Searching for Grasping Opportunities on Unmodeled 3D Objects

Martin Rutishauser and Markus Stricker
Image Science Laboratory
Swiss Federal Institute of Technology
CH-8092 Zürich, Switzerland
{rutis,stricker}@vision.ee.ethz.ch

Abstract

We are investigating the problem of removing 3D objects from a heap without having recourse to object models. As we are relying on geometric information alone, the use of range data is a natural choice. To ensure that we see opposite patches of the object surfaces, we use up to three range views from different directions. These views are triangulated using the data points as vertices. After merging the views, the resulting surface description is segmented into patches which correspond to object parts. We present a novel approach to search for grasping opportunities on a selected part. We allow all combinations of two vertices to be possible contact points for the two finger gripper. Based on evidence accumulation of local features, a quality measure is defined for each of these vertex pairs. A discrete optimization algorithm which is based on Tabu-Search then tries to find several good grasping configurations. Global constraints which are not part of the objective function (e.g. collisions) have to be respected.

Keywords: Grasping, range data, unmodeled objects, Tabu-search.

1 Introduction

Perceptual and reasoning capabilities are crucial if robots are to work in unstructured or weakly structured environments. A large class of tasks in such a context is the manipulation of objects. Up to now, the paradigm in this field has been the object model based approach for object recognition and pose determination. However, there are many material-handling tasks where it is desirable that the robot vision system also has the ability to deal with objects of which it has no stored models.

Examples include the clearing of objects from floors, working spaces, conveyor belts, cafeteria trays and alike. Isolation of unknown objects from a heap or from a dense object stream (as can be found in a part feeding system in manufacture) is another application. A system that is able to work without object models can also be used for exception handling. There can be object streams (e.g., on conveyor belts) which are mixed. The robot vision system knows object models for most of the objects which have to be grasped, but every once in a while an unknown object is arriving. Whenever the vision system realizes that it cannot recognize the object it invokes an auxiliary system (not relying on object models) which then analyzes the scene and derives possible gripping points.

BMVC 1995 doi:10.5244/C.9.28

From the given examples it is evident that techniques for object manipulation without the use of object models can be useful in many application areas. Up to now however, little work has been done by the robot vision community for the case of scenes consisting of heaps of unmodeled objects.

In this work we report a method to infer from sensor data alone sufficient information for a robot to grasp and remove unmodeled 3D objects. The goal is to efficiently find grasping opportunities on objects that are unknown to a manipulation system, i.e., objects that are unmodeled and arbitrarily shaped. As we are relying on geometric information alone, the use of range data is a natural choice. They offer a direct way to extract all surface attributes needed. We use up to three range views from different directions. In a first processing step, these views are triangulated using the data points as vertices. After merging the views, the resulting surface description is segmented into patches which correspond to object parts in the scene.

We present a novel approach to search for grasping opportunities on a selected part. We allow all combinations of two vertices to be possible contact points for the two finger gripper. Based on evidence accumulation, several local features are combined to define a quality measure for the vertex pair under consideration. Examples of prominent features are measures for the oppositeness of the vertices, the expected force and torque the fingers have to exert and the distance between the two vertices.

A discrete optimization algorithm then tries to find a good grasping configuration. Usually, there are several ways of grasping an object. So, in contrast to standard optimization problems, we are not trying to find the optimal grasping configuration, but merely a good and stable one, i.e., a suitable local optimum of the quality measure. We use a Tabu-search strategy, where knowledge about the optimization problem can nicely be integrated into the search engine. Since the Tabu-search memorizes attributes of visited states, the search can be interrupted and global constraints which are not part of the objective function (e.g. collision detection) can be checked. The search is stopped if a prominent local optimum of the quality measure satisfies all global constraints, i.e., an acceptable grasping position is found.

