FAST GRASP PLANNING AND EVALUATION FOR DEXTROUS MANIPULATION

Ervin Tóth

Budapest University of Technology and Economics
Department of Control Engineering and Information Technology
Pázmány Péter stny. I/D, room 313; H–1117 Budapest, Hungary
ervin@sch.bme.hu

Abstract: This paper presents a contact point placement algorithm, which is used in an experimental control system of an industrial robot equipped with a dextrous hand. The algorithm, apart from the geometric model, takes into account other properties of the object, e.g. frictional coefficient, parts of its surface that can be touched, and orientation constraints. Furthermore, it considers the obstales around the object with respect to the position of the robot arm before grasping the object.
A two-level robot programming language, which was written for the robot-hand system, is briefly introduced. *Copyright © 2000 IFAC*

Keywords: Robotic manipulators, robot programming, evaluation.

1. INTRODUCTION

Form and force closure have been discussed in the literature for many years. However, the distinction between the two has been often unclear. The difference lies in the way the contact between the object and the constraining bodies is modeled. 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. It can be proved that in certain circumstances force and form closure are equivalent to each other (Rimon and Burdick, 1998). In this paper the computation of the force closure is focused.

The goal was to make a robot control system, which uses the detailed model of the robot and the environment for path/grasp planning, visualization and object recognition. The graphic model has to be detailed enough for precise computations but also these computations must be performed real-time. The control system is responsible for handling of control signals, achieves the path planning and develops object grasping. The robot program is part of the robot controller. It is a sequence of simple operations, which build the robot task. In addition to the traditional information (joint variables) the control system uses the result of the vision system such as the type, position and orientation of the recognized objects.
With the use of the graphic models of the virtual reality, efficient high-level robot programming and simulation is available. A two-level language has been developed, which is used to describe robot tasks. The higher (strategic, task-oriented) level of the language contains indirect commands, which will be replaced by the path planner by direct movement and grasping commands based on the visual and other sensor information (some sensor data may come from a simulator). Therefore on the second level of the language there are only direct movement commands. The language is called SRPS (Simple Robot Programming System), and it is responsible for controlling a 6-degree-of-freedom industrial robot equipped with a TUB-PC three-fingered dextrous hand. 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 closure computing algorithms in detail.

2. CONTACT FORCE SIMULATION

The virtual reality is used to teach the corner points of the movement for off-line programming and display the actions of the robot for the human operators. The VR visualization 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 (Tél and Tóth, 2000). 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. During VR visualization there are two different methods to view the robot and its environment: it can be a view from an arbitrary user-defined viewpoint, or it can be an overlaid image with the real camera images. In order to put the virtual world together with the real environment, the calibration of the system is required. Since the machine vision part of the robot controller does not use calibrated cameras, camera calibration process is achieved in order to display the results in the original camera images (Tél, 1998). 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. The fidelity of the virtual reality is essential, since the robot operator's decisions are based on visual information.

The VR visualization and other computations use boundary representation (B–Rep) scheme to describe objects, that is, the models are made out of flat surfaces. For the sake of simplicity, these surfaces are stored as a sequence triangles. It is assumed that a fingertip touches at most one triangle of the object at a time. For grasp computations the fingertip is treated as a sphere, however, in the visualization level, it remains a B–Rep. With these assumptions the minimum–distance calculations required for contact force simulation are simplified to finding the distance between a plane and a sphere. For curved objects the surface normal is generalized so the normal vector of the surface element touched by the fingertip is not necessarily equal to 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). With the notations of Fig. 1. $\mathbf{v}_i$ are the vertex normals, $\mathbf{s}$ is the surface element normal to be computed, a_i, b_i, c_i are distances of the inspected point from the vertices and edges along the line between a vertex and the surface point. The surface element normal can be expressed as

$$\mathbf{s}' = \frac{\mathbf{v}_1 \cdot a_2}{a_1 + a_2} + \frac{\mathbf{v}_2 \cdot b_2}{b_1 + b_2} + \frac{\mathbf{v}_3 \cdot c_2}{c_1 + c_2}, \quad \mathbf{s} = \frac{\mathbf{s}'}{|\mathbf{s}'|} \qquad (1)$$

which means that the vertex normals are linearly interpolated and normalized along the surface.

