Proceedings of Australasian Conference on Robotics and Automation, 3-5 Dec 2012, Victoria University of Wellington, New Zealand.

Proximity Sensing and Reactive Control for Safe Manipulation

Changmook Chun, Chansu Suh and Sungchul Kang
Korea Institute of Science and Technology, Republic of Korea
changmook@kist.re.kr

Abstract

This paper presents a new proximity sensing algorithm, and a real-time reactive control for collision avoidance and preparation for safe collision for robots which have passive safety components such as variable stiffness joints or mechanical joint torque limiters. Although passive safety components make robot safe under accidental collision, they cannot work at or around specific poses of robots. In order to detect obstacles, we get depth images from a RGBD image sensor (Kinect™ for Windows®) and convert them into sets of point clouds. The points in the clouds are classified into 'robot', 'obstacle' and 'ignored'. The reactive controller, which shares its basic principle with that of potential field method, calculates virtual forces on the robot by the points identified as 'obstacle', and avoids collision between them. Simultaneously, the controller also prepares soft (safe) collision by changing the pose of the robot so that the passive safety components function effectively. We implement the algorithm on Safe-and-Speedy Arm I (SS-Arm I), and successfully demonstrate it with a task that has a constraint on the orientation of the end-effector.

1 Introduction

It is the safety in unexpected collision that autonomous robots should guarantee to operate in unpredictable dynamic environments. The safety of robots that interact with humans, has several issues that must be resolved; accurate perception of environment including the humans, reliable planning and control of the robots (or *active* safety), and fail-safe hardware and software mechanisms (or *passive* safety) that prevent or minimize harm under unexpected or uncontrollable collisions. Flacco *et al.,* propose a new approach to the problem of avoidance and safe handling of collision in human-robot interaction which are the fundamental components of safety [Flacco *et al.*, 2012]. They describe and solve the issues of active safety such as perception, planning and control, however, the aspect of passive safety is not addressed. In this paper, we propose an alternative approach to the safety problem that considers both active and passive safety components.

The importance of collision avoidance has brought a plethora of algorithms that try to find a collision-free trajectory and continuously revise it. Gaytle *et al.,* propose a planner that simultaneously finds a path and maintains for multiple tasks in a dynamic environment [Gaytle *et al.*, 2007]. The algorithm maintains a forest of rapidly exploring random trees (RRTs) [LaValle and Kuffner, 2001], by splitting, extending, and merging them in response to moving obstacles and robots. Ferguson *et al.,* propose an algorithm that is based on the D* [Stentz, 1995] and RRT [LaValle and Kuffner, 2001] algorithms in [Ferguson *et al.*, 2006]. The planner revises a path by removing invalid parts of the trees and re-extending them. Yang and Brock suggest an elastic roadmap framework that connects milestones and move them in response to moving obstacles in the workspace [Yang and Brock, 2010]. The framework has a controller that gives high priority to critical tasks such as satisfying posture constraint and avoidance of collision. There also have been planning studies that consider the velocity and acceleration of obstacles. Van den Berg *et al.,* introduce the concept of 'acceleration-velocity obstacle' to derive a formulation for 'reciprocal collision avoidance' of multiple robots [van den Berg *et al.*, 2011]. The algorithm considers both non-holonomic as well as holonomic constraints. Fiorini and Shiller propose a heuristic method of planning tree that yields a near-optimal solution that utilizes kinematics of robots [Fiorini and Shiller, 1998]. The algorithm presented in [Haddadin *et al.*, 2010] calculates a trajectory which is based on impedance controller with virtual springs and dampers in Cartesian coordinates.