2 Related Work

Work on deriving grasps for single unmodeled objects includes [1] and [9]. Boissonnat describes a method to find stable grasps for a robot gripper without the use of object models, merely from an analysis of the objects silhouette which is approximated by a polygonal sequence. The grasps are ranked by quality which is determined by a criterion having four components. Stansfield proposes to grasp single unmodeled objects with a knowledge-based approach which draws on theories about human grasping behavior. From range data a representation of the sensed object in the form of a set of up to five aspects is generated. This symbolic representation is used by a rule-based system to derive a set of possible grasps.

Several authors have employed generic object models where the type of the admitted objects is known, but where the dimensions of the instances occurring in the scene have to be determined. A vision system of a planetary explorer which is supposed to automatically collect rock samples is presented in [4]. Pebbles which are not touching each other are partially buried in sand, and range images of them are taken. The visible surface parts serve to estimate shape and pose parameters of superquadrics. In [11], a system is described

which is able to remove objects from a heap, one by one. The heap is lying on a base plane and single range views and/or intensity images are taken from an essentially vertical direction. Thus, it is not possible to see vertical or overhanging surfaces. However, they assume that only convex objects are admitted, more specifically objects from the postal domain, i.e., flats, parcels and tubes. This generic model knowledge helps them to interpret the views and to identify grasps. The authors of [6] have also been working with boxes and cylinders in a project for the US Postal Service. They tried to physically understand object configurations using range images.

Our work differs from the above in that it deals with heaps of unknown objects which do not need to be concave or conform to a generic model. Our emphasis is on identifying grasping opportunities in the heap rather than recognizing objects. This could be termed *action-based recognition* and it is interesting to compare it to function-based object recognition, proposed by [10]. The latter is a generic form of object recognition which is based on the detection of features in the object instance which (after evidence accumulation) allow to identify the function for which people use the object and therefore the object class. In our approach we do not recognize object classes but action classes that a robot might perform on a certain part of a heap. The means to arrive at the recognition of grasping opportunities is accumulation of evidence as in [10].

3 Input Data and Preprocessing

If we disregard object models, then geometrical properties must be extracted directly from the data. This is because we have virtually no a priori knowledge about the world the robot is working in. The determination of the grasping points has to rely exclusively on shape information. We use range data because they provide an explicit description of the object's surface and therefore a direct way to extract all relevant properties. As we are using a two-fingered parallel gripper, it is mandatory for our system to see opposite surface patches.

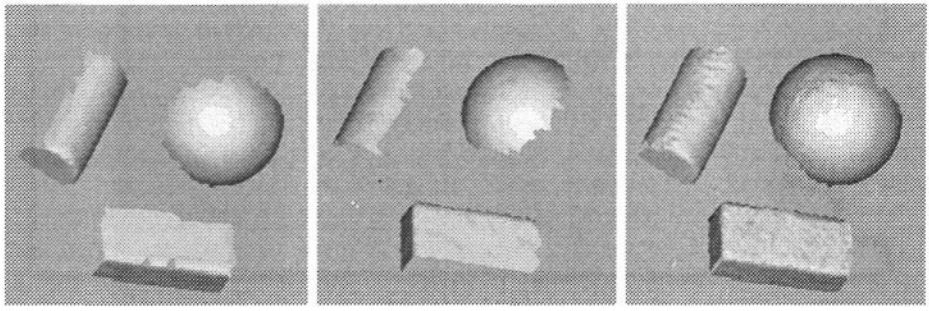

Figure 1: *Merging and segmentation of range images.* From left to right: Surface seen by first sensor, surface seen by second sensor, result after merging.

To ensure this, we use three range sensors which are positioned equally spaced around the scene. The integration of the three data sets into a global surface description is done as described in [8]. Each of the acquired data sets is tessellated into triangles. Subsequently, the surface points in overlapping regions are pushed towards each other to reduce the measurement error. The amount of displacement and its direction are calculated using a

modified Kalman minimum-variance estimator. A retriangulation brings the data set into its final form. Figure 1 shows an example scene. At this stage our representation of the scene consists of several surface regions. Each region is described by a set of connected triangles and their corresponding vertices.

