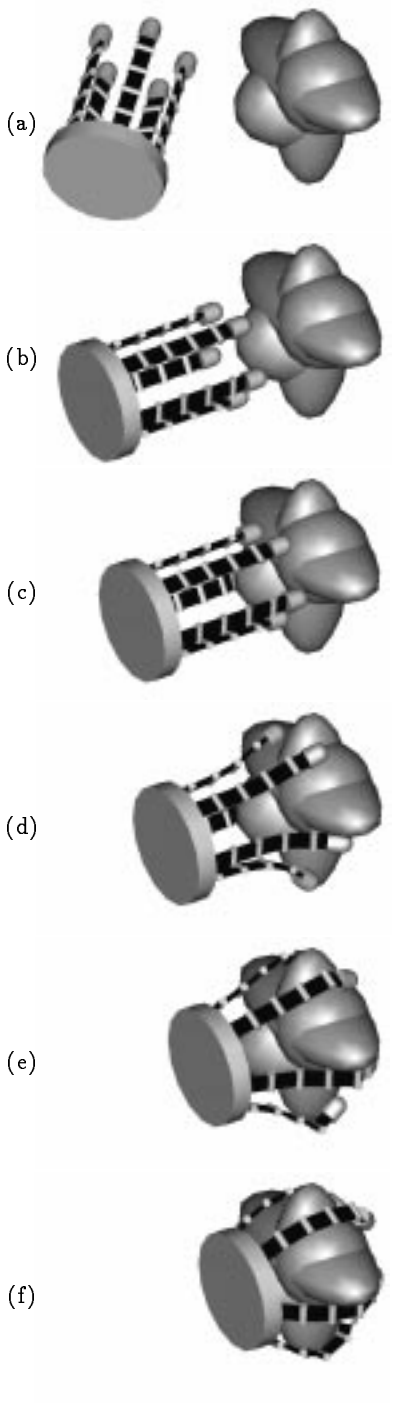

Fig. 4. Example of the algorithm's performance. (a), (b) Phase 1, hand reorientation; (c) to (f) – hand approach and grasp execution: (c) a first contact with the object is made, (d),(e) two intermediate positions. The final position is shown in (f) and in Figure 5.

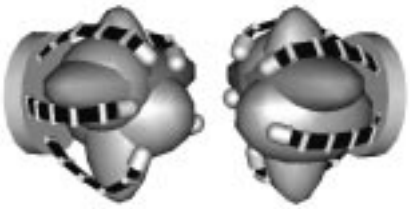

Fig. 5. Two views of the final position of the hand and the object shown in Figure 4(f).

References

[1] C. Laugier, A. Ijel, and J. Troccaz. Combining vision-based information and partial geometric models in automated grasping. In *IEEE Int. Conf. Rob. & Autom.*, Raleigh, NC, March 1987.

[2] Y. Park and G. Starr. Optimal grasping using a multifingered robot hand. In *IEEE Int. Conf. Rob. & Autom.*, Cincinnati, OH, May 1990.

[3] V.-D. Nguyen. Constructing force-closure grasps. *Int. Journal of Robotics Research*, 7(3), June 1988.

[4] N. Pollard. The grasping problem: Toward task-level programming of an articulated hand. Master's thesis, MIT, May 1989.

[5] N. Pollard. Planning grasps for a robot hand in the presence of obstacles. In *IEEE Int. Conf. Rob. & Autom.*, Atlanta, GA, May 1993.

[6] T. Yoshikawa and K. Nagai. Manipulating and grasping forces in manipulation by multifingered robot hands. In *IEEE Int. Conf. Rob. & Autom.*, Sacramento, CA, April 1991.

[7] B. Mishra, J. Schwartz, and M. Sharir. On the existence and synthesis of multifinger positive grips. Technical Report 259, Courant Institute of Mathematical Sciences, New York, NY, 1986.

[8] D. Reznik and V. Lumelsky. Sensor-based motion planning for highly redundant kinematic structures: II. The case of a snake arm manipulator. In *IEEE Int. Conf. Rob. & Autom.*, Atlanta, GA, May 1993.

