PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 43, NO. 3, PP. 207–214 (1999)

STABLE OBJECT GRASPING WITH DEXTROUS HAND IN THREE-DIMENSION

Ervin TÓTH

Department of Control Engineering and Information Technology
Budapest University of Technology and Economics
H–1117 Budapest, Pázmány Péter sétány. 1/D, Hungary
e-mail: ervin@sch.bme.hu

Received: Nov. 30, 1999

Abstract

This paper considers a grasp planning scheme for dextrous hands. The grasp is assumed to be a precise one, which means that only the fingertips of the hand are in contact. The most important algorithm of the grasp planner is the placement of contact points in the presence of friction. Based on a heuristic search, a number of grasp configurations are generated. A proposed method for evaluation of the configurations and determination whether a grasp is a force closure, is introduced. These algorithms are used in the experimental control system of an industrial robot, which the dextrous hand is attached to. A two-level robot programming language, which was written for the robot-hand system, is briefly introduced.

Keywords: grasp planning, grasp evaluation, robot programming.

1. Introduction

Form and force closure have been discussed in the literature for many years. However, the distinction between the two has often been unclear. The difference lies in the way how the contact between the object and the constraining bodies is modelled. Form closure means that if enough constraints are placed in a motion of an object, then the object is immobilized. Force closure means that any external wrench (i.e. force and torque) can be balanced by the proper combination of finger forces. RIMON and BURDICK have shown that in certain circumstances force and form closure are equivalent to each other [5]. In this paper we focus on the computation of the force closure.

At our institute a language called SRPS (Simple Robot Programming System) has been developed for the TUB-PC three-fingered dextrous hand attached to a NOKIA-Puma 560 industrial robot. The most important feature of the language is that it has two levels: a high, task-oriented level and a low, motion-oriented one. On the higher level indirect commands exist, such as ‘Grasp a given object’ or ‘Move the dextrous hand above a given object without colliding with the environment’. The language is upwardly compatible, that is, on the high level all the instructions of the low level can be used. The high-level part, naturally, contains commands which

require sophisticated algorithms. This paper demonstrates the grasp planning and evaluating algorithms in detail.

2. The Object Model

A virtual reality display is used to visualize the actions of the robot for the human operators and this usually has to be done in real-time. It is also a requirement to determine that the robot can do the desired action, so that it does not clash with itself and/or the surrounding items of the environment. Hence an efficient multi-level collision-detection algorithm was implemented.

There are two fundamentally different ways to view the robot during the visualization: it can be an overlaid image with the real camera images or can be a view from an arbitrary viewpoint. In order to put the virtual world together with the real environment, the calibration of the system is required. It means that the parameters of a camera, e.g. position, orientation, focal length, etc. are identified based on pictures taken by a real camera, and used in the visualization stage [1]. The system runs on a Windows NT workstation and was developed using the OpenGL graphic library for realtime display and Visual C for all the other computations.

Our system uses boundary representation (B–Rep) scheme to describe objects. For grasp computations the fingertips are treated as a sphere. The normal vector of the surface element touched by the fingertip, however, not necessarily equals the normal of the triangle. For curved objects vertex normals are defined, which are the weighted sum of the triangle normals in a vertex shared by the neighbouring triangles. The weighting factors are the areas of the triangles (the vertex normals are not of unit length). The computed normal of a surface element of a triangle is the linear combination of the three surrounding vertex normals. The arbitrary surface normal in a certain point is generated by linearly interpolating and normalizing the vertex normals along the surface.

The contact force computation for simulation is based on the penetration of the fingertip model to the object model. Let the penetration vector, which is parallel with the surface element normal and points from the surface to the deepest point of the finger, be $\mathbf{p}$. The contact force is $\mathbf{F} = -\left(K_c + B_c \cdot \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t}\right) \cdot \sqrt{|\mathbf{p}|} \cdot \mathbf{s}$ where K_c specifies the nonlinear stiffness of the material and B_c is the damping [2].

3. The SRPS Language

Our goal was to develop a robot programming language which is capable of describing complex motions, e.g. the robot should be controlled either in joint coordinates and as a function $f(x, y, z, t)$, too. If the system is used without external information, that is, for simulation, the contact forces have to be simulated. In our case it means collision and minimum-distance computations between the model of the dextrous hand and the surrounding objects. The language is a frame system, which

means that certain actions do not need to be directly defined, only the starting and ending positions and the path and grasp planner will do the rest automatically.