When the active safety strategy fails, *e.g.,* malfunc-

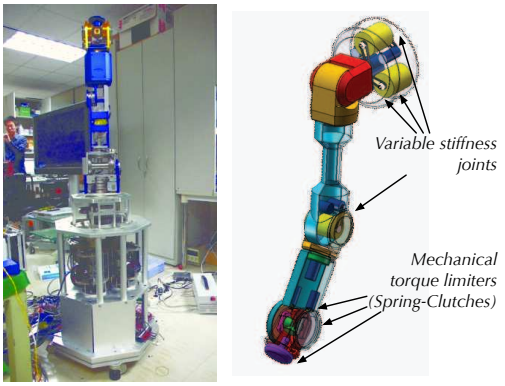

Figure 1: SS-Arm I manipulator, and its conceptual model. It has passive safety components at its joints.

tion of sensors, failure in finding a collision-free trajectory in real-time, abrupt error in the power system, the passive safety components are required. Mechanical safety devices such as variable stiffness joints (VSJ) or spring-clutches (SC) are the alternatives that may ensure the safety of humans and robots in inevitable collisions *e.g.,* [Haddadin *et al.*, 2008], [Choi *et al.*, 2011], [Kim and Song, 2010], [Lee *et al.*, 2009], and [Park and Song, 2010]. VSJ can mitigate the impact of unexpected collision with an object by adjusting the stiffness of a joint [Choi *et al.*, 2011]. Wolf *et al.,* develop a compact variable stiffness mechanism and integrate it with their DLR manipulator to ensure the safety of the robot [Wolf *et al.*, 2011]. A spring-clutch which is composed of a cam, cam-followers, push-rods and springs limits excessive torque that is transmitted through it [Lee *et al.*, 2009]. Those passive, mechanical devices absorb or limit the impact energy under the collision between robots and objects. However, the passive devices that are installed at the joints of a robot may be ineffective when the robot is at or near a singular pose so that the impact force of the collision does not transmitted to the mechanisms.

Hence, the active and passive technologies should collaborate and compensate each other; the passive components backup the active strategy in case they may fail, and the active safety technologies, especially planning and control, should consider the configurations of robots to avoid such situations in which the passive components cannot function.

In this paper, we propose (1) a new proximity sensing algorithm that uses depth images from RGBD sensors such as Kinect™ for Windows®, and (2) a reactive control algorithm to prevent possible collision against obstacles and to prepare for safe collision if it is unavoidable. The proximity sensing algorithm generates a set of point cloud from the depth image and discriminates points from robot and obstacles around it utilizing the kinematic structure and forward kinematics of the robot. Novak *et al.* propose capacitance-based proximity sensor to detect obstacles around robotic arm [Novak *et al.*, 1992], however the capacitive sensors have relatively short range of sensing and are too much sensitive to the change of electric field around them. Brock *et al.* demonstrate an obstacle avoidance for mobile manipulation using SICK laser scanner [Brock *et al.*, 2002]. The SICK sensor has been quite successful for 2D navigation, however it has limitation in 3D applications because the sensor scans only 1-dimensional proximity data. In [Kuhn and Henrich, 2007], Kuhn and Henrich propose a method of measuring distances from a 2D image of an environment. However, this method is not suitable for dynamic environments because of its accuracy issues. Recently, a method of calculating distances between robot manipulator and obstacles from depth images has been proposed in [Flacco *et al.*, 2012]. The algorithm, which is believed to be the current *state-of-the-art*, approximates robot manipulator as a series of spheres whose centers are named as control points. Then, the manipulator avoids collision by calculating the distances between the obstacles and the control points. We propose that our algorithm is more efficient than that of Flacco *et al.* because we approximate robot manipulator with only 2 cylinders. Our algorithm also considers the preparation of safe collision with the help of passive mechanical safe elements, VSJ and SC, that is not addressed by the work of Flacco *et al.*