We then partition the surface description into surface patch assemblies which hopefully have a one to one correspondence with objects or object parts. The segmentation is based on detecting surface and surface-orientation discontinuities. Based on the spatial arrangement of the patches, an object is selected and its surface description is passed on to the next processing step: the searching for grasping opportunities. In [7] we give a detailed description of the segmentation and selection scheme and discuss the problem of over segmentation, i.e., the creation of "phantom" objects.

4 Searching for Grasping Opportunities: An Optimization Problem

We intentionally refrain from doing any further surface segmentation (e.g. into planar or quadric patches) or from performing a global surface parameterization. We think that the main advantage of a segmentation into surface primitives would be a data reduction. In our case, the surface primitives do not really help the calculation of the grasping quality of a contact pair because most of the features we use have a very local nature. In addition, stepping from one surface primitive to an adjacent one may result in an abrupt change of the quality measure. Finally, a global surface parameterization is not done as it clearly contradicts the requirement of handling "arbitrarily" shaped objects.

During grasping, we want both fingers of the robot hand to touch the object and thus support the grasp[1]. For a two finger gripper this means that we have to find two suitable contact points on the surface. As already mentioned, we allow all vertices of a specific object to be candidate contact points, i.e, all combinations of two vertices form a possible grasping configuration. Based on evidence accumulation, we define a quality measure for each of the vertex pairs. Hence, the search for good grasping opportunities becomes a constrained, discrete optimization problem: find a vertex pair with a good grasping quality and which is reachable by the robot. It is important to note that we are not primarily looking for the optimal grasp but for several good grasping opportunities. The obvious reason is that there are usually several ways of firmly grasping an object. So, in contrast to standard optimization problems, we are trying to find a reasonably good and stable grasping configuration which satisfies the global constraints.

Since all vertex pairs of an object are admitted in the optimization, one challenge is to construct an optimization method that can cope with the large amount of data with a reasonable computational complexity. Another challenge is to find the local maxima which are separated by large areas in which the quality measure is low.

We use a Tabu-search strategy, where knowledge about the optimization problem can nicely be integrated into the search engine. In our application, the search can be thought of as letting the two fingers of the robot slide over the object's surface. Starting from one point with closed fingers, the search continuously tries to improve the grasping quality. As soon as a "good" grasping configuration is found, the global constraints are checked. If the grasp cannot be carried out (collision or no inverse kinematic solution), the search

[1] Although not very likely, one finger grasps are also possible, e.g., for objects with holes

is continued. As soon as a valid grasp is found, the search is stopped and the grasping parameters are sent to the robot. If no adequate grasping points can be found within a certain number of iterations, the object is marked as non graspable.

5 Grasp Quality / Objective Function

Defining the quality of a grasping configuration involves the computation of local as well as global features. Examples of local features are the surface normals, the surface curvatures or the expected friction forces and torque the fingers have to exert[2]. Examples of global features are collision detection or the question whether the robot actually can reach the grasping position. They depend on the configuration of the whole scene. Due to their dependency on global arrangements, their computational complexity is also higher. And they are binary features in contrast to the local features, which can take float values.

Based on these observations we decided to treat the two classes of features differently in the optimization process: All local features are part of the quality function and the global features are used as additional constraints. In this way, we relieve the objective function from complex computations and thus, speed up the search for good grasping opportunities. The constraints are only used to decide on the "usability" of a configuration found to be good.

5.1 Local Features

For the case of a two-finger-gripper with parallel jaw, we have identified the following attributes for a pair of contact points and their small neighborhoods of triangles[3]:

- the surface normals at the two contact points should be anti-parallel and they should be in a relation of oppositeness.
- the contact points should not be too far apart (due to a limited gripper opening).
- the friction force at the contact points should be minimal.
- the torque the fingers have to exert should be minimal.
- the surface at the contact points should not be too concave.
- the height of the patch pair relative to the base should be maximized (reduces the chance of collisions).

