SUPER-GRIP: AN EXPERT SYSTEM FOR GRASPING BOXES

Michel Pelletier Vincent Hayward

McGill Research Center for Intelligent Machines
3480 University Street, Montréal, Québec Canada H3A 2A7

ABSTRACT

This paper is concerned with the development of an expert system which determines the best grasp configuration to pick up a rectangular box. This two level system first chooses from a predetermined set of grasps, the ones best suited for the task using a set of rules, and then computes a quality index for each possible configuration. The program is written in LISP and was tested for many different objects and situations. The grasps found are not necessarily optimal but are efficient since they are similar to the grasps a human would choose in the same situation. Because the program gives fast results and doesn't perform tedious computations, the best grip found can easily be used as a sub-optimal solution, or as a starting point for an optimizing program.

INTRODUCTION

A better utilization of robots in the manufacturing industry and in other fields such as aerospace, requires more dexterous end-effectors than those currently available. An *'ideal' end-effector* should be capable of manipulating many different objects even in the presence of uncertainty. The best example of this type of device is of course the human hand.

In the recent years, robotic researchers have been looking at the way humans grasp an object of arbitrary shape, and began to build mechanical devices with properties similar to that of a human hand [1][2][3]. A number of other researchers facing the limitations of technological implementations have settled on designs which, while retaining much of the principles underlying human prehension, bear little resemblance with human hands [4][5][6].

Hunt qualifies mechanical systems such as multi-fingered grasps, as systems with 'superabundant' freedoms [7]. Conventional control engineering techniques, applied to the synthesis of a controller for dexterous grippers, are in general inappropriate to deal with the complexity inherent to such systems [8].

The character of 'superabundancy' of grasps leads to several computational problems for the synthesis of a grasp. For example, Kerr and Roth have shown that the optimal placement of grasp points from the point of view of stability could be reduced to a linear programming formulation [9]. However, only a small numbers of constraints have been taken into account, namely three contact points. Including a larger number of constraints (more fingers, surface contact, reachability, etc.) would quickly lead to the well known computational difficulties. Similarly, graph matching methods applied to the problem of determining a static grasp may lead to an astronomical number of possibilities*.

Instead, we adopted a technique known in Artificial Intelligence as 'generate and test' which provides us with a tool to implement a grasp planner combining two approaches. Roughly speaking, the 'generate and test' strategy consists of splitting a task into two cooperating modules. The first module, in general based on heuristics, is designed to generate candidate solutions without insisting neither on uniqueness (the best solution) nor completeness (will find all solutions). The second module, in general based on analytical models, is designed to verify that the candidate solutions are indeed valid, and in the case of multiple candidates, to rank them and pick the best. This multi-stage strategy can be further improved by the addition of a third module meant to refine the best solution, using a steepest descent algorithm for example.

PREVIOUS APPROACHES

Prehension is a complex operation which has been divided into four basic steps by Tomovic *et al* [8]: (1) Recognition of the object, (2) choice of a grasp, (3) approach and preshaping of the hand, (4) and finally, closure of the fingers and shape adaptation. Sub-problems include: determination of internal forces to ensure stability, determination of the finger-joint motions to produce desired object motions, determination of grasp merits in view of particular tasks, determination of the work-space, determination of feedback control algorithms, *etc.*

We would like to point out that these four basic synthesis steps do not seem to be particular to grasping, but appear to be present in the design and the control synthesis of many complicated mechanical systems. Consider for example the case of the design of a robotic assembly cell. The first step

* If m is the number possible contact points on the object and n the number of contacts the gripper can produce, the number of combinations is $m!/n!(m-n)!$ [10].

CH2809-2/89/0000-0510 $1.00 © 1989 IEEE

is to analyze the parts to be assembled and the second step is to select a mating strategy. Then, the lay-out of the cell follows from the assembly strategy and the kinematic constraints of the robots. In some cases, passively adaptive devices such as RCC's will make up for the model discrepancies and compliant motions will lead to the parts mating. In other cases, compliant motion will be obtained through force feedback. In any case, the basic four steps for dealing with the control synthesis of a superabundant mechanical system are present.