[9] J. Napier. The prehensile movements of the hand. *Journal of bone and joint surgery*, November 1956.

[10] S. Hirose and Y. Umetani. The development of a soft gripper for the versatile robot hand. *Mechanism and machine theory*, 13, 1978.

[11] M. Cutkosky. On grasp choice, grasp models, and the design of hands for manufacturing tasks. *IEEE Trans. on Robotics and Automation*, 5(3), June 1989.

[12] J. Pettinato and H. Stephanou. Manipulability and stability of a tentacle-based robot manipulator. In *IEEE Int. Conf. Rob. & Autom.*, Scottsdale, AZ, May 1989.

[13] G. Chirikjian and J. Burdick. Design and experiments with a 30-DOF robot. In *IEEE Int. Conf. Rob. & Autom.*, Atlanta, GA, May 1993.

[14] J. Trinkle, J. Abel, and R. Paul. An investigation of frictionless enveloping grasping in the plane. *Int. Journal of Robotics Research*, 7(3), June 1988.

[15] K. Mizra, M. Hanes, and D. Orin. Dynamic simulation of enveloping power grasps. In *IEEE Int. Conf. Rob. & Autom.*, Atlanta, GA, May 1993.

[16] J. Craig. *Introduction to Robotics, Mechanics and Control*. Addison-Wesley, Reading, MA, 2nd edition, 1989.

[17] M. Kaneko and K. Tanie. Contact point detection for grasping of an unknown object using self-posture changeability. In *IEEE Int. Conf. Rob. & Autom.*, Cincinnati, OH, May 1990.

[18] C. Bard, J. Troccaz, and G. Vercelli. Shape analysis and hand preshaping for grasping. In *Int. Workshop On Intelligent Robots and Systems*, Osaka, Japan, November 1991.

[19] R. Fearing. Simplified grasping and manipulation with dextrous robot hands. Technical Report AI Memo 809, MIT, Cambridge, MA, November 1984.

as the palm. During the step each fingertip is expected to move over a fixed distance s. The corresponding step of the palm is of variable length, $\sigma, \sigma \leq s$ (execution of many such steps in rapid succession results in continuous motion).

Each iteration consists of four operations (parts) labeled (a)-(d), which proceed until all elements of `TipState` are set to `Done` – that is, until no fingertips can move any longer in the directions chosen. No motion takes place until the calculations in all four parts are over. Important remark: the discussion below about motion generated in each step should be thought of as computational procedures, not actual motions, until the end of the iteration.

Briefly, part (a) of the iteration in Phase 2 keeps track of which fingers are in contact with the object. Parts (b) and (d) generate motion for the fingers so as to make them wrap around the object. These parts make use of two "canned" procedures, $Head2Tail$ and $Tail2Head$, for snake motion planning directly imported from [8]. Part (c) is responsible for planning a small step forward of the whole unit, (palm + `Frozen` fingers), toward point T along the line P_oT. Finally, part (d) takes care of some gaps between the finger tails and their roots that appear in the previous computational operations. The output from each iteration is a set of step values for all joints of all fingers and for the palm; at this point, the actual step motion takes place, simultaneously in all DOF involved; and then the process repeats.

More specifically, in part (a) of one iteration of Phase 2, the elements of `TipState` currently set to `Frozen` are reset to `Contact` if the corresponding fingertips sense the surface of O. This latter information is obtained from $FTipSensor(m)$, which returns `ON` if the fingertip of finger F_m senses the surface of O, returning `OFF` otherwise.

Part (b) starts out by planning a step motion for the fingertips. This is done by issuing a call to the routine $FollowBoundary(m)$, whose task is to find a point, H, at distance s from the tip of F_m and lying on the surface of O and in the workspace of F_m. This is a well-defined operation since the finger workspace is planar and so only two local directions – left or right – are possible; out of the two, only one, left, achieves the motion desired for finger hugging.

Point H is then passed to $Head2Tail$, which in turn executes a fingertip-to-tail *propagation* [8] for finger F_m. This computational procedure sequentially calculates the positions of all links of the finger so as to avoid collisions with the object while, on the other hand, it make the links "cling" to it. One side effect of this computation is the creation of a small gap between the tail of finger F_m and the corresponding root on the palm. These gaps are then processed in part (c), resulting in a step motion calculation for the palm.