For a vertex pair (p_1, p_2) and a feature i from the above list, we define a *primary feature value* $f_i(p_1, p_2)$. This value measures the deviation of the feature i from its optimum. Obviously, they depend on a variety of physical and geometrical properties. In order to combine the primary feature values in a meaningful way, they have to be normalized. Only then it is possible to set the relative importance of the features which is a highly desirable property for any conjunctive combination of evidence.

We use the following Gauss function to transform $f_i(p_1, p_2)$ into the *normalized feature values* $\overline{f}_i(p_1, p_2)$:

$$\overline{f}_i(p_1, p_2) = \exp \frac{-f_i^2(p_1, p_2)}{\sigma_i^2}$$

[2]Force and torque depend on the center of gravity which by itself is a global object property. But once estimated, the larger neighborhood of a vertex does not influence the force/torque calculation anymore.

[3]We assume point contacts of the fingers.

Thus, we limit the range of a single feature to the unit interval. A value of 0.0 means that the feature is absent and a value of 1.0 that the feature is "highly" present. The parameter σ_i controls the spread of a feature i and is independent of the vertex pair under consideration. The Gauss function has the nice property that all its derivatives vanish in its maximum. Consequently, minor deviations from the optimal state of a feature cause only minor changes of the the normalized feature values. For example, small displacements of contact points result in only a small change of the normalized "oppositeness" feature.

By selecting different σ_i for different features, the relative importance of a feature can be controlled. It is important to note that we do not change the range of the normalized feature value, but the range of primary feature values that are accepted as good. For a large σ_i, larger deviations from the optimum are ranked acceptable than in the case of a small σ_i.

5.2 Combining Local Features

To calculate the quality function, we have to combine all values of the local features in a consistent way. This means that if one feature is 0.0, the resulting quality should be 0.0, too. Fuzzy set theory provides several functions for the conjunctive combination of evidence, the so-called triangular norms, or T-norms which have the appropriate qualitative effect, among them $T_1(a,b) = \min(a,b)$ or $T_2(a,b) = a \times b$. Using the latter, our final grasping quality measure for one vertex pair becomes:

$$q(p_1,p_2) = \prod_i \overline{f}_i(p_1,p_2) = \exp \sum_i \frac{-f_i^2(p_1,p_2)}{\sigma_i^2}$$

where $q(p_1,p_2)$ stands for "quality of vertex pair (p_1,p_2)", The goal of the optimization procedure is therefore to maximize the quality function $q(p_1,p_2)$ over all possible vertex pairs.

Since we admit all vertex pairs, it is evident that most of them have a very low quality. The shape of the function can best be described by comparing it with a large sea with only a few prominent islands denoting acceptable vertex combinations.

6 Solving the Optimization Problem

6.1 Tabu Search Strategy

Tabu search [2] is a general heuristic procedure for discrete optimization. It is computationally more efficient than simulated annealing [5] and unlike neural networks [3], constraints can be enforced throughout the search. Furthermore, the fact that it is a search rather than a method which converges to a single solution bears significant advantages in our application. If the optimization is subject to constraints of which only some are modeled by the objective function, then even a global optimum might have to be discarded once all the constraints are evaluated. In this case, tabu search will leave this optimum in the "most promising direction".

This behavior is achieved by augmenting a steepest ascent method with two mechanisms: One to prevent the search from returning to a recently visited optimum by labeling some points in the neighborhood as tabu, hence the name tabu search, and another to drive the search into areas of the search space that have not yet or the least been explored. As an extension of a local neighborhood search, in tabu search we first evaluate the objective

function for all the points in a neighborhood around the current point in the search space. Based on the history of the most recent moves we execute the tabu function which labels some of the moves in the neighborhood as tabu. If a non-tabu move exists which improves the objective function value then it is accepted otherwise we invoke the diversification strategy. Frequency diversification employs a record of the frequency with which different moves have been executed to determine the direction in which the search leaves a local optimum.