Some parts of the functions implemented in the SRPS can be found in the ARPS, the programming language of the NOKIA Puma robot. Certain programs written in SRPS can (with minor modifications) be transmitted to the ARPS interpreter. The new functions mainly deal with the control of the robot hand (high-level grasp synthesis) because the ARPS is not applicable for this task. Since the robot carries items during its movement, condition-systems had to be developed which determine them in a given configuration if the hand holds the object or not.

With the graphic model the reference points of the robot track can be generated. It is especially important when the robot cannot move free because the environment contains obstacles. The virtual robot can be positioned near the desired end positions, then collision-free configurations must be manually found. From now on, the path planning algorithm can find the movement between the end points so the operator does not have to find and teach it. In SRPS level this means that there are indirect moving commands between endpoints. The output of the path planning is the complete SRPS program, which contains simple (direct) instructions only, which can be directly transmitted to the robot controller. The reference points in both stages are collision-free of course.

To demonstrate the possibilities of the SRPS, a few low-level functions are listed below.

LOAD – used to load pre-defined coordinates. These are the Denavit–Hartenberg joint (wrist) coordinates. In the SRPS program a coordinate-configuration record is referred to with its number.

FRAME/UNFRAME – defines/deletes a new coordinate system in which the following commands work. This is used for navigation close to the manipulable objects, using their own coordinate system.

GO, GOS – moves the robot between two reference points. With *GO* the path planning is done in joint coordinates, with *GOS* is along a straight line in 3D. The starting position is the actual position of the robot, the end coordinate is given with its number. If a collision is found during the movement, the program stops with an error message.

GRASP – directs the hand into one of the loaded hand positions. If a segment of a finger collides with an object, the phalanx stops. The other phalanxes and fingers move further until another collision or the desired position. The command has several parameters, such as maximum contact forces, and list of the segments which can touch the object. There are three simplified versions of this command, such as *PINCH*, *SNAP* and *GRIP* [3]. These commands move the fingers into predefined positions, so these do not need to be taught. Furthermore, these have much smaller numbers of parameters.

OPEN – straightens the fingers and translates them to the edge of the palm. During the movement collision must not occur. (If so, the program shows an error message.)

The high-level functions:

GOIN (Go indirect) – this command is not interpreted by the ARPS but is replaced by the path planning algorithm with series of *GO* and *GOS* commands.

GRASPIN (Grasp indirect) – directs the grasp planner to develop a grasp. The grasp planning algorithm tries to find a prehensile grasp on the object. It takes into account the properties of the object, e.g. frictional coefficient, center of gravity, parts of its surface that can be touched, and orientation constraints. Furthermore, it considers the obstacles around the object with respect to the initial and final position of the robot arm before approaching and after grasping the object.

4. Contact Point Generation

The theory of grasp planning is also studied for a long time. Generally, the object to be grasped has arbitrary shape, given in a form of geometric model, transformed from a modeller software (CAD) or derived from sensor data. Because of the big amount of data required to describe even a medium-complex object, a direct (closed-form) solution for finding the optimal grasp does not exist. To overcome this, we assume that using heuristics to locate contact points on the object, a number of grasp candidates can be found, the best of which is selected. Precision grasps are considered (only the fingertips, which are hemisphere-shaped, touch the object). Unfortunately, it cannot be told how close our best candidate is to the optimal grasp. However, a number of measures to qualify grasps has been proposed, e.g. [4] , that measures which external forces can be balanced by a grasp. From this point of view, the best ones are the force-closure grasps, which resist all external forces and torques. This induces a special problem: the TUB-PC hand has three fingers, and a 3D force-closure grasp requires at least four contact points. The solution is to find a concave corner on the object, which can provide more than one contact point for a single fingertip.

Our contact point generator has two fundamentally different algorithms, one for the form-closure grasp generation, the other for arbitrary grasps. The planner decides whether a form-closure grasp is possible, and if not, runs the second algorithm. First, concave corners are searched. A corner (vertex) is concave, if all the edges meeting in the vertex are on the same side of a plane, and the plane, in the vicinity of the vertex, lies inside the object. The resulting grasps can be divided into three categories: a) which contain only concave corners as grasp points, b) which contain no concave corners, c) mixed. The latter ones can be either form or force closures. Only those concave vertices are considered which are accessible for the fingertip of the dextrous hand. Next, the vertex normals of the concave corners are searched for triplets $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$, where $(\mathbf{v}_1 \cdot \mathbf{v}_2)$, $(\mathbf{v}_2 \cdot \mathbf{v}_3)$, $(\mathbf{v}_3 \cdot \mathbf{v}_1)$ are all close to -0.5, which means that these vectors span a quasi-regular triangle. (The triplets are assigned a value w: $\sum_{\substack{i=1,2,3 \\ j=2,3,1}} (-0.5 - (\mathbf{v}_i \cdot \mathbf{v}_j))^2$ and are sorted in ascending order.) If the smallest w is larger than a given $u_{\max}$ value, the algorithm fails. We found

out that a reasonable value for $w_{\max}$ is 1. The remaining ones ($w < w_{\max}$) become grasp candidates.