The reactive control for safe manipulation algorithm is based on the potential field approach, *e.g.,* [Ge and Cui, 2002] and [Hwang and Ahuja, 1992]. Rapidity is the most important factor in dynamic environments where obstacles and robots are moving. The concept of potential field approaches is to define a potential function and then move to the direction of the gradient of the potential function. It is computationally efficient because it calculates only the next step of movement, not the whole trajectory. Since most sampling-based motion planning algorithms such as PRM and RRT generate a whole trajectory, they spend unnecessary time on avoiding obstacles. Therefore, we propose an algorithm that shares its basic idea with the potential field method. We do not yet consider the velocity and acceleration of obstacles to implement real-time control; a collision avoidance algorithm that utilizes the velocity and acceleration of obstacles, is introduced in [van den Berg *et al.*, 2011]. However, they consider only simple obstacles such as circles and spheres. Even though an optimal collision avoidance may be possible by incorporating the velocity and acceleration information of obstacles, it requires more computational effort. Therefore, we only consider the configurations of obstacles in this study.

The paper is organized as follows: in Section 2, we

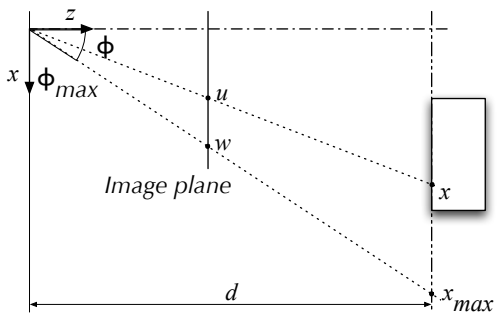

Figure 2: Projection geometry of the conversion of depth image.

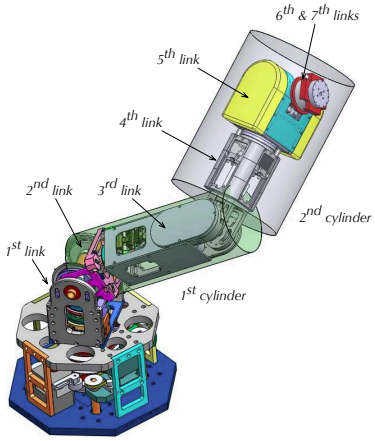

Figure 3: Approximation of SS-Arm I.

explain proximity sensing algorithm that uses a RGBD image sensor. Section 3 describes collision avoidance and preparation for safe collision algorithm and the key idea behind it. Implementation and the demonstration of the algorithm with SS-Arm I manipulator is explained in Section 4. The conclusion and future work are summarized in Section 5.

2 Proximity Sensing

Recently, RGBD sensors which give us RGB color and depth information simultaneously have been one of the most promising technologies in robotics area, especially for indoor applications such as human pose estimation [Shotton *et al.*, 2011], obstacle detection [Greuter *et al.*, 2011], and path planning for industrial robots [Csiszar *et al.*, 2012]. Kinect™ for Windows® is very popular among them, because of its robustness, reliability, and low cost. Hence, we utilize it as a depth image sensor for the proximity sensing algorithm.

2.1 Conversion of Depth Image to Point Cloud

The first step for the detection of obstacles is to convert the depth image obtained from the Kinect™ to the 3D point cloud with respect to a Cartesian coordinate system whose origin is located at the center of the Kinect™ sensor, because the horizontal and vertical coordinates of the depth image are described in pixel coordinates of the sensor.

The conversion is straightforward; we use the principle of similarity (See Figure 2) and the horizontal coordinate, for example, in Cartesian coordinate system is calculated as follows:

$$x = d\frac{u}{w}\tan\phi_{max},$$

where d is the depth (distance) between Kinect™ and the object, u is the horizontal pixel coordinate of the pixel in the depth image, w is the maximum horizontal pixel coordinate ($w = 160$ for 320×240 resolution depth image), and ϕ_{max} is the *half* of the horizontal field of view (57°) of Kinect™ [Microsoft, 2012]. Similarly, the conversion of the vertical coordinate is

$$y = d\frac{v}{h}\tan\psi_{max},$$

where v is the vertical pixel coordinate, h is the maximum vertical pixel coordinate ($h = 120$ for the image of the same resolution as above), and ψ_{max} is also the *half* of the vertical field of view (43°) of Kinect™.