The dynamic interplay between a steepest ascent with respect to a neighborhood which is modified by the tabu function and diversification strategy to leave local optima make tabu search a powerful tool to solve complex optimization problems. The success of applying tabu search largely depends on a suitable choice of the neighborhoods, an effective tabu function as well as a good diversification strategy. In the following subsections we describe our choices of these tabu search components that enable us to solve the optimization problem stated in the previous section.

6.2 Neighborhood of a State

In our application, a search state consists of a vertex pair (p_1, p_2). To define the neighboring states, we collect all vertices which are directly connected to p_1 and p_2 as defined by the triangular tessellation. The neighborhood of the state then consists of all possible combinations of these vertices. Doing this, the search can be thought of as a wobbling and sliding of the two fingers of the robot on the object's surface.

When starting the search, the "fingers" are most of the time moving directly towards a good grasping configuration. It would therefore be desirable to have a larger neighborhood. Unfortunately, the calculation of the neighborhood is quadratic in the number of its vertices. Thus, the speedup we gain by visiting less states on the way to the maximum is easily lost by using more time for the calculation of a single state. We avoid this problem partially by doing a resampling of the surface and defining "jump points". The density of these jump points is chosen in such a way that each "ordinary" vertex has at least 4 jump points within a range of three edges of the triangulation (see Figure 2). The jump points of p_1 and p_2 are also added to the collection of neighboring vertices and thus their combinations become part of the neighborhood of the present state, too.

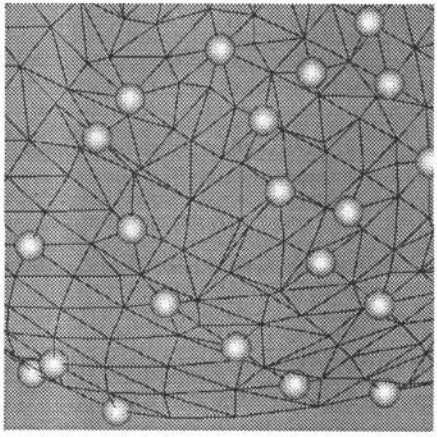

Figure 2: Jump points on a triangulated surface

Note that we are not trying to optimally place these jump points as they are only used for faster moving towards local maxima. As soon as the search approaches a maximum, the jump points will lose their influence.

6.3 Move Attributes and Diversification

Attributes describing changes from one search state to the next, called move attributes, are used to prevent the search from returning to a recently visited optimum. This is done by not allowing moves whose attributes appeared less than t (typically < 10) iterations ago. In our application the move attributes are:

- the orientation of the translation vector $C_{i+1} - C_i$, where C_i is the center of the straight line connecting the two contact points at iteration i,
- the orientation of the straight line connecting the two contact points.

Using these move attributes, we forbid all moves which go in the opposite or same direction as a move done within the last t iteration, unless the orientation is changed. This prevents the search from entering a loop with a cycle time less than t.

Move attributes are ineffective for preventing long term looping. Instead, diversification strategies have to be introduced which push the search into areas of the search space that have not yet or the least been explored. We do this by changing the quality function according to the number of times a state has been visited. If we denote by $n_{p_{1,2}}$ the number of times the state (p_1, p_2) has been visited, and by n_{p_1} (resp. n_{p_2}) the number of times p_1 (resp. p_2) was part of a vertex pair, then our new q has the following form:

$$q_{next}(p_1, p_2) = q(p_1, p_2) - n_{p_{1,2}} \cdot R_{state} - (n_{p_1} + n_{p_2}) \cdot R_{vertex}$$

Here, the term $n_{p_{1,2}} \cdot R_{state}$ penalizes a state that is visited several times and $(n_{p_1} + n_{p_2}) \cdot R_{vertex}$ imposes a penalty if a vertex is often used. The constants $R_{vertex} \ll R_{state} < 1$ are used to tune the diversification. The rational behind using two different penalties is that the first one helps in moving out of a good local maxima, while the second one forces a finger to move if it is stuck in a certain position. It is important to notice that q_{next} is only used for selecting the best next state from all the neighboring states. The goal is still to optimize $q(p_1, p_2)$.