In part (c), the step motion of the palm toward T is computed by averaging the gaps created in part (b). The averaging is done along the axis P_oT; the resulting palm step, σ, is of variable length and is never longer than $s, \sigma \leq s$, due to $Head2Tail$'s property that the tail drift is never larger than the head step [8]. As soon as the first contact between the palm and the object is made – in which case the $PalmSensor()$ routine returns `ON` – part (c) is turned off (i.e., the hand's forward motion stops).

The situation prior to part (d) is as follows – the fingers whose fingertips are currently in contact with the object's surface have been displaced by the step motion as computed in the fingertip-to-tail propagation. Gaps, if any, separate the finger tails from the palm. Both the palm and all fingers which are still `Frozen` have moved along $\hat{p}$ by σ. As a result of this motion, the gaps between finger tails and roots change somewhat, though they are not closed.

Closing these gaps is the function of part (d) which initiates a tail-to-fingertip propagation for each non-frozen finger. This brings the separated tail of each finger back to its root on the palm. This is accomplished by calls to $Tail2Head$ for every finger that is in contact with the object (see [8] for details). Also in part (d) a test for finger *overextension* is carried out: a finger is considered to be overextended if the magnitude of the step planned for its tip after both the $Head2Tail$ and $Tail2Head$ calls is less than some threshold ε. If that is so, the corresponding element of `TipState` is set to `Done`. Finally, the results of the iteration are output for the physical execution of the step motion.

IV. Example

This example illustrates the performance of the grasp planning algorithm $Hugger$, see Figures 4 and 5. The hand has five fingers, each with five links and five revolute joints. All finger links are of equal length. The object to be grasped is a complex curvilinear body, as shown. Figure 4a shows the initial position; Figure 4b, the position after Phase 1. At this point the hand is ready for Phase 2 in which the approach to the object and the grasping take place. At the moment shown in Figure 4c, the first contact with the object (perhaps by one of the fingers) takes place. Figures 4d,e,f show some intermediate positions of the hand; the final position is shown in Figure 5. Observe how different and "optimal" the positions of the fingers are and how firm the resulting grasp looks. All the calculations for sensor-based motion planning, complete with animated graphics motion on the screen of a workstation, was done in real time.

fingers to "face" T; this is done by simply making the axis $\hat{p}$ pass through point T.

Phase 2 – Palm approach and object hugging: The hand moves along the $\hat{p}$ axis toward T, while keeping each finger "frozen", until the first contact with the object is made. Thereafter, as long as a finger is in contact with the object, its fingertip behaves like the head of a snake, following the object surface and "pulling" the rest of the finger with it. As other fingers get involved into this process, a 3D "smothering" effect takes place whereby the object is "hugged" by the fingers from different directions. Meanwhile, to help form a better grasp, the palm continues moving forward, so as to accommodate the fingers' motion, until it comes in contact with the object, at which time it stops. The fingertips continue their motion along the object's surface until they cannot extend any further.

We are now ready to formulate the algorithm more precisely.

III. The *Hugger* Algorithm

The grasp planning algorithm, called *Hugger*, is presented in Figure 3 below; it is written in pseudo-code. As before, M is the number of fingers, and N, the number of links.

The inputs to *Hugger* are (i) an initial hand configuration C_{init} of the whole-sensitive hand, and (ii) coordinates of some point T in the interior of the stationary object O. The shape and orientation of the object are not known. A succession of hand configurations produced by *Hugger* and executed in real time – say, at a rate of 20 to 50 configurations per second – culminates in a final configuration C_{final} in which the palm is brought in contact with the object and the fingers "hug" the surface of O in an enveloping fashion.

For simplicity, assume that in the initial configuration C_{init}, all of the hand's fingers are extended, i.e., $\theta_{mn} = 0, \forall m, n$. Assume also that initially the hand is far enough from the object O, so that it can rotate about its origin P_o with its fingers completely extended, without the danger of colliding with O.