A grasp may also be viewed as a process to utilize the redundancy available in a dexterous gripper. This redundancy is used in all the phases of a grasp, including the grasp control once multi-finger contact is established, as well as for the re-allocation of the grasping as used in haptic procedures. During these phases, redundancy is used to trade-off the range of motion, the capacity to produce velocities and forces, the internal forces, the grasp stability, and the passive compliance [11].

In order to fight the combinatorics, Iberall *et al.* have proposed a virtual finger concept based on psycho-physic observations [12]. Virtual fingers are sets of one or several fingers that can be grouped by pairs to produce oppositions, thus forming an "opposition space" of reduced dimension. However, their analysis, not being based on the underlying physics of grasping, is dependent on the human hand structure.

Another approach to the problem of grasping is to concentrate on the last steps of the grasping procedure. As a consequence, a major effort is put into the design of gripping mechanisms with an extensive built-in capacity to adapt to various objects and situations with little need for external control [6][13].

METHODOLOGY

In this paper, we concentrate on the second problem, the choice of a grasp, in a quasi-static situation. Our approach does not presume of a particular emphasis put onto the design of the gripper or onto its controls, but rather investigates the appropriateness of the generate and test paradigm applied to grasp selection.

The rest of this paper is concerned with the design of an expert system that chooses the best grasp mode to use under different circumstances for one simple primitive shape: the box. This problem is limited but still complicated because of the many factors influencing grasp. The best grasp chosen must take into account the size of the box, its location and constraints, the weight and friction coefficients, the directions in which more strength and mobility are needed *etc.*

An optimal solution to this problem could be found using the method developed by Li *et al.* and Hsu *et al.* [14][15]. Although this technique would give excellent results, it is rather complex and necessitates numerical methods to optimize a multi-dimensional goal function. Due to the computational problems mentioned earlier, the optimal solution may take very long to find by this method alone.

For most usual grasps, this amount of computations may not be necessary, humans don't seem to do anything similar when grasping an object. The human grasping process is partly knowledge-based: we decide how to grasp an object almost instantaneously just by judging its nature. The approach we use here tries to imitate this process: Instead of performing a complete search for the best grasping configuration, we determine the best one from a predetermined set of standard grasp modes (Fig. 1) and approach directions. The information about the strength and mobility of each grasp is precomputed as a function of the box parameters and stored in the grasps database.

The grasps found by our program are not geometrically optimal: the fingers are always positioned in the center of the surfaces when possible, and the palm is considered parallel to the box surface it is facing. This limits the number of possible approach directions to 24. The solutions found may be considered as sub-optimal grasps and could be used as starting points for an optimization search. This system could also be implemented as part of a complete hierarchical prehension system. The development of such systems was addressed by Rao *et al.* and Tomovic *et al.* [16][8].

This expert system is written in LISP and is called SUPER-GRIP. The following sections describe how the program works and how it was implemented. The results from the sample session show examples of grasps chosen by SUPER-GRIP.

DESCRIPTION

The program chooses the best grasp mode to grip a box with a three fingered gripper. The box may have arbitrary dimensions (book, credit card, television...), weight and friction coefficients which are defined by the user. The box may also be constrained by any of its surfaces, i.e., surfaces that can't be used for grasping because they are in contact with the environment (a table for example) or because they will be used for a particular task (the point of a pen for example). The grasp chosen will satisfy all restrictions due to weight and constrained surfaces, and will maximize the quality index of the grasp according to the task requirements. These requirements are determined by the user as the wrench of needed forces in each direction (3 linear forces and 3 torques), and the twist of needed velocities for the task. The input values should range between 0 and 1, where 0 means that no force (or velocity) is needed in that particular direction, and 1 means that force (or velocity) is needed most in that direction. These values describe the shape of the force and velocity task ellipsoids as defined by Li and Sastry, and will be compared to the force and velocity grasp ellipsoids to determine the quality of the grasp [14]. The quality index used is inspired from the one defined by Hsu *et al.* and is defined as follows [15]:

$$Q = \gamma u_t + (1 - \gamma) u_w$$

where Q is the quality index (performance measure), u_t and u_w are the radius of the largest velocity and force task ellipsoids (in the twist and wrench spaces) that can be embedded

in the corresponding grasp ellipsoids, and γ is the selection parameter which has been set to 0.5.