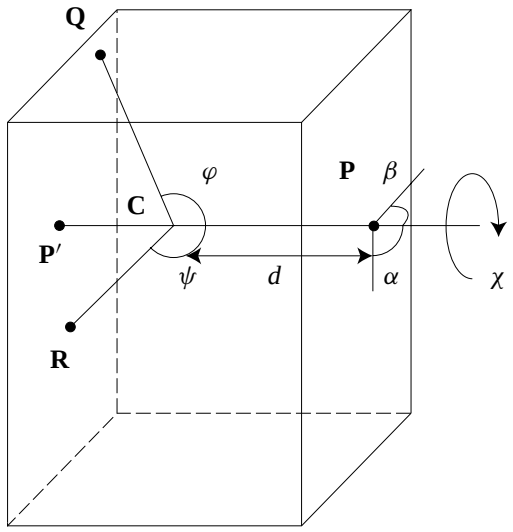

Fig. 1. Parameters that alter the grasp star

The second algorithm tries to place a Y-shaped 'grasp star' so that the penetration points on the object surfaces are the contact points. First, a surface is picked. Through its center point **P** a ray is shot inside the object which is parallel to the surface normal. Between the (farthest) penetration point **P′** on the opposite side, and the starting position, a **C** point is picked so that $\mathbf{PC} = 2\mathbf{P'C}$. From **C** two rays are shot, their intersection point with the object is **Q** and **R** (the angles between **CP**, **CQ** and **CR** are 120 degrees, see *Fig. 1*). At this point, we constructed a three-tip star whose tips are the contact points. The algorithm checks whether the **CP**, **CQ** and **CR** lines are inside the corresponding friction cones. If so, a grasp candidate is ready; if not, we try to compensate with altering the star. There are a number of possibilities: the **C** point can be altered, the angles of the star can be modified, and the star can be rotated around the **CP**, **CQ** and **CR** lines. These operations take into account the surface normal in the contact points so that the accumulated deviation from the surface normals becomes smaller. Random modification is also applied, thus the algorithm resembles the simulated annealing optimization method. During the algorithm, the following criteria are checked: a) two contact points cannot be too close to each other, b) the contact points must be reachable for the hand, c) a contact point cannot be too close to an edge.

We generate several grasp candidates. If possible, the starting position is not only a surface point, but also a vertex normal of a concave corner, resulting a mixed type grasp. Finally, the palm position of the dextrous hand is calculated for the grasp candidates. The palm position is selected so that the accumulated distance

of the palm and the surrounding objects is maximal. The calculation of the finger positions is an inverse geometry task: a finger has 3 degrees of freedom.

5. Grasp Evaluation

The necessary and sufficient conditions of stable grasp are checked. The static equilibrium of fingertip forces and Coulomb friction are considered. The necessary and sufficient conditions are: a) total moment of fingertip forces is zero, b) total force of fingertip forces is zero, c) each fingertip force is in the friction cone. The first condition is satisfied by selecting the fingertip forces so that they intersect in **C**. The necessary and sufficient condition for the second is:

$$e_i^T(e_j + e_k) \le 0; \quad e_j^T(e_i + e_k) \le 0; \quad e_k^T(e_i + e_j) \le 0,$$

where e_i is the unit vector of the i^{th} fingertip force. The third condition is a restriction about Coulomb friction. The method we use, proposed in [6], calculates a measure by determining the set of external wrenches (grasp wrench space, GWS) that can be resisted by distributing one unit force over all grasp points. For linear computations, the set of forces within the friction cones at contact point i are approximated by a linear combination of a finite set of n unit force vectors $f_{i,j}$ at the friction cone boundaries:

$$f_i = \sum_{j=1}^{n} \alpha_{i,j} \cdot f_{i,j}, \quad \alpha_{i,j} > 0, \quad \sum_{i=1}^{n}\sum_{j=1}^{k} \alpha_{i,j} \le 1.$$