6.4 Tabu Search Algorithm

After having discussed all relevant details of the search, the execution flow can be described as follows:

(A) Calculate jump points. Select an initial state $S = (p_1, p_2)$ with p_1 and p_2 being adjacent vertices. Set $q_{best} = q(p_1, p_2)$.

(B) Compute q and the move attributes for all the states in the neighborhood N of S.

(C) If any q is larger than q_{best}, a new optimum is found. Set $q_{best} = q$ and S_{best} to the corresponding state. Move to the new state by setting $S = S_{best}$ and go back to **(B)**.

(D) Based on the move attributes, remove all states from N that are tabu. For each remaining state, calculate q_{next} and select the state with the highest q_{next} as new state S. Update the list of move attributes. If S_{best} did not change within the last l iterations, it is classified as a prominent optimum: go to **(E)**. Else, continue the search by going back to **(B)**.

(E) As no improvement was made within the last l iterations, check the global constraints (collision etc.) of the grasping configuration S_{best}. If these constraints are not satisfied or q_{best} is lower than a fixed threshold q_{min}, mark S_{best} as tabu for the rest of the search and set $q_{best} = 0$. Go back to **(B)**.

(F) S_{best} is a valid grasping configuration. Stop the search and let the robot execute the grasp.

7 Results

We have run numerous tests on many different objects, among them the ones displayed in Figure 3. To demonstrate the independence of our algorithm from its initialization, we have repeated the search several times, always starting from a different, randomly selected vertex pair. We always find a grasp with a satisfactory quality ($q > q_{min}$). The results are listed in Table 1. For *all examples*, we set the parameters to the following values: $q_{min} = 0.95$, $R_{state} = 0.5$, $R_{vertex} = 0.05$, $t = 7$ and $l = 20$. These settings are not critical, except for q_{min} which controls the acceptance of a grasping configuration and therefore depends on the robot hand.

For the three generic objects (box, cylinder and sphere), our algorithm terminated in less than 40 iterations (mean). As $l = 20$, this implies that a good grasping configuration was found in less than 20 iterations (mean). The advantage over a brute force approach is evident since we need to evaluate the quality measures $q(p_1, p_2)$ for only 1% of all possible vertex pairs (n_{pairs} in Table 1). Finally, a pure steepest ascent method is likely to be insufficient (see n_{down} in Table 1).

For the arbitrarily shaped object (ASO), the large standard deviation of n_{iter} is due to the randomly selected initial vertex pairs. In fact, the minimum n_{iter} was equal 5, whilst the maximum was 618. This indicates that the initial vertex pairs of the search should be selected somehow intelligently by making a rough guess. It appears to be a good heuristic to initialize the search with the two vertices closest to the center of gravity. For this choice, the result for the ASO is $n_{iter} = 44$.

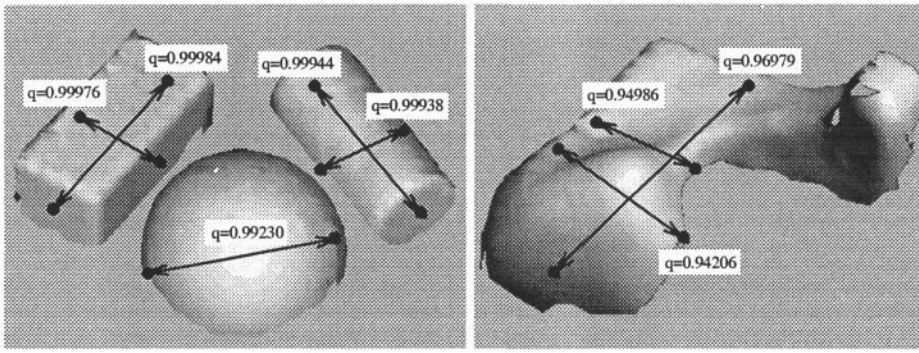

Figure 3: *Rendered 3D descriptions of the sample objects and grasps found by the search.*