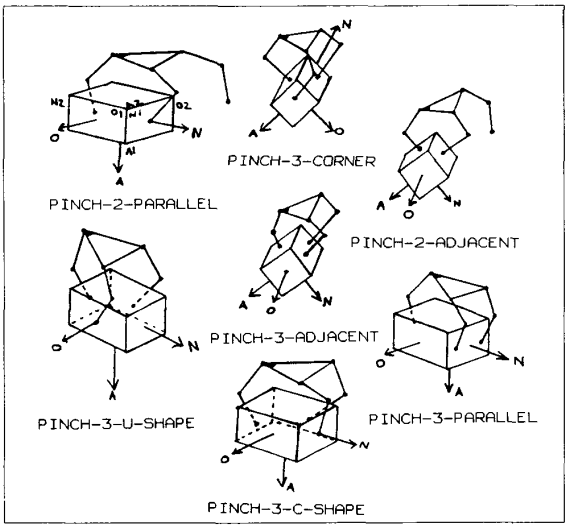

Figure 1 Grasp Modes

The 'generate and test' strategy is performed at two distinct levels in the program. After having pre-processed the user's input into a basic list of assertions, an inference engine (forward chainer) decides which grasps from the database to try, based on heuristic information inside the high-level rules. These rules are the heart of the expert system because they decide what the situation is, and what calculations will be necessary.

The second level is analytical. Once the high-level system decides to try a grasp because it looks good, the low-level system (function TRY) will verify if it is possible. If the grasp is possible, the system computes the position of the fingers on the surfaces. Normally they are placed in the center of the surfaces, but it is not always possible if the box is large. An offset may be necessary, see Fig. 2. Finally, the quality indices for all valid directions of approach of the grasp are computed and remembered.

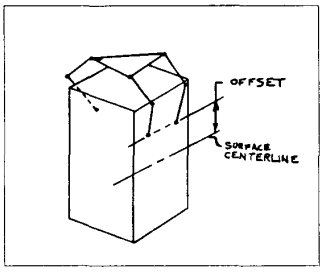

Figure 2 Offset Configuration

All the possible grasps are kept in a list and when the high-level system decides to stop trying grasps, the best one is selected. The system will also indicate if the object can't be grasped successfully. The general block diagram of SUPER-GRIP is shown in Fig. 3.

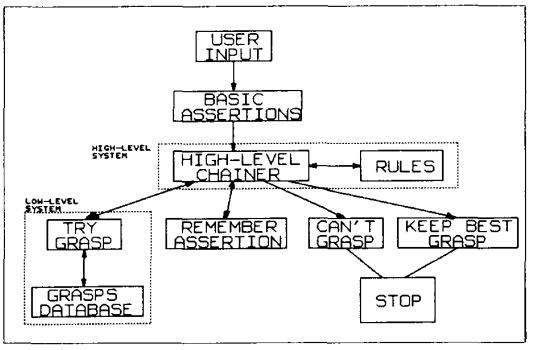

Figure 3 Block Diagram

IMPLEMENTATION

Inference Engine

Before invoking the high-level chainer, the system must have preliminary information and a list of basic assertions deduced from the user's inputs:

- All gripper dimensions and specifications.
- List of box surfaces: X1, X2, Y1, Y2, Z1 and Z2. Where Z2 is the bottom and X1 is opposed to X2, Y1 to Y2, and Z1 to Z2.
- List of grasping surfaces: N1, N2, O1, O2, A1 and A2. These surfaces are related to the gripper and will be assigned to box surfaces according to a *matching*.
- A preliminary list of assertions which includes properties OPPOSED, SMALL (means one dimension is very small), LARGE, SLIPPERY, and CONSTRAINED.
- A list of the 24 possible matchings (directions of approach) between X Y Z and N O A.
- All the information about the box: size, weight, friction coefficients, constrained surfaces, and the force and velocity task requirements.