A key data structure used in *Hugger* is an M-element array called `TipState`, whose elements store the current state of a particular finger. Each element of `TipState` takes one of three values: (i) `Frozen`, (ii) `Contact`, or (iii) `Done`, meaning that the corresponding finger is (i) frozen in its extended position, i.e., with all its joint angles set to zero, (ii) currently sensing the object's surface at its fingertip, (iii) completely stretched over the object's surface. Initially, all elements of `TipState` are set to `Frozen`.

The algorithm is broken down into two processing *phases*: (1) hand reorientation and (2) palm approach with object hugging. For hand reorientation, the angle

---

Algm: $Hugger(C_{init}, T)$
Inputs: Configuration vector C_{init} of the hand;
3D coordinates of point T
(T lies inside object O).
Output: A sequence of collision-free configurations,
each representing all $M \cdot N$ links of all
fingers plus the palm.

Phase 1: Hand reorientation:
  Set $\vec{t} = (T - P_o)$;
  Set α = angle between vectors $\hat{p}$ and $\vec{t}$;
  Set α_{step} = small fraction of α;
  Set $\vec{r} = \hat{p} \times \vec{t}$;
  While $\alpha > 0$ do:
    With P_o fixed, rotate hand by angle α_{step} about $\vec{r}$;
    Set $\alpha = \alpha - \alpha_{step}$;
    Output current hand configuration;
  EndWhile.

Phase 2: Palm approach with object hugging:
  While $\exists m$ st TipState[m] $\neq$ Done do:
  a. For each m st TipState[m] = Frozen do:
      If $FTipSensor(m)$ = ON then:
        Set TipState[m] = Contact;
      EndIf.
    EndFor.
  b. For each m st TipState[m] = Contact do:
      Set $H = FollowBoundary(m)$;
      Call $Head2Tail(m, H)$;
    EndFor.
  c. If $PalmSensor()$ = OFF then:
      Set $\sigma = \frac{1}{M} \sum_m [(j_{m0} - R_m) \cdot \hat{p}]$;
      Translate palm by distance σ along $\vec{t}$;
      For each m st TipState[m] = Frozen do:
        Translate F_m by distance σ along $\vec{t}$;
      EndFor.
    EndIf.
  d. For each m st TipState[m] = Contact do:
      Call $Tail2Head(m, R_m)$;
      If $FTipStep(m) < \varepsilon$ then:
        Set TipState[m] = Done;
      EndIf.
    EndFor.

    Output current hand configuration.
  EndWhile.

End *Hugger*.

Fig. 3. Pseudo-code for *Hugger*, a grasp-planning algorithm for a whole-sensitive, multi-fingered hand.

α between the palm's normal vector $\hat{p}$ and the line passing through the palm's origin P_o and point T, is calculated first. With point P_o fixed, the hand is then rotated over the angle α such that the palm's normal is aligned with line P_oT.

The purpose of Phase 2 of *Hugger* is to realize the rest of the operation; in it, the hand approaches the object until the first contact is made; then the fingers slide along the object, with the links clinging to it; meanwhile, the palm moves slowly toward the object so as to accommodate the fingers' motion, until it contacts the object; all this continues until the grasp closes and the fingertips cannot move any further. One iteration of Phase 2 corresponds to one small step which involves all joints of all fingers as well

hand consisting of two rotating jaws and a palm. Their strategy takes into account the object's physical properties such as its weight, and consists of (i) partly lifting the object on its side, (ii) placing one of the jaws under the lifted side, and (iii) picking up the object completely by closing the jaws and bringing it in contact with the palm, thus creating a "smothering" effect. A testbed has been developed by Mizra *et al.* [15] for simulating physical phenomena specific to power grasps (e.g., finger slipping, object rolling, etc.). To our knowledge, the only algorithmic work addressing hug-style grasps in 3D is that by Pollard [4] where a multi-finger hand grasps polyhedral objects in a cylindrical, wrap-like fashion.

The remainder of this paper is organized as follows. In Section II the hand's kinematics and sensing model are described, along with an overview of our approach. The motion-planning algorithm for the problem at hand is formulated in Section III, followed by a computer simulation example, Section IV, which demonstrates the algorithm's performance.