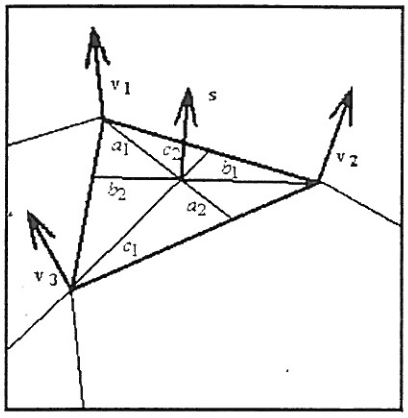

Fig. 1. Surface normal for curved objects. The computed normal of a surface element of a triangle is the linear combination of the three surrounding vertex normals.

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

$$\mathbf{F} = -\left(K_c + B_c \cdot \frac{d\mathbf{p}}{dt} \right) \cdot \sqrt{|\mathbf{p}|} \cdot \mathbf{s} \qquad (2)$$

where K_c specifies the nonlinear stiffness of the material and B_c is the damping (Maekawa and Hollerbach, 1998).

3. THE SRPS LANGUAGE

The robot programming language is capable of describing complex motions, e.g. the robot can be controlled either in joint coordinates or in 3D as a function $f(x,y,z,t)$. Furthermore, certain flexibility had to be provided to handle unknown data, e.g. in case of gripping often it is not known how much the robot fingers have to be (geometrically) deflected. The result is the SRPS, which was tested on the VR model of the NOKIA–Puma 560 industrial assembly robot and the TUB–PC three–fingered dextrous hand. If the system is used without external information, that is, for simulation, the contact forces have to be simulated. In this case it means collision and minimum–distance computations between the model of the dextrous hand and the object to be manipulated. The language is a frame system, which means that certain actions do not need to be directly defined, only the starting and ending

positions are necessary, and the path and grasp planner will do the rest.

Some parts of the functions implemented in the SRPS can be found in the new version of the ARPS, the programming language of the 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 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's especially important when the robot can't 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. Now the path planning algorithm is used to find the movement between the end points so the operator doesn't 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.

To demonstrate the usage of the SRPS, a brief introduction to the low-level commands is given bellow.
LOAD – used to load predefined coordinates. These are stored as the Denavit–Hartenberg parameters.
FRAME/UNFRAME – defines/deletes the coordinate system in which the commands work. This is used for navigation close to the manipulatable objects, using their own coordinate system. The frame hierarchy works a stack of transformation matrices.
GO, GOS – moves the robot between two reference points. With *GO* the path planning is done in joint coordiantes, with *GOS* is along a straight line in 3D. The starting position is the actual position of the robot, the end coordinate is given as a parameter. If a collision is found during the movement, the program stops with an error message.
GRASP – Directs the hand into one of the pre- or user-defined hand positions. If a segment of a finger collides with an object, the phalanx stops. The other phalanxes and fingers move further to 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* (Lantos, 1998). These commands move the fingers into predefined positions, so these do not need to be taught. Furthermore, these have much smaller number 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 are the following:
GOIN (Go Indirect) – this command is not interpreted by the ARPS but is replaced by the path planning algorithm with series of GOs and GOS'.
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. GRASP PLANNING

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, direct (closed-form) solution for finding the optimal grasp does not exist. To overcome this, it is assumed that using heuristics to locate contact points on the object, a number of grasp candidates can be found, of which the best is selected. Precision grasps are considered (only the fingertips, which are hemisphere-shaped, touch the object). Unfortunately it can not be told how close is the best candidate to the optimal grasp. However, a number of measures to qualify grasps has been proposed, e.g. (Ferrari and Canny, 1992), which measures what 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 form-closure grasp generally 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.

The contact point generator has two fundamentally different algorithms, one for 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, convace 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, lays 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 is searched for triplets $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$, where $(\mathbf{v}_1.\mathbf{v}_2)$,$(\mathbf{v}_2.\mathbf{v}_3)$, $(\mathbf{v}_3.\mathbf{v}_1)$ (where . denotes the dot product) are all close to –0.5, which means that these vectors span a quasi-regular triangle. The triplets are assigned a value w

$$w = \sum_{\substack{i=1,2,3 \\ j=2,3,1}} \left(-0.5 - \left(\mathbf{v}_i \cdot \mathbf{v}_j\right)\right)^2 \qquad (3)$$

and are sorted in ascending order. If the smallest w is larger than a given w_{max} value, the algorithm fails. A reasonable value for w_{max} is 1. The remaining ones ($w < w_{max}$) become grasp candidates.

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 **PC** = 2**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. 2.).

