AN EFFICIENT GRASP PLANNING SYSTEM USING IMPULSE-BASED DYNAMIC SIMULATION

Zhixing Xue, Ulrich Stadie, J. Marius Zoellner and Ruediger Dillmann

Forschungszentrum Informatik
Haid-und-Neu-Str. 10-14, 76137 Karlsruhe, Germany
e-mail: xue@fzi.de
web page: http://www.fzi.de/ids/

Keywords: Grasp Planning, Impulse-Based Dynamic Simulation

Abstract. Grasping and manipulation are the most important functions for service robots. To grasp a object with a robotic hand, the finger joint positions and the relative pose between hand and the object need to be considered. This high dimensional space is explored efficiently using an impulse-based dynamic simulation in this paper. The graspable region of the robotic hand is analyzed. The sampled Cartesian poses build a trajectory in this graspable region, along which the object moves in the impulse-based dynamic simulation by applying impulses onto it. During the whole simulation process, the fingers try to close to grasp the object. If at least three contact points are found between the hand and the object, the grasp quality is evaluated. Stable grasps are saved into a grasp database, which can be accessed during the real execution to grasp the object with the real robotic hand. We have tested the simulation process with different objects.
1 Introduction

Planning the grasps for a robotic hand with many degrees of freedom is a difficult problem because of the high dimensionality of the configuration space. Besides the relative frame between the hand and object, the finger joint positions are also to be determined. The fingers should collide with the object to be able to apply forces onto it, so that the grasp can resist external disturbances such as gravity. In this paper, we propose a novel method to plan force closure grasps using an impulse-based dynamic simulation with linear time complexity, to explore the dexterous manipulation space of a robotic hand and to quickly find the corresponding finger positions that enable the hand to grasp the object.

2 Grasp Planning System

To grasp and manipulate the objects in the household by a service robot, a grasp planning system is needed computing automatically the pose of the hand relative to the object and finger positions with which the object can be grasped firmly. The grasp quality is analyzed with some given physical criteria using the found contact points between the object surface and the fingers. Different contact types have been developed to model the forces applied by the fingers at the contact points, which can only act against the object. With a fixed reference point to the object, the torque acting onto the object by the modeled force can also be computed, which forms a 6D vector wrench together with the force. A force closure grasp [1] can apply a wrench required to resist any external disturbances and is the most studied grasp property in the literature. After the grasp is determined, the forces at the contact points can be optimized as a linear matrix inequalities problem [2]. To apply these optimized forces to the object, the necessary torques acting at the finger joints can be computed using the hand Jacobi matrix, so that the grasp can be performed with the joint torque impedance control [3].

The main difficulty of the grasp planning problem is that it deals with a high-dimensional space. Besides the internal degrees of freedom of the robotic hand, the relative position and orientation between the hand and the object are also to be considered. It can be solved in either forward or backward direction. The forward solution involves the finger forward kinematic to close the fingers and uses the collision detection technique to detect the finger joint positions at collision, such as the grasp planning simulator “GraspIt!” [4]. The backward method is object centered. Contact points are randomly [5] or analytically located on the object surface to evaluate the grasp quality. An inverse kinematic algorithm for the finger is used to find the corresponding feasible finger joint position.

3 Dynamic Simulation Systems

The dynamic simulation of multi-body systems plays an important role in the robotic field. Its task is to compute the movements of multiple rigid bodies that are linked together by specific joints to form one multi-body system under the consideration of forces that are acting on these bodies while the system is moving and interacting with other bodies. The relations between the bodies are described and governed by constraints. These constraints can be used to model joints, collisions and permanent contacts. A joint constraint reduces the degrees of freedom of a multi-body system. After the constraints are defined for the bodies in the system, a simulation method computes the corresponding internal forces that prevent the joints from breaking. Several linear time dynamics simulation methods have been developed to satisfy these constraints.
The penalty method adds a force to a multi-bodied system, if a constraint is not satisfied \[6\]. The direction and the magnitude of this correcting force depends on the constraint violation. While the penalty method is easy to implement, it is not suitable for accurate results.

The reduced-coordinate formulation method can provide a more accurate simulation. Holonomic constraints reduce the degrees of freedom of a multi-body system permanently. For a multi-body system a parameterization is required to reduce the number of coordinates that describe the system’s state to a minimum. The equations of the motion can be expressed via reduced coordinates by using a Lagrangian formulation of the problem. Algorithms based on this formulation have a time complexity of \(O(n^4)\). The first linear-time algorithms were based on a Newton-Euler formulation \[7, 8\].