The resulting generalized force (wrench) at the i^{th} contact can be expressed as

$$w_i = \sum_{j=1}^{n} \alpha_{i,j} \cdot w_{i,j}, w_{i,j} \begin{pmatrix} f_{i,j} \\ \lambda \cdot (r_i \times f_{i,j}) \end{pmatrix},$$

$w_{i,j}$ is called primitive contact wrench. The GWS can be calculated as $GWS = \text{ConvexHull}\left(\bigcup_{i=1}^{n} \{w_{i,1}, \ldots, w_{i,m}\}\right)$. The time-consuming calculation of the convex hull can be sped up applying incremental calculations [6]. This means that in the beginning of the convex hull computation each friction cone is represented only by three primitive contact wrenches. Those cones which take part in spanning the weakest surface of the convex hull, will be supplemented by additional primitive contact wrenches, thus making the cone approximation more precise. (The weakest surface is the one which lies closest to the origin.) This process continues until the weakest plane converges. The grasp measure is the radius of the maximal ball which can be drawn inside the convex hull. To establish whether a grasp is a force closure, the following condition is checked [7]: if the origin of the wrench space (R^6) lies exactly inside the convex hull of the primitive contact wrenches, then the grasp is a force closure.

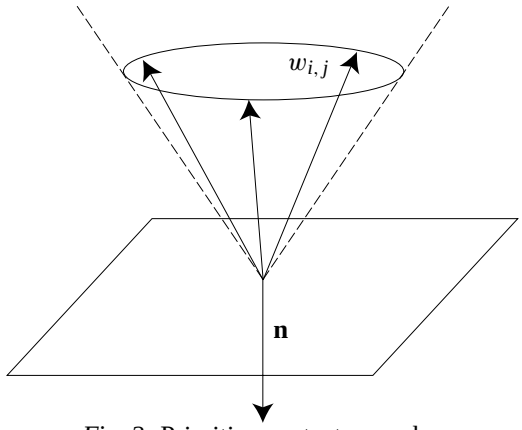

Fig. 2. Primitive contact wrenches

6. Conclusion

This paper proposed a method for finding and evaluating stable grasps on an arbitrary shaped object. Extended polygonal models supplied by material properties are applicable for contact force simulation. A two-level robot programming language has been introduced to describe complex operations. As a part of this language, a grasp planning algorithm was introduced, which generates many grasp candidates based on a heuristic search. Finally, an evaluation function for grasp stability was shown. So far we assumed that the geometric object model is fully known. In real environments, this requirement often cannot be satisfied. In the near future the generation of a partial object model from sensor (mainly visual) data is considered.

Acknowledgement

Support for the research for stable object grasping is provided by the Hungarian National Research Programs under grant No. FKFP 0417/1997 and OTKA T 029072.

References

[1] TÓTH, E. – TÉL, F.: Intelligent Robot Control System with Graphic Model Based Programming and Stereo Vision. In: *Proc. IEEE Int. Conference on Intelligent Engineering Systems*, Poprad, Slovakia, 1999, pp. 57–62.

[2] MAEKAWA, H. – HOLLERBACH, J. M.: Haptic Display for Object Grasping and Manipulating in Virtual Environment. In: *Proc. IEEE Int. Conference on Robotics & Automation*, Leuven, Belgium, 1998, pp. 1586–1592.

[3] LANTOS, B.: Some Possibilities to Increase the Intelligence in Robot Control Systems. *Proc. IEEE Conference on Intelligent Engineering Systems*, Vienna, Austria, 1998, pp. 7–18.

[4] FERRARI, C. – CANNY, J., Planning Optimal Grasps. In: *Proc. IEEE Int. Conference on Robotics & Automation*, Nice, France, 1992, pp. 2290–2295.
[5] RIMON, E. – BURDICK, J.: On Force and Form Closure For Multiple Finger Grasps. *IEEE Trans. on Robotics and Automation*, 1996.
[6] BORST, C. – FISCHER, M. – HIRZINGER, G., A Fast and Robust Grasp Planner for Arbitrary 3D Objects. In: *Proc. IEEE Int. Conference on Robotics & Automation*, Detroit, Michigan, 1999, pp. 1890–1896.
[7] MISHRA, B. – SCHWARTZ, J. T. – SHARIR, M.: On the Existence and Synthesis of Multifinger Positive Grips. Algorithmica, Special Issue: *Robotics*, **2** (1987), pp. 541–545.