II. The Model; Overview of the Approach

Hand Kinematics. The hand consists of the *palm* and a set of M *fingers* attached to it and labeled counterclockwise as $F_m, m = 1, ..., M$. The palm can be any rigid body; for the sake of simplicity and without loss of generality, assume that the palm is a round platform, as shown in Figure 2. The palm's *front side* (plane), the origin P_o, and the unit vector $\hat{p}$ normal to the palm are defined as shown in the figure and fully describe the palm's position.

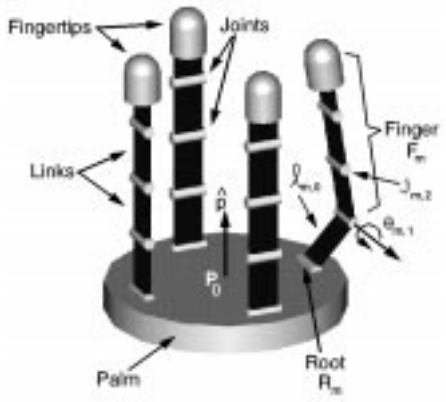

Fig. 2. The model of the hand. The hand has M fingers, each with N links; here $M = 4, N = 4$. Each finger operates in a plane that passes through the vector $\hat{p}$ normal to the palm's plane. Each link l_{mn} can rotate about the corresponding joint j_{mn}; the associated joint angle is θ_{mn}.

The fingers are attached to the palm at their *root* points $R_m, m = 1, ..., M$. Each finger F_m is made up of a chain of N *links* $l_{mn}, n = 0...N-1$, serially connected to each other via revolute *joints* j_{mn}. Joint $j_{m,0}$ coincides with the root R_m. For procedural purposes to become clear later, it is convenient to think of the root as being attached to the palm, and the coinciding end of the finger, called its *tail*, attached to the finger. The endpoint $j_{m,N}$ of F_m is called the *fingertip*. All joint axes in a finger are parallel to each other and to the palm's front plane. For every joint j_{mn}, its associated *joint angle* θ_{mn} is measured about the corresponding axis of rotation. The workspace of finger F_m is therefore confined to the plane perpendicular to the joint axes and passing through vector $\hat{p}$. This planar structure makes a finger identical to an N-link *snake* considered in [8].

In practice the hand would be attached to (the endpoint of) an arm manipulator which would *preconfigure* – that is, bring it into the vicinity of the object to be grasped and orient it (say its vector $\hat{p}$) as necessary. In this work, the hand preconfiguration operation is assumed to be done – we focus on the grasp planning proper. A *configuration* $\mathcal{C}$ of the hand is defined by a $(6 + M \cdot N)$-dimensional vector made up of (i) 6 rigid-body variables specifying the position and orientation of the palm relative to a fixed reference frame, and (ii) M sets of N numbers, each set describing positions of the N joint angles of finger F_m. The (more redundant) representation accepted in this work makes use of *homogeneous matrices* [16]: a configuration $\mathcal{C}$ includes the position/orientation of the palm represented by a 4x4 homogeneous matrix, and the positions of M fingers, each represented by the 3D cartesian coordinates of each of its $(N + 1)$ joints (including the fingertip).

Sensing. The hand is *whole-sensitive*: it possesses tactile sensing, such that any point in its body can detect, on contact, the presence of nearby objects. At all times the hand is aware of its current configuration $\mathcal{C}$ and of the position of point T (see "Object" below).

Object. The object to be grasped, O, is a stationary rigid body. Its shape, dimensions, and orientation in space are unknown. What is known – for example, from the hand preconfiguration stage [18] – is some point T inside the object. The reasoning behind choosing T is to have the object more or less centered relative to the hand; T can be, for example, the object's center of gravity or the center of its projection as viewed from the origin P_o. In general, the algorithm is not sensitive to the choice of T.