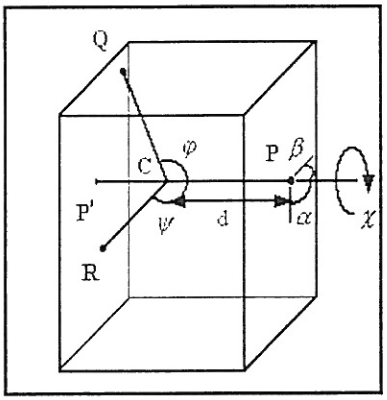

Fig. 2. Parameters that alter the grasp star: center point C (used for center of gravity tropism); angles α, β, χ and distance d (at each contact point, used for friction cone center tropism); φ, ψ (used for optimal hand configuration).

At this point, a three-tip star is constructed 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 found; if not, compensation with altering the star is performed. 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 a) the surface normal in the contact points so that the accumulated deviation from the surface normals becomes smaller;

b) the distance of point **C** from the center of gravity of the object;

c) the regularity of the **PQR** triangle.

Random modification is also applied, thus the algorithm resembles to the simulated annealing optimization method. During the algorithm, the following criteria are checked:

a) two contact points can not be too close to each other;

b) the contact points must be reachable for the hand;

c) contact points can not be too close to an edge.

It must be noted that certain elements of the object may be excluded from the search, e.g. those which are too close to a supporting surface.

Several grasp candidates are generated. If possible, the starting position is not only a surface point, but 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. Calculation of the finger positions is a redundant inverse geometry task: a finger has has 4 degrees of freedom.

5. GRASP EVALUATION

The necessary and sufficient conditions of the stable grasp are checked. The static equilibrium of fingertip forces and Coulomb friction is considered. The necessary and sufficient conditions are (Yoshikawa and Nagai, 1991):

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:

$$e_i^T\left(e_j + e_k\right) \le 0;\quad e_j^T\left(e_i + e_k\right) \le 0;\quad e_k^T\left(e_i + e_j\right) \le 0 \qquad (4)$$

where e_i is the unit vector of the i^{th} fingertip force. The third condition is a restriction about Coulomb friction.

The algorithm of grasp evaluation calculates a measure by determinig the set of external wrenches (grasp wrench space, *GWS*) that can be resisted by distributing one unit force over all grasp points without fingers strarting to slide (Borst, *et al.*, 1999).

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:

$$f_i = \sum_{j=1}^{n} \alpha_{i,j} \cdot f_{i,j},\quad \alpha_{i,j} > 0,\quad \sum_{i=1}^{n}\sum_{j=1}^{k} \alpha_{i,j} \le 1 \qquad (5)$$

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

$$w_i = \sum_{j=1}^{n} \alpha_{i,j} \cdot w_{i,j}, w_{i,j} = \begin{pmatrix} f_{i,j} \\ \lambda \cdot (r_i \times f_{i,j}) \end{pmatrix} \quad (6)$$

where $w_{i,j}$ is called primitive contact wrench (Fig. 3.). The factor λ is used to transform the torques to force dimension so that the length of the 3D force component and the transformed torque component falls into the same order of magnitude.

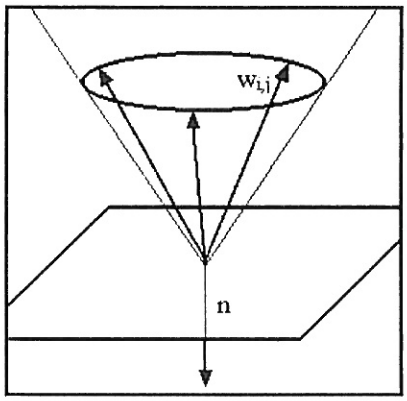

Fig. 3. Primitive contact wrenches (PCW's). The resulting contact wrench is a convex combination of the PCW's.

The grasp wrench space can be calculated as

$$GWS = \text{ConvexHull}\left(\bigcup_{i=1}^{n} \{w_{i,1}, \ldots, w_{i,m}\} \right) \quad (7)$$

It must be noted that another possibility would be to determine the GWS by distributing not more than a unit force at each contact point. This would be more accurate as it takes more the finger force limitations than the cumulative force costs into consideration. In this case,

$$GWS = \text{ConvexHull}\left(\bigoplus_{i=1}^{n} \{w_{i,1}, \ldots, w_{i,m}\} \right) \quad (8)$$

where $\oplus$ denotes the Minowski sum. However, the calculation of the Minkowski sum for three contact points with eight PCW's each, means the computation of the convex hull of 8^3=512 6D vectors. This computation took a little more than 10 seconds on a Pentium 166.