object	# of vert.	global q_{max}	q_{best} mean	q_{best} σ	n_{iter} mean	n_{iter} σ	n_{down} mean	n_{pairs} mean	n_s
box	605	0.9998	0.9994	2.8e-05	36.1	6.7	18.4	1.2%	210
sphere	713	0.9923	0.9998	3.5e-05	39.8	8.2	17.6	0.7%	238
cylinder	437	0.9994	0.9920	2.1e-02	29.5	6.1	13.6	1.3%	145
ASO	744	0.9698	0.9530	1.1e-02	117.9	137.0	57.6	1.4%	255

Table 1: *Search results for the test objects:* q_{max} = global max found by exhaustive search, q_{best} = quality of final grasp, n_{iter} = total number of iterations, n_{down} = number of moves that decrease the quality, n_{pairs} = % of vertex pairs for which q was evaluated, n_s = number of samples run.

8 Conclusions

We have presented a novel way of efficiently detecting grasping opportunities on arbitrarily shaped objects. Based on evidence accumulation of several local features, a quality measure is defined for each grasping configuration which then has to be optimized. Since all vertex pairs of an object are admitted as possible contact points for the two finger gripper, this yields a challenging optimization problem. The Tabu-search strategy we employ to solve this problem is able to find good grasping opportunities in only a few searching steps.

We have implemented a complete system which performs all the tasks described in this paper, from data acquisition to the actual grasping. Several scenes have been successfully analyzed.

References

[1] J.-D. Boissonnat. Stable matching between a hand structure and an object silhouette. *IEEE Trans. on Pattern Anal. and Machine Intell.*, 4(6):603–612, November 1982.

[2] F. Glover and M. Laguna. Tabu search. In C. R. Reeves, editor, *Modern heuristic techniques for combinatorial problems*, pages 70–150, Osney Mead, Oxford OX2 0El, GB, 1993. Blackwell Scientific Publications.

[3] J. J. Hopfield and D. W. Tank. "Neural" computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.

[4] K. Ikeuchi and M. Hebert. Task oriented vision. In *Image Understanding Workshop*, pages 497–507, 1990.

[5] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[6] P.G. Mulgaonkar, C.K. Cowan, and J. DeCurtins. Understanding object configurations using range images. *IEEE Trans. on Pattern Anal. and Machine Intelligence*, 14(2):303–307, 1992.

[7] M. Rutishauser and F. Ade. Detecting grasping opportunities in range data. In *ISPRS Intercommission Workshop: From Pixels to Sequences*, volume 30 5W1, pages 237–244, March 1995.

[8] M. Rutishauser, M. Stricker, and M. Trobina. Merging range images of arbitrarily shaped objects. In *Proceedings of the CVPR'94*, pages 573–580, 1994.

[9] S.A. Stansfield. Robotic Grasping of Unknown Objects: A Knowledge-based Approach. *The Intl. Journal of Robotics Research*, 10(4):314–326, August 1991.

[10] L. Stark, L.O. Hall, and K.W. Bowyer. Methods for combination of evidence in function-based 3-d object recognition. *Intl. Journal of Pattern Recognition and Artificial Intelligence*, 7(3):573–594, 1993.

[11] C.J. Tsikos and R.K. Bajcsy. Segmentation via Manipulation. *IEEE Transactions on Robotics and Automation*, 7(3):306–319, June 1991.