Overview of the approach. In the spirit of Fearing [19], who breaks down the grasping operation into four *phases* (approach, initial touch, initial grab, and stabilization), our motion planning strategy is structured as follows:

Phase 1 – Palm reorientation: Since the relative orientation of the hand relative to the object is initially arbitrary, the hand is rotated so as to cause its palm and

tive to the particular arrangement of those few contacts produced. For example, the notion of *force closure* [3, 6], associated to a grasp whose contact points completely restrict the physical displacement of an object, may not be possible for certain objects if there aren't enough contacts available [7]. Additionally, force closure is highly dependent on the particular arrangement of contacts over the surface of the object.[1] The search for techniques robust enough for the fingertip grasp problem, as well as to the model inaccuracies, leads to the idea of *ranges* (rather than points) of acceptable fingertip contact on the boundary of the (polygonal) object [3].

This work attempts to develop a strategy that addresses some of the limitations above. Namely, we ask the following questions:

- Can a strategy be developed so as to "unlock" the hidden power of the hand's many DOF for the purpose of robust grasping? In particular, how can one make use of the other surfaces of the hand and fingers, besides the fingertips, to contribute to the grasp and to reduce its sensitivity to the arrangement of hand/object contacts?
- As in an octopus, an increased number of DOF should result in more flexible and robust collision avoidance and grasping, while the added redundancy should produce more contacts. How can one use this redundancy without jeopardizing (real-time) computational feasibility?
- By bringing in on-line sensory feedback and thus in principle being able to handle unknown or moving objects, can a strategy be developed capable of handling a large number of grasping situations?

The main contribution of this paper is a novel, powerful strategy for grasp planning that addresses the above questions. The approach suggested makes use of the recent work on motion planning for highly redundant kinematic structures. The algorithm $\mathcal{A}_m$, introduced in [8], allows a redundant whole-sensitive arm manipulator (a *snake*) to efficiently operate in a complex planar environment with unknown, arbitrarily shaped obstacles. $\mathcal{A}_m$ also possesses an interesting property that is of special interest in the task at hand (pun intended!), see Figure 1: as the head of the snake follows an obstacle boundary, the snake's body is naturally brought to a tight fit over that boundary.

Note that structurally a single finger can be thought of as a small snake with, say, 3 or 4 (in general N) links and revolute joints, attached to the hand and operating in a single plane.[2] By arranging the planes of the fingers in an appropriate 3D pattern and by making use of $\mathcal{A}_m$'s components, a flexible grasping strategy is produced. The side-effect of collision avoidance and obstacle boundary following becomes "hugging"; the ability to tightly fit over the object's surface helps the fingers distribute their effort for firm grasping.

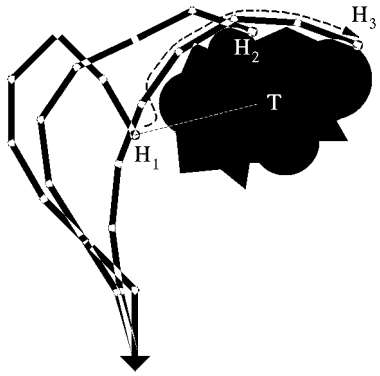

Fig. 1. Shown are 3 superimposed snapshots of snake manipulator motion under the $\mathcal{A}_m$ algorithm [8]. As the head of the snake advances along the obstacle boundary (H_1, H_2, H_3), its links are brought closer and closer to that boundary. At some point the snake becomes completely extended and contacts the obstacle in multiple points.

The idea of enveloping grasps as a means to achieve robustness has been suggested before. The notion of a *power grasp* in which fingers are placed cylindrically around the object as opposed to a *precision grasp* in which the fingertips are made to "pinch" the object goes back to the medical/taxonomic works of Napier [9]. Hirose and Umetani [10] have exploited this idea by designing a hand with two tentacles which are made to "unroll" around a 2D object of interest. This line of work has been recently brought back into the research community's attention in the light of grasp modeling and classification [11]. In a work centered mostly around control techniques rather than on motion-planning, Pettinato and Stephanou [12] consider a multiple DOF snake manipulator whose links are made to wrap in tentacle-like fashion around an object of interest. The idea of wrap grasps has been also pursued by Kaneko and Tanie [17], in the context of contact point detection, and by Chirikjian and Burdick [13] in their experiments with a 30-DOF planar snake: a disk-shaped object is made to rotate by (i) wrapping the snake around the object, and (ii) propagating multiple "peristaltic" waves through the snake's body. Trinkle *et al* [14] have studied enveloping grasps with a rather simplified

[1] It becomes less so as more and more contacts are applied. Another measure is to consider "soft" (friction) contacts although this complicates the analysis.