The high-level rules are processed by a standard forward chainer which has been augmented to support negative sentences and procedural attachement [17]. The syntax of the rule clauses and the assertions has been extended to deal with predicates that contain lists of surfaces that satisfy certain properties, in addition to ordinary predicates (e.g. (NOT PARALLEL (S1 S2))). This feature is necessary for clarity and to avoid repetition in the assertions list. Since all the properties of surfaces used in the program are commutative, the chainer doesn't care about the order of elements in the assertions. To allow the use of negations of assertions inside the high-level rules, the chainer recognizes the negations NO and NOT and uses a form of Closed World Assumption (CWA) based on

the syntactic convention that if a positive statement is not present in the assertion list, its negation is assumed [18]. Since our system is non-monotonic, the order in which the rules are processed is also important. The rules that determine a certain property of the object or its surfaces must be placed before the rules using this property. This feature allows simple high-level rules and prevents from having a very long list of assertions that contains all lists of not parallel surfaces, not adjacent surfaces etc.

The procedural attachement is done in the action clauses. These lines may start with four different functions: REMEMBER, TRY, KEEP and CANT. The REMEMBER function is just like in a standard chainer, it remembers the action as a new assertion, replacing assigned variables if necessary. The CANT function tells the chainer to stop the search, the object can't be grasped. The KEEP function also stops the search, but first finds the best grasp yet in the possible grasps list. The TRY function is the low-level system that tries a grasp mode.

High-Level Rules

The current version of SUPER-GRIP contains 15 high-level rules. We tried to design these rules in an efficient way so that the decision process resembles the human one, and that no unnecessary computations are performed. Only a limited number of possibly interesting grasps should be tried. To achieve this goal, we arranged the rules roughly in the following order:

(1) Obvious cases: The box is too thin, too slippery or too constrained to be picked up. (3 rules)
(2) Availability of surfaces: Determine free parallel surfaces, adjacent surfaces, and free corners. (4 rules)
(3) Poor grasps: no parallel surfaces available. Try the PINCH-2-ADJACENT, PINCH-3-ADJACENT and PINCH-3-CORNER grasps. (3 rules)
(4) Other cases: Try the better grasps and keep the best one. These include the PINCH-2-PARALLEL, PINCH-3-PARALLEL, PINCH-3-U-SHAPE, PINCH-3-C-SHAPE and PINCH-3-CORNER grasps. (5 rules)

An example of each of these rules is shown here. It should be noted that the properties PARALLEL and ADJACENT imply that the surfaces concerned are not constrained, i.e. are free to be used for grasping.

```
(RULE GRIP1
      (IF (SMALL (s1 s2))
          (CONSTRAINED (s1)))
      (THEN (CANT GRASP  TOO SMALL TO PICK UP)))

(RULE SURFACES1
      (IF (OPPOSED (s1 s2))
          (NOT CONSTRAINED (s1))
          (NOT CONSTRAINED (s2)))
       (THEN (REMEMBER PARALLEL (s1 s2))
             (REMEMBER OBJECT HAS PARALLEL SURFACES)))

(RULE GRIP6
      (IF (OBJECT HAS NO FREE CORNERS)
          (OBJECT HAS NO PARALLEL SURFACES))
      (THEN (TRY PINCH-2-ADJACENT)
            (TRY PINCH-3-ADJACENT)
            (KEEP BEST)))

(RULE GRIP8
      (IF (PARALLEL (s1 s2))
          (CORNER (s1 s3 s4)))
      (THEN (TRY PINCH-3-U-SHAPE (?  s3 s1 s2 ?  s4))
            (TRY PINCH-3-C-SHAPE (s1 s2 ?  s3 ?  s4))))
```

Low-level System

The action clauses that try a grasp have the following syntax:

```
(TRY grasp (optional suggested matching for N1 N2 O1 O2 A1 A2))
```