2.2 Approximation of Robot Manipulator

The second step of the proximity sensing is to determine whether the points should be considered as part of the robot or that of obstacles in a pre-specified sensing volume around the robot. To make the determination easier, we *heuristically* approximate a manipulator as a series of cylinders. Suppose that we have a robot manipulator which is composed of n links, and we approximate it with p ($\leq n$) cylinders. If we want the k^{th} cylinder to represent the j^{th}, $(j+1)^{th}$, ..., $(j+q)^{th}$ links, we attach the cylinder to the j^{th} link and set the radius of it, r_k, to the minimum so that all the $(q+1)$ links lie inside the cylinder at any configuration possible. The length of the k^{th} cylinder, l_k, is the sum of all the lengths of the $j^{th}, (j+1)^{th}, \ldots, (j+q)^{th}$ links at the home pose of the robot. For example, as shown in Figure 3, we approximate the SS-Arm I manipulator with two cylinders; the first cylinder, which is attached to the 2^{nd} link, represents the 2^{nd} and 3^{rd} links, because the two links are always contained in the cylinder at any rotation of the joint between them. The radius of the cylinder is set to 150 mm. Similarly, the second cylinder, which is attached to the 4^{th} link, contains the remaining 4^{th}, ..., 7^{th} links and its radius is set to 250 mm.

2.3 Recursive Algorithm to Discriminate Points

Using the cylinder-approximation, we propose a recursive proximity sensing algorithm. The algorithm begins with transforming all the coordinates of the points from the Kinect^{TM} sensor frame to the body frame of the 1^{st} cylinder of the robot. At this stage, we mark all the points as *ignored*. The objective of the proximity sensing algorithm is to classify each point as either *ignored*, *manipulator*, or *obstacle*. Since the algorithm is recursive, we explain it assuming that we are to classify the points for the k^{th} cylinder. At this stage, all the coordinates of points are described with respect to the body frame of the $(k-1)^{th}$ cylinder. First, we examine the mark of each point. Because the points on the $j^{th}, (j+1)^{th}, \ldots, (j+q)^{th}$ links might have been marked as *obstacles* in the previous steps, it is necessary to check whether the points are *real* obstacles or not with respect to the k^{th} cylinder. Hence, if a point is marked as *ignored* or *obstacle*, we transform the coordinates of the point to the body frame of the k^{th} cylinder. Figure 4 shows an example; at the k^{th} step of sensing, part of the $(k+1)^{th}$ cylinder is considered as an obstacle to the k^{th} cylinder. However, at the next step, those points marked as *obstacle* in the previous step are changed to *manipulator*, because they are considered as part of the $(k+1)^{th}$ cylinder.

Suppose that a point is given by $p_i = (p_i.x, p_i.y, p_i.z)$. If $p_i.z < 0$ or $l_k K_{shp} < p_i.z$, we do not change the mark of the point. (The meaning of the K_{shp} will come shortly.) Otherwise, we calculate the distance, d_i, between the centerline of the cylinder and p_i. If $d_i < r_k$, we mark p_i as *manipulator*. If $r_k \leq d_i < SR$, where SR is pre-specified sensing range, we mark p_i as *obstacle*. Otherwise, p_i is marked as *ignored*. Next, we recur the procedure described above with the $(k+1)^{th}$ cylinder.

The *ad hoc* tuning parameter, K_{shp}, which stands for 'sensing height parameter' should be equal to or greater than 1.0 to cover the whole manipulator links. Figure 5 visualizes the effect of the sensing height parameter for SS-Arm I. Adjusting the sensing volume by the parameter enables the proximity sensing algorithm to cover the whole manipulator regardless of its pose. Table 1 shows the algorithm in pseudo-code.