[2] A human finger presents three links moving in one plane, plus a joint at the root of the finger which allows for some limited adjustment of that plane.

Multi-Finger "Hugging": A Robust Approach to Sensor-Based Grasp Planning*

Dan Reznik and Vladimir Lumelsky
University of Wisconsin-Madison, Madison, WI 53706, USA

Abstract—**We consider the problem of planning the grasp operation for a multi-finger hand. The hand is expected to be able to handle three-dimensional objects of arbitrary unknown shapes as long as they are of "reasonable" size so as to make the grasping operation meaningful. The object geometry is not known beforehand. To provide input information for the interaction necessary, a whole-sensitive hand is assumed – every point of its surfaces possesses tactile sensing. The important property of the problem formulation is that in the grasp produced the hand and the fingers are expected to "hug" the object in a manner a human hand holds an apple – with the maximum contact with its surface for a firm, comfortable grasp. A novel, powerful strategy for real-time grasping is suggested which makes use of recent techniques for highly redundant sensor-based planar motion planning. An example is given that illustrates the performance of the approach.**

I. Introduction

In this paper, we consider the problem of grasp planning. A hand (perhaps attached to an arm manipulator) with multiple fingers is requested to pick up a three-dimensional (3D) object. The object can be of arbitrary shape and in an arbitrary position. Its dimensions are expected to be "reasonable", that is, large or small enough to make the grasping operation meaningful. Its geometry is not known beforehand. To provide input information necessary for interacting with the object, a *whole-sensitive* hand is assumed – every point of its surfaces possesses tactile sensing which provides for the detection of surrounding objects on contact. The objective is to provide a firm, reliable grasp; all computations involved should be feasible in real time.

The important property of our problem formulation is that in the grasp being sought the fingers are expected to "hug" the object in a manner a human hand holds an apple – with the whole palm and with the maximum contact between the surfaces of the object and the fingers, thus producing a firm and comfortable grasp (alternatively, envision an octopus capturing its prey).

Most of the literature on robot grasping concentrates on issues such as hand/object orientation in pick-and-place operations, and dexterity in fine manipulation. These issues usually entail limiting the contact to the tips of fingers only, which typically requires a rather unique positioning of every link of every finger. The price for this precision is computational difficulties, making it hard for one to take full advantage of the redundancy inherent in multi-fingered hands. In other words, a "delicate" grasp technique is likely to lack in (i) the ability to fully exploit complex hand structures, and (ii) the robustness and flexibility needed to handle a large number of cases. In particular, the following are properties found in the existing approaches:

- **Low complexity of the hand:** To keep the problem computationally tractable, hands with few degrees of freedom (DOF) are considered, with an extreme case being the popular parallel-jaw mechanism [1]. While its geometric simplicity is appealing from an algorithmic and hardware design viewpoint, its suitability for general-purpose applications is unlikely. More complex hands usually come with only few fingers (3-5) and few link-per-finger (2-3) topologies. Higher levels of redundancy would only skyrocket computational costs.

- **Complete object models, off-line processing:** Most approaches assume that the object to be grasped is a well-known, algebraic entity, e.g., a polygon/polyhedron or some other set of analytic surfaces. The grasp planning then becomes an off-line computational procedure in which the goal is to optimize some grasping-quality criterion [2, 3, 4, 5]. This framework precludes an on-line feedback, such as from sensing, and an adaptability to previously unknown or changing circumstances.

- **Fingertip grasps:** In planning grasps for multi-finger hands, a typical assumption is that only the tip of each finger may contact the object to be grasped. This leaves unexploited other parts of the hand that could potentially provide additional support and grasp firmness. Combined with only few fingers available (see above), the resulting grasp is not likely to be robust as it will be highly sensi-

*Supported in part by the NSF Grant IRI-9220782 and DOE (Sandia Laboratories) Grant 18-4369C.