y, o_z)R^{-1}$, where $Trans(x, y, z)$ is the homogeneous transformation corresponding to a translation of the origin to (x, y, z).

7. Discussion

Building Handey has been quite difficult. This reflects the usual difficulty of building large systems. We feel that it was feasible at all due to the use of very simple and robust algorithms for the constituent modules, such as the path planner and recognition module.

The most significant lesson we have drawn from our experience so far with Handey is the need for a systematic and efficient way of dealing with the number of options available while constructing a plan. It is instructive to consider the number of geometrically different ways one could go about stacking two blocks. Consider the block symmetries, the hand symmetries, multiple kinematic solutions, multiple grasp points, and multiple paths. In most cases we don't care which solution is chosen but, unfortunately, many of the possible solutions can be impossible due to the presence of nearby objects or limitations in the robot's joint angles, etc. Handey simply lists all possible solutions, ranks them heuristically, and tries them sequentially until one works. While adequate in the short term, this strategy leaves much to be desired. In earlier work [Lozano-Pérez and Brooks 85], we have considered the use of constraints as a mechanism for making these decisions. Constraint propagation and satisfaction, however, can be extremely difficult and computationally expensive. This area requires a great deal of further work.

Future work on Handey will also attempt to expand its capability to do sensor-guided assembly and to do meaningful error detection and recovery.

Acknowledgments

This work was funded primarily by the Office of Naval Research under contract N00014-85-K-0214. Additional support was provided by an NSF Presidential Young Investigator Award (Lozano-Pérez) and the French CNRS (Mazer). The visits of Lanusse and Tournassoud to MIT were funded by their home organizations in France: E. T. C. A. and INRIA respectively.

Bibliography

A. P. Ambler, H. G. Barrow, C. M. Brown, R. M. Burstall, and R. J. Popplestone, "A versatile system for computer-controlled assembly." *Artificial Intelligence*, Vol 6, 1975, pp 129–156.

J. Barber, R. A. Volz, R. Desai, R. Rubinfeld, B. Schipper, and J. Wolter, "Automatic two-fingered grip selection," in *Proc. IEEE Conf. Robotics and Automation*, San Francisco, 1986, pp. 890–896.

J. Canny, "A Computational Approach to Edge Detection," *IEEE Trans. on PAMI*, Vol 8, No 6, 1986, pp 679-698.

G. Giralt, R. P. Sobek, R. Chatila, "A multilevel planning and navigation system for a mobile robot: A first approach to Hilare," in *Proc. 6th IJCAI*, Tokyo, 1979.

K. Ikeuchi, H. K. Nishihara, B. K. P. Horn, P. G. Sobalvarro, S. Nagata, "Determining grasp configurations using photometric stereo and the PRISM binocular stereo system." *Robotics Research*, Vol 5, No 1, 1986, pp 46–65.

O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proc. IEEE Conf. Robotics and Automation*, St. Louis, 1985, pp. 500–506.

T. Lozano-Pérez, "A Simple Motion Planning Algorithm for General Robot Manipulators," in *Proc. 5th AAAI,* Philadelphia, 1986, pp. 626–631.

T. Lozano-Pérez and R. A. Brooks, "An Approach to Automatic Robot Programming," MIT AI Lab., AIM 842, 1985.

T. Lozano-Pérez and W. E. L. Grimson, "Off-line planning for on-line object localization," in *Proc. of FJCC.*, Dallas, 1986.

N. Nilsson, "A mobile automaton: an application of artificial intelligence," in *Proc. 1st IJCAI*, 1969, pp. 509–520.

R. P. Paul, "Modeling, trajectory calculation, and servoing of a computer controlled arm," Stanford AI Lab., AIM 177, 1972.

P. Tournassoud, T. Lozano-Pérez, and E. Mazer, "Regrasping," in *Proc. IEEE Conf. Robotics and Automation*, 1987.

P. H. Winston, "The MIT Robot," in *Machine Intelligence 7*, Edinburgh University Press, 1972, pp 431–463.