For clarification of the description, we show the procedure with the SS-Arm I manipulator, which can be regarded as a serial connection of two bodies; upper-arm and lower-arm cylinder (See Figure 6). The homogeneous transformation matrices, T_{BK}, T_{UB}, T_{LU} are determined because we know all the joint angles and forward kinematics.

Table 2 shows the proximity sensing algorithm customized to SS-Arm I and Figure 7 shows the result of the classification for SS-Arm I.

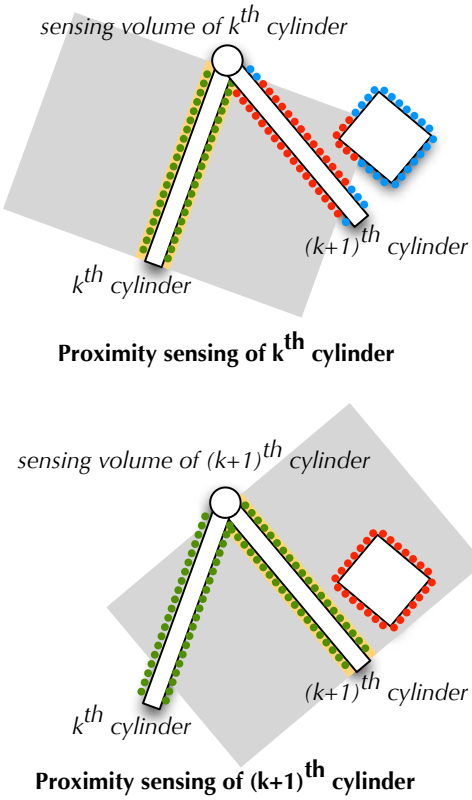

Figure 4: Recursive proximity sensing. Red, green, and blue dots represent points marked as *obstacles* ,*manipulator*, and *ignored*, respectively.

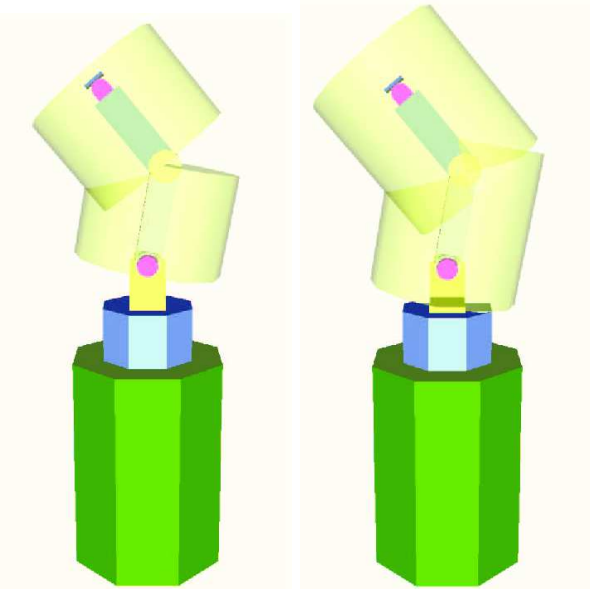

Figure 5: The effect of the sensing height parameter, K_{shp}. The figures show the sensing volume when K_{shp} is set to 1.0 and 1.5, respectively.

Table 1: Classify Points

classify_points(*points*, T, h)

```
1  for i = 0 to width do
2    for j = 1 to height do
3      if points[i][j] = 'ignored' ∨ points[i][j] = 'obstacles' then
4        transform(points[i][j], T)
5        if 0 < Points[i][j].z < h then
6          d ← distance_from_centerline(points[i][j])
7          if d < radius_of_cylinder then
8            points[i][j] ← 'manipulator'
9          else if d < sensing_range then
10           points[i][j] ← 'obstacles'
11         else
12           points[i][j] ← 'ignored'
13   end
14 end
```