The grasp name must be contained in the grasp database along with all the information needed to determine its quality: grasping conditions, offset computations, surfaces used by the grasp, and strength and mobility of the grasp in all directions. An example of this information is shown here:

```
(SETQ PINCH-2-PARALLEL
    '((USED-SURFACES (N1 N2 A2))
      (CONDITIONS (((DIST '(N1 N2)) < DIST-PINCH) AND
                  ((DIST '(O1 O2)) > WIDTH-HAND) OR
                   (NOT ((CONSTRAINED O1) AND
                         (CONSTRAINED O2))))))
      (OFFSET (LET* ((r (/ (- DIST '(N1 N2)) FINGER-DIST) 2))
                    (depth (SQRT (- (EXPT FINGER-LENGTH 2)
                                    (EXPT r 2))))
                    (offset (POSIT (- (/ (DIST '(A1 A2)) 2)
                                      FINGER-LENGTH
                                      (COND ((> FINGER-DIST
                                                (DIST '(N1 N2)))
                                             depth)
                                            (T 0))))))
                '(0 0 0 0 ,(- offset) ,offset)))
      (WRENCH (1
               (* 2 u)
               (* 2 u)
               (* 2 v)
               (* u (DIST '(N1 N2)))
               (* u (DIST '(N1 N2)))))
      (TWIST  (0.8 0.5 1 0.2 1 1)))
```

The function TRY interprets this information and performs the necessary computations in the following order for all valid assignments deduced from the rule lines:

- Get the next valid matching: There are 24 possible matchings between surfaces X Y Z of the box and the N O A axes of a grasp, which represent the possible approach directions. The program loops through these matchings and tries the valid ones. A matching is valid if the surfaces used by a grasp (e.g. surfaces N1, N2 and A2 for the PINCH-2-PARALLEL grasp shown above, see also Fig. 1) are not constrained. The optional suggested matching in the action clause saves time when the system already knows which surfaces are not constrained. It reduces the number of possible assignments tried by the low-level system.
- Check the grasping conditions: These are the conditions for which a particular grasp is possible. They usually depend on the object's dimensions and friction coefficient.
- Calculate the offset.
- Compute the grip wrench and twist: They determine the force and velocity that a particular grasp can produce on a box in the six possible directions: translation and rotation in N, O and A. The wrench values were deduced from simple static calculations, while the twist was assigned qualitative appreciations of mobility determined by the authors. Before actually using these results, the system must rotate the wrench and twist from the $N\ O\ A$ to the $X\ Y\ Z$ axis system.

- Check if the object can be lifted: The system compares the weight to the available wrench computed previously.
- Compute the quality index: This index was defined earlier as being the sum of the largest force and velocity task ellipsoids that can be embedded in the force and velocity grasp ellipsoids. Before comparing these values, the effects of the offset and the gravity force must be added.
- Store results in the list POSSIBLE-GRASPS.

SIMULATION RESULTS

Many tests were run to test the validity of our system. One of these tests is presented here. A pencil must be picked up from a table and used to write something. The pencil is long in the X direction (12 cm.), and is constrained by the surface touching the table (Z2) and the point with which we want to write (X1). Strength is needed along the pencil's axis, and we need to exert moments perpendicular to the pencil's axis to move it and create shapes. Velocity is also needed to easily move the point in directions perpendicular to the pencil's axis. The following input was derived from these facts:

```
(SETQ BOX '((SIZE 12 2 2)
            (WEIGHT 0 2)
            (u 0 8)
            (v 0 2)
            (CONSTRAINED Z2 X1)
            (FORCE-NEEDED (1 0.8 0.8 0.3 1 1))
            (VELOCITY-NEEDED (0 0.8 0.8 0 0.8 0.8))))
```