The time-consuming calculation of the convex hull can be sped up applying incremental calculations (Borst, *et al.*, 1999). 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 spannig 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 center of the coordinate system.) 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 (Mishra, *et al.*, 1987): 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.

The Qhull package is used for the computation of the convex hull (Barber, *et al.*, 1996). The computation of the three contact wrenches, each represented by a different number of PCW's in R^6, can be seen on Table 1. The tests were run on a Pentium 166.

Table1. Convex hull calculation times of 6D vectors.

Number of primitive contact wrenches at one contact point	Convex hull calculation time (ms)
3	8
4	12
6	25
8	65
12	151
16	304

6. SUMMARY

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 shown, which generates many grasp candidates based on a heuristic search. A simulated annealing-style optimization method is applied to these candidates to enhance the stability of the grasp. Finally, an evaluation function, which is proved to be run in realtime, measures the stability of the grasps.

7. FUTURE WORK

So far it was assumed that the geometric object model is fully known. In real environments, this requirement often can not be satisfied. When using sensor data based object model, the model is incomplete. In the near future the generation of a partial object model from sensor (mainly visual) data is considered: computing the three-dimensional shape of an object from multiple pictures taken at known locations. First, a fast algorithm is executed for computing the bounding volume of the object

(Bendiksen, *et al.*, 1999). The bounding volume is presented as a set of planar cross-sections. As it was shown, the form closure part of the grasp planner heavily depends on the concavities of the object. Therefore in the next step the concavities of the object is searched via 'space carving'. This method determines for every voxel of a given portion of the space whether it is part of the object or not. Space carving is based on the equivalence class of 3D shapes, which reproduce the input photographs (Kutulakos and Seitz, 1998). The existence of a special member of this class, called *photo hull*, can be proven. The photo hull can be directly computed from the images.

8. ACKNOWLEDGEMENT

Support for the research of fast contact point placement is provided by the Hungarian National Research programs under grant No. FKFP 0417/1997 and OTKA T 029072.

REFERENCES

Barber, C. B., D. P. Dobkin and H. Huhdanpaa (1996). The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, **Vol. 22**, pp. 469-483.

Bendiksen, A., and G. D. Hager (1999). Fast 3D Boundary Computation from Occluding Contour Motion. In: *Proc. 1999 IEEE International Conference on Robotics & Automation*, pp. 1934-1941.

Borst, C., M. Fischer, and G. Hirzinger (1999). A Fast and Robust Grasp Planner for Arbitrary 3D Objects. In: *Proc. 1999 IEEE Int. Conference on Robotics & Automation*, pp. 1890-1896.

Ferrari, C. and J. Canny (1992). Planning Optimal Grasps. In: *Proc. 1992 IEEE Int. Conference on Robotics & Automation*, pp. 2290-2295.

Kutulakos, K. N., S. M. Seitz (1998). A Theory of Shape by Space Carving. *Technical Report TR692, Computer Science Dept., U. Rochester.*

Lantos, B. (1999). Some Possibilities to Increase the Intelligence in Robot Control Systems. In: *Proc. 1998 IEEE Conference on Intelligent Engineering Systems*, pp. 7-18.

Maekawa, H. and J. M. Hollerbach (1998). Haptic Display for Object Grasping and Manipulating in Virtual Environment. In: *Proc. 1998 IEEE Int. Conference on Robotics & Automation*, pp. 2566-2573.

Mishra, B., J. T. Schwartz and M. Sharir (1987). On the Existence and Synthesis of Multifinger Positive Grips. *Algorithmica*, **Special Issue: Robotics**, pp. 541-545.

Rimon, E. and J. Burdick (1998). Mobility of Bodies in Contact I.: A 2nd Order Mobility Index for Multiple-Finger Grasps. *IEEE Trans. on Robotics and Automation*, **Vol. 14**, pp. 696-708.

Tél, F. and E. Tóth (2000). Stereo Image Processing and Virtual Reality in an Intelligent Robot Control System. *Journal of Intelligent and Robotic Systems*, **Vol. 27**, pp. 113-134.

Tél, F. (1998). Stereo vision system for intelligent robots. In: *Proc. Symposium on Intelligent Systems in Control and Measurement.*

Yoshikawa, T. and K. Nagai (1991). Manipulating and Grasping Forces in Manipulation by Multifingered Robot Hands. *IEEE Trans. on Robotics and Automation*, **Vol. 1**, pp. 67-77.