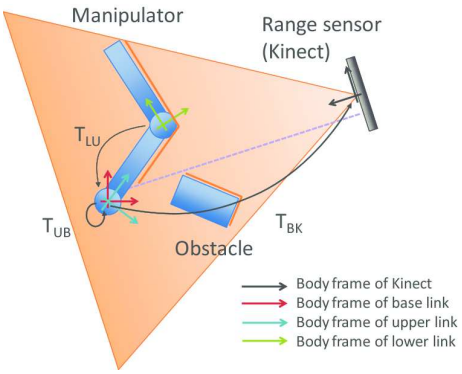

Figure 6: Configuration of Kinect™ and SS-Arm I

Table 2: Proximity Sensing for SS-Arm I

proximity_sensing(q, T_{BK}, T_{UB}, T_{LU})

1 $depth \leftarrow get\_depth\_images()$;
2 $points \leftarrow depth\_image\_to\_3d\_points(depth)$;
3 $h_{base} \leftarrow 0$
4 $h_{upper} \leftarrow height\_of\_upper\_link$
5 $h_{lower} \leftarrow height\_of\_lower\_link$
6 $\hat{h}_{upper} \leftarrow K_{shp} h_{upper}$
7 $\hat{h}_{lower} \leftarrow K_{shp} h_{lower}$
8 $points \leftarrow initialize\_point\_type$('ignored')
9 $points \leftarrow classify\_points(points, T_{BK}, h_{base})$
10 $points \leftarrow classify\_points(points, T_{UB}, \hat{h}_{upper})$
11 $points \leftarrow classify\_points(points, T_{LU}, \hat{h}_{lower})$

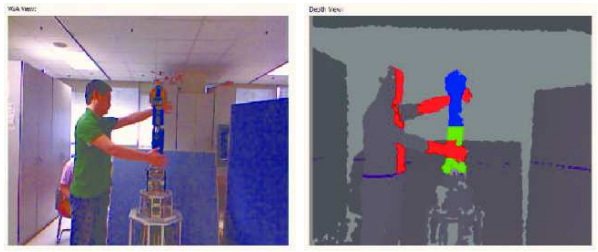

Figure 7: The result of classification: red(*obstacles*), green(*upper arm*), blue(*lower arm*), and gray(*ignored*). The strip of points colored by magenta represents the horizontal plane which is used for the alignment of the RGBD sensor.

3 Collision Avoidance and Preparation for Safe Collision

Before discussing the reactive control for safe manipulation, we briefly describe the key features and safety components of the SS-Arm I manipulator. SS-Arm I is a 7 degree-of-freedom manipulator which is similar to the structure of a human arm. It has three VSJ's for its first three joints so that it is possible to change the stiffness of them, an elbow joint, and three orthogonal joints for its wrist. The VSJ's are developed and installed as passive safety components under unexpected collision, *i.e.,* those mechanical spring components absorb impact energy before the robot detects the collision and moves back from the obstacle actively. However, there exist certain configurations of the robot and relative locations of colliding objects where the passive safety strategy cannot work, because there are only 3 VSJ's at the shoulder of the robot. Figure 8 shows such a situation. Hence, we propose a new safety strategy that considers preparation of a safe collision as well as collision avoidance. The strat-

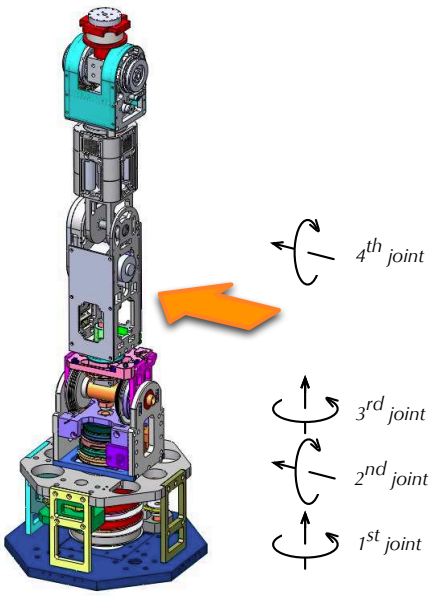

Figure 8: An example of dangerous collision! When an object collides with the manipulator in the pose shown above along the direction depicted with a tangerine arrow, the three VSJ's installed in the first three joints cannot absorb impact energy at all.

egy also enables *real-time* reactive control of the robot manipulator because of the small computational effort of the algorithm.