The SUPER-GRIP program gave the following results for this particular grasp:

```
$ (forward-chain)
(RULE SURFACES1 SAYS PARALLEL (Y2 Y1))
(RULE SURFACES1 SAYS OBJECT HAS PARALLEL SURFACES)
(RULE SURFACES2 SAYS ADJACENT (Z1 Y2))
(RULE SURFACES2 SAYS ADJACENT (Z1 Y1))
(RULE SURFACES2 SAYS ADJACENT (Z1 X2))
(RULE SURFACES2 SAYS ADJACENT (Y2 X2))
(RULE SURFACES2 SAYS ADJACENT (Y1 X2))
(RULE SURFACES2 SAYS OBJECT HAS ADJACENT SURFACES))
(RULE SURFACES3 SAYS CORNER (Z1 Y2 X2))
(RULE SURFACES3 SAYS CORNER (Z1 Y1 X2))
(RULE SURFACES3 SAYS OBJECT HAS FREE CORNERS))
(PINCH-3-U-SHAPE  IS O.K.:  Q=0.770, MATCHING: (X1 X2 Y2 Y1 Z2 Z1))
(PINCH-3-U-SHAPE  IS O.K.:  Q=0.311, MATCHING: (Z2 Z1 Y1 Y2 X1 X2))
(PINCH-3-C-SHAPE  IS O K    Q=0.436, MATCHING: (Y2 Y1 Z2 Z1 X1 X2))
(PINCH-3-C-SHAPE  IS O K    Q=0.769, MATCHING: (Y1 Y2 X1 X2 Z2 Z1))
(PINCH-3-PARALLEL IS O K    Q=0 860, MATCHING: (Y2 Y1 X2 X1 Z2 Z1))
(PINCH-3-PARALLEL IS O K    Q=0.646, MATCHING: (Y2 Y1 Z2 Z1 X1 X2))
(BEST GRASP IS PINCH-3-PARALLEL WITH MATCHING (Y2 Y1 X2 X1 Z2 Z1))
(THE QUALITY INDEX IS 0 86)
--END--
$
```

DISCUSSION

From these results, we first observe that the object has free parallel surfaces, free adjacent surfaces, and free corners. For this particular grasp, three configurations give good results (index > 0.7): the PINCH-3-U-SHAPE, the PINCH-3-C-SHAPE and the PINCH-3-PARALLEL. Theses grasps are represented graphically in Fig. 4.

The best grasp between the three is the PINCH-3-PARALLEL grasp. This is not surprising since it is the way humans normally grasp pencils (except for the supplementary contact

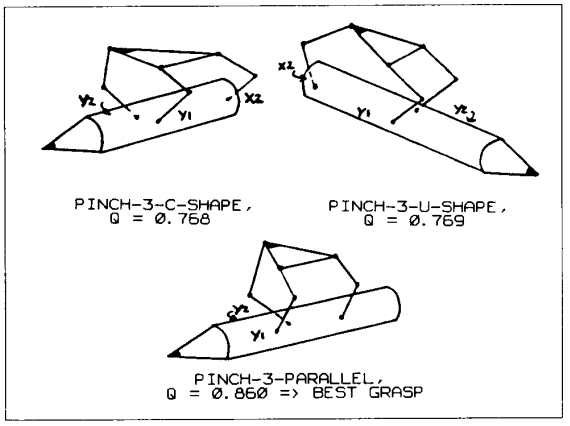

Figure 4 Test Results

at the base of the index). The two other good grasps seem quite uncomfortable at first for the human hand, but for a dexterous gripper, they seem quite acceptable. The mobility of the pencil though, is not as good as in the best one, mostly because of the contact point at the pencil's opposite end. Taking these facts into account, our results seem very reasonable.

One obvious deficiency of SUPER-GRIP is the fact that it is only valid for objects that can be approximated to boxes. Further work should include geometric reasoning on the object to allow classification of different objects in a limited set of primitives. Also, the grip twist definitions in the grasps database should be determined correctly by using the grasp maps of each grip mode. The actual version of the program uses fixed values which do not take into account the size of the box and other important parameters.

Because the rules and grasps are contained in separate files with simple structures and syntax, it is relatively easy to modify them. The rules could be modified to obtain a different strategy for deciding grasps which is more efficient for a given situation, and the grasp list can be augmented with other grasps (lateral grasp for example). The modification of the database of grasps could be made even easier if the offset calculations were replaced by simpler information (for example, the reach of each grasp) that could be interpreted by a dedicated function.