The Lagrange multiplier method offers several advantages. The use of Lagrange multipliers allows to extend a model during the simulation without the need of a new parameterization. It also allows the handling of non-Holonomic constraints. Baraff \[9\] described a linear-time method for acyclic models that is based on Lagrange multipliers having the same time complexity as the reduced coordinate method of Featherstone \[7\]. The disadvantage of the Lagrange multiplier approach is that it has a drift problem due to numerical errors. To make the necessary corrections caused by this drift problem an additional stabilization method is required. One possible solution is to apply extra forces \[10\]. Another solution is the stabilization method \[11\] by adding additional terms to the constraints to compensate drifting.

An impulse-based simulation method predicts for every joint in the system its state after the next simulation step. If the predicted joint state is not valid, an impulse is estimated and applied to the joint. Before the simulation step is performed, the correction impulses are computed in an iterative process until all the predicted joint states in the system are valid. After the simulation step, the unsatisfied velocity constraints are solved by iteratively computing the impulses that cause instantaneous velocity changes. Because the impulse-based methods handle position and velocity constraints without an acceleration-based formulation \[12\], there is no numerical drift problem and no extra stabilization step is needed. Weinstein et al.\[13\] use the predictions to determine the necessary impulse by solving a non-linear equation. This equation can be efficiently solved by approximating the required velocity change. Dependencies between different joints are solved by an iterative method, which handles joint constraints, models with loops, collisions and contacts with friction \[14\]. This iterative process can also be interrupted at any time to get a preliminary result. To simulate multi-body systems with complex models, Bender et al. suggest in \[15\] another impulse-based method that allows, as a result of the linearization, to describe dependencies between joints by a system of linear equations (SLE), with a time complexity \(O(n^3)\) in the worst case. This was further improved by the same author to \(O(n)\), which allows the simulation with complex models to be in real-time. This impulse-based dynamic simulation is used in our work to plan grasps efficiently.

4 Use Impulse-Based Dynamic Simulation For Grasp Planning

In this paper, we propose a novel approach to plan force closure grasps using the impulse-based dynamic simulation. The high dimensional configuration space, consisting of finger joint positions and the relative frame between hand and object, is efficiently sampled and tested. In the impulse-based dynamic simulation \[15\], the object moves in the graspable region of the hand while the fingers are trying to close with some closing strategies. This way, the relative frame is sampled by the object movement. At each simulation step, a hand configuration corresponding to the object frame can be automatically found, where the fingers are closed and collide with the object. No explicit search in the hand configuration space is needed. All the contact points
Figure 1: (a) a service robot with two KUKA LWR Arms and Schunk Anthropomorphic Hands. (b) A found force closure grasp by the proposed method.

are detected by the impulse-based dynamic simulation. The contact points between the fingers and the object are used further to evaluate the grasp quality. If there are more than two contact points, the grasp quality is computed to check if it is a force closure grasp. By a force closure grasp, the hand frame relative to object and the finger positions are saved in a grasp database. The grasp database will be accessed in the on-line phase. A grasp which can be performed collision free and is allowed by the arm reachability can be executed by the real robot, as shown in Fig. 1(a).

Notice that no dynamic in the simulation is considered by the grasp quality computation. While the object dynamically moves further in the simulation, it is treated as a static object by the grasp planning. Only the contact points between the robotic hand and the object are used to evaluate the grasp quality, because at the time of the actual real execution, the object does not have the simulated dynamic. During the real grasping process, the object remains static and will be grasped by the robotic hand at the computed contact points.

The object movement and the finger closing are implemented by applying impulses onto the grasp object and finger links in every simulation step. The direction of the impulse onto the object points along the trajectory. Whereas the direction of the impulse onto the finger link is perpendicular to its rotational joint, which rotates the joint. The magnitudes of the impulses are limited by the tolerance of the collision checking, so that the impulses can be solved quickly in several iterations by the dynamic simulation.

4.1 Graspable Region

If the object lies in the region between the thumb and other fingers of a robotic hand, it can be well grasped and manipulated. Without explicit analysis of the dexterous manipulation space, we have manually defined a cube between the fingers of the Schunk Anthropomorphic Hand [16], where the thumb is rotated opposite to the other fingers. This graspable region is a continuous Cartesian space. To explore this region with the object’s pose, discretization is needed. The sampled object poses build the trajectory, along which the object moves in the dynamic simulation. The positions are sampled with a fixed step size along the three axes, shown as the red dots in Fig. 2(a). To sample the orientations at each of these positions, 12
Figure 2: The graspable region of the robotic hand is equally sampled. The sampled via points build the trajectory, along which the object moves in the dynamic simulation. Red dots in 2(a) represent the sampled positions. Orientations are generated using the 12 vertices of an icosahedron as the first rotational axis (the blue one).

points equally distributed on a sphere are chosen to determine the first rotational axis. This is generated by an axis from the origin to one of the vertices of a regular icosahedron, which is located at the origin. Around this axis, it is then rotated in a fixed step size. Each time, a sampled orientation is created. With $30^\circ$ as the rotation step size, each via point along the trajectory has totally 144 orientations. One of the orientations generated this way using icosahedron is presented in Fig. 2(b).