CONCLUSION

In this paper we have described an expert system that chooses task-oriented grasping modes for handling boxes using a generate and test' strategy. After observing the results for different situations, it has been shown that the program is fast and efficient in finding good grasps. Obvious situations are resolved almost instantaneously and the grasps found satisfy the desired specifications very well. The computations are fairly simple and there is no need to try a large number of configurations. The high-level system infers important facts

about the object and its situation, drastically reducing the computational complexity at the lower level. The best grasp found by the program can be used as a sub-optimal solution, or can be further refined by an optimization module.

REFERENCES

[1] S. C. Jacobsen, E. K. Iversen, D. F. Knutti, R. T. Johnson, K. B. Biggers, "Design of the UTAH/MIT dexterous hand", IEEE Int. Conf. Robotics and Automation, San Francisco, Ca, April 1986.

[2] J. K. Salisbury, C. Ruoff, "The design and construction of a dexterous mechanical hand", Proc. ASME Computer Conference, 1981.

[3] L. Bologni, S. Caselli, C. Melchiorri, "Design Issues for the U.B. robotic hand", Nato Advanced Research Workshop on Redundant Robots, Salo, Lago di Garda, Italy, June 1988.

[4] T. J. Doll, H.-J. Schneebeli, "The Karlsruhe Hand", Nato Advanced Research Workshop on Redundant Robots, Salo, Lago di Garda, Italy, June 1988.

[5] A. Rovetta, P. Cavestro, M. Fenzi, "Redundancy of interactions between objects and robotized hands in prehension", Nato Advanced Research Workshop on Redundant Robots, Salo, Lago di Garda, Italy, June 1988.

[6] Y. Umetani *et al.*, "Our contribution in systems and bioengineering (Collected Résumé)", Laboratory of Systems and Bio-Engineering, Tokyo Institute of Technology, 1980.

[7] K. H. Hunt, personal communication.

[8] R. Tomovic, G. Bekey, W. J. Karplus, "A strategy for grasp synthesis with multifingered robot hands", IEEE International Conference on Robotics and Automation, Raleigh, Nc, pp. 83-89, April 1987.

[9] J. Kerr, B. Roth, "Analysis of multifingered hands", Int. J. Robotics Research, Vol. 4, No. 4, 1986.

[10] J. T. Feddeman, S. Ahmad, "Determining a static robot grasp for automated assembly", IEEE Int. Conf. on Robotics and Automation, San Francisco, Ca, April 1986.

[11] V. Hayward, "An analysis of redundancy from several view-points", Nato Advanced Research Workshop on Redundant Robots, Salo, Lago di Garda, Italy, June 1988.

[12] T. Iberall, J. Jackson L. Labbe, R. Zampano, "Knowledge-based prehension: capturing human dexterity", IEEE International Conference on Robotics and Automation, Philadelphia, Pa, pp. 82-87, April 1988.

[13] N. Ulrich, R. P. Paul, R. Bajczy, "A medium-complexity compliant end effector", IEEE Conf. Robotics and Automation, Philadelphia, Pa, April 1988.

[14] Z. Li, S. Sastry, "Task oriented optimal grasping by multifingered robot hands", IEEE International Conference on Robotics and Automation, Raleigh, Nc, pp. 389-394, April 1987.

[15] P. Hsu, Z. Li, S. Sastry, "On grasping and coordinated manipulation by a multifingered robot hand", IEEE International Conference on Robotics and Automation, Philadelphia, Pa, pp. 384-389, April 1988.

[16] K. Rao, H. Medioni, G.A. Bekey, "Robot hand-eye coordination: shape description and grasping", IEEE International Conference on Robotics and Automation, Philadelphia, Pa, pp. 407-411, April 1988.

[17] P.H. Winston, B.K.P. Horn, "LISP: Second Edition", Addisson Wesley, 1984.

[18] M. R. Genesereth, N. J. Nilsson, "Logical Foundations of Artificial Intelligence", Morgan Kauffman, 1987.

[19] J.K. Salisbury, M.T. Mason, "Robot hands and the mechanics of manipulation", Prentice Hall, 1986.