4.2 Finger Closing In The Dynamic Simulation

The finger closing in the dynamic simulation is implemented by a PID controller. Each finger joint of the robotic hand is modeled as a motor-hinge joint in the impulse-based dynamic simulation. The motor-hinge joint is implemented as a PID controller. It determines for each joint the torque that is enforced onto the rotational axis of the joint if a difference between the current angle and the desired angle of the joint is detected. The behavior of the PID controller is governed by three values. The proportional value depends directly on the difference of the desired angle and the current angle of the joint. A high value is selected to make the joint react in relation to this angular difference, causing the joint to rotate faster if the current angular difference is large. The integral value reduces overshooting the desired angle to a minimum amount. This increases the responsiveness of the joint to reach its new desired angle quickly. The last value is the derivative value which was tuned to make the joint settle quickly on its desired angle once it has been reached to minimize oscillation which could cause the finger to lose a contact to the grasped object.

Depending on the state of collision between the finger and the object, there are two cases of finger closing: finger flexing and finger extending. Finger flexing is active if the finger does not collide with the object. The maximum joint values of the finger joints are set as target position to the PID controller, to close the finger. If there are contact points between the finger and the object, finger extending is activated, which opens the finger a little from its current position. This brakes the closing process and avoids applying two big forces onto the object, which may push the object out of the hand. The fingers are only supposed to maintain in contact with the object without hindering the movement of the object. If the object is moving towards this finger, finger extending helps the object to get its target via point along the trajectory. Otherwise, if the contact points are lost due to the minor opening, the finger will close again by the finger.
flexing in the next simulation step. If the finger is fully closed but still without contact with the object, the finger is reset to its opening position to close again. By finger opening and closing, the minimum and maximum joint values are always checked to guarantee that the found grasps can be executed by the real robotic hand.

The abduct and adduct movements of the fingers are also used for finding grasps. Two adjacent fingers influence each other if at least one of them collides with the object. If both of them are in contact, the two fingers spread apart from one another to find contact points in a bigger region on the object surface, which could improve the stability of the grasp. If a finger is not in contact with the object, but its neighboring finger is, the former finger will lean towards the finger in contact to increase the chances of coming into contact with the object during the closing process. Currently, no explicit self collision between the fingers is performed. The interpenetration between adjacent fingers is avoided by the dynamic simulation.

### 4.3 Movement Along The Trajectory

After the object is grasped, it tries to move along the predefined Cartesian trajectory in the graspable region. The translational and rotational movement is performed separately. There are two possibilities for the sequencing of the positions and orientation. The initial intention was to move the object first to a new position and to rotate through all the orientations there as shown in Fig. 2(b), before it moves to the next position. But early tests revealed that, caused by the rotation of the object, a large percentage of orientations were unreachable due to obstructions. Mainly these obstructions are the fingers themselves being unable to clear the way for the object, because they are already fully opened. Therefore the trajectory was adjusted. The grasp object moves along the trajectory in a steady orientation. Once all via points with this orientation have been reached, the object is rotated to its next orientation and moves along the trajectory with different positions. During this movement, the fingers try to close constantly. If at least three contact points are found, the grasp quality is evaluated and stable grasps are saved. This way, the whole configuration space is sampled: not only the Cartesian poses between the hand and the object, but also the finger positions. The selected Cartesian poses are explicitly tested. Also the via points between them along the trajectory are tested during the movement.
Because the trajectory is defined for the hand specific graspable region, and no geometric model of the possible objects is taken into account, not all the via points on the trajectory can be reached by the object. Thus the movement of the object in the simulation can not always be completed. A reopening of the hand is needed in some situations. Small objects can not be grasped if they are at the top of the graspable region of the hand. The thumb is very important for grasping. If the object does not collide with the thumb, it is difficult to find a grasp with only the other fingers. Sometimes the object collides with the palm of the hand or the fingers at their opening position where a grasp is impossible to be found. In all of these situations, the fingers are set to their initial opening positions. All the previously simulated dynamic states are cleared. The object is placed at its next pose along the trajectory. The simulation continues further with finger closing. The whole process has following steps:

1. Set the fingers to their opening positions
2. Place the object to its next Cartesian pose along the trajectory
3. If interpenetration between the hand and the object is detected, goto step 2
4. Activate the finger closing process, which works continuously during the following simulation process
5. If no contact with the object can be found, goto step 1
6. Otherwise evaluate the grasp quality and move the object further along the trajectory by applying impulse onto the object until all poses have been tested.
7. If the movement can not be finished and reopening of the hand is needed, goto step 1
8. If all the via points on the trajectory are tested, the simulation is finished

5 Experiments

The experiment is designed to move an object in the dynamic simulation along the introduced Cartesian trajectory for the specific robotic hand. The trajectory consists 20736 poses, which is combined by 144 sampled positions in the graspable region and 144 orientation defined using a regular icosahedron. In every simulation step the number of contact points between the test object and the hand is observed and if a force closure grasp is possible, i.e. three or more contact points are found, then the grasp quality is calculated. Four test objects are chosen for the experiment, as shown in Fig. 3 and Fig. 4:

- A cylinder with a radius of 5 cm and a length of 40 cm.
- A tin that was created by laser scanning
- A sphere with a diameter of 7 cm.
- A box with the dimensions of 5 cm · 5 cm · 25 cm.

The results for each simulation run of the test objects are presented in Table 1. It shows the total simulation steps, until the object movement along the trajectory is finished. The time needed and the number of found force closure grasps are also listed in the table. The experiments were performed on a desktop computer with 64-bit Linux system. About 16000 simulation steps can be finished in a minute. Fig. 4 shows a typical grasping situation with the box during the simulation.
<table>
<thead>
<tr>
<th></th>
<th>Sphere</th>
<th>Cube</th>
<th>Cylinder</th>
<th>Tin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation steps</td>
<td>744148</td>
<td>648245</td>
<td>567421</td>
<td>612195</td>
</tr>
<tr>
<td>Time required [min]</td>
<td>45.8</td>
<td>40.2</td>
<td>34.5</td>
<td>37.4</td>
</tr>
<tr>
<td>Grasps found</td>
<td>122345</td>
<td>232286</td>
<td>121722</td>
<td>160560</td>
</tr>
</tbody>
</table>

Table 1: Experimental results with the four test objects in the dynamic simulation.

Figure 4: A sequence of the simulated grasping process, presented from left to right and top to bottom. At the start of the simulation the fingers start to close. After the box is grasped, impulses are applied onto the box to move it along the trajectory in the graspable region of the hand. The red friction cones indicate the detected contact points between the fingers and the object.
6 Conclusions And Discussion

In this paper, we have proposed a novel combination of the impulse-based dynamic simulation and the grasp planning. The graspable region of the robotic hand is analyzed and sampled as a trajectory of Cartesian poses. In the dynamic simulation, the object is moved by controlling impulses along this trajectory. The robotic hand is also modeled in the dynamic simulation. All its joints are controlled by a PID controller. Different finger closing strategies using the PID controllers have been implemented to find the grasps quickly. Finger flexing closes the fingers to the object. Finger extending maintains the contact points between the fingers and the object. The abduct and adduct movements of the fingers are used to increase the grasped region on the object surface. Due to collision or unreachability, the object sometimes can not be moved further along the trajectory. In these situations, the fingers are reopened and the simulation continues with the object starting at the next via point. The simulation finishes if the object has finished its movement along the trajectory. During the whole simulation process, if there are at least three contact points between the hand and the object, the grasp quality is evaluated. Good grasps with a high quality are saved in a grasp database and can be executed during the real execution. Experimental results show, that this method of grasp planning samples the whole graspable region of the robotic hand and finds consecutive grasp poses during the continuous dynamic simulation process. For each of these grasp poses, the finger joint values are automatically found by the dynamic simulation, so that an exhaustive search in the high dimensional hand configuration space is avoided.

The drawbacks of this method are also to be mentioned. Currently, only objects of a convex shape can be used. This is a major limitation of the used impulse-based dynamic simulation [15]. For each hand Cartesian pose relative to the object, only one hand posture is found. There is of course endless other hand postures, which form the feasible grasps. This was analyzed in a previous work [3], where a grasp with sub-optimal grasp quality can be found with a fixed hand Cartesian pose.

We are currently working on extending the impulse-based dynamic simulation to be a dynamic robot simulation. The idea is that before the real robot performs its action, it can plan the physical results of the action first. This is very important if the uncertainty of the sensor measurement is taken into account by the real execution.

ACKNOWLEDGEMENTS

This work was funded as part of the research project DESIRE by the German Federal Ministry of Education and Research (BMBF) under grant no. 01IME01E. The research leading to these results has received funding from the European Community’s Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 216239.

REFERENCES