3.1 Repulsive and Preparatory Terms

Under our safe manipulation strategy, the controller of a robot manipulator has two operational modes; normal and collision avoidance & preparation (CA&P) mode. When there is no obstacle in the sensing volume of the manipulator, the controller is in normal mode. As the name suggests, the robot moves along a pre-defined trajectory or path normally. However, when obstacles are detected in the sensing volume, the controller changes its operational mode to CA&P mode. This section describes the reactive control in CA&P mode in detail.

Reactive control in CA&P mode shares its basic idea with that of potential field approach; obstacles around robot manipulator generate repulsive force which increases as the distance between the robot and obstacles decreases. Suppose that the robot manipulator and an obstacle are located as shown in Figure 9. Each point, p_j, on the obstacle is supposed to exert virtual repulsive forces f_1^j and f_2^j on the first and second cylinder, respectively. The gross force on the i^{th} cylinder by the obstacle is

$$f_i = \sum_j f_i^j(d_i^j), \tag{1}$$

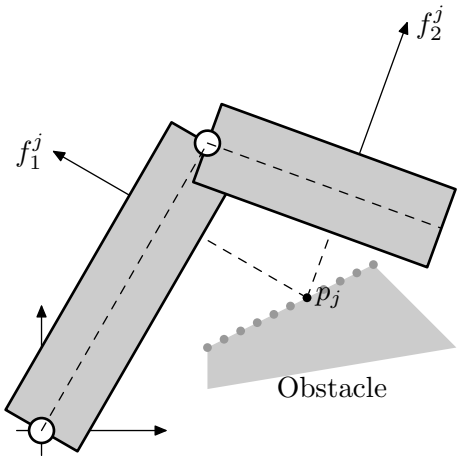

Figure 9: An example configuration of SS-Arm I manipulator and an obstacle.

where d_i^j is the distance between the point p_j and i^{th} cylinder. Each repulsive force, f_i, pushes the i^{th} link away from the obstacle by δx_i^r.

Calculation of the *repulsive* joint angles to move the two cylinders away by δx_i^r is trivial. Let $q = (q_1, q_2)^\top$ denote the joint angles of the manipulator. q_1 is the joint angles for the first cylinder to move, and q_2 is the joint angles for the second cylinder except those contained in q_1. For example, $q_1 = (\theta_1, \theta_2)^\top$ and $q_2 = (\theta_3, \theta_4)^\top$ for the SS-Arm I manipulator, where θ_i is the angle of the i^{th} joint. The Cartesian coordinate of each cylinder is x_i and

$$x_1 = F_1(q_1) \quad \text{and} \quad x_2 = F_2(q), \tag{2}$$

where F_i represents the forward kinematics for the i^{th} cylinder. Then the joint angles to move the two cylinders are

$$\delta q_1^r = J_1^+ \delta x_1 \quad \text{and} \quad \delta q^r = J_2^+ \delta x_2, \tag{3}$$

where J_i^+ denotes the pseudo-inverse of the Jacobian of F_i, and the superscript r means they are 'repulsive' terms.

The calculation of the *preparatory* term depends on the specific kinematic structure of the SS-Arm I manipulator, as shown in Figure 8. If an obstacle or obstacles are detected, the relative location of the obstacle with respect to the SS-Arm I, especially regarding the orientation of the 2^{nd} joint, matters. Suppose that there is an obstacle near the 1st or 2nd cylinder. As the angle between the axis of rotation of the 2^{nd} joint and the vector from the center of the first or second cylinder to the obstacle approaches to either 0° or 180°, the VSJ at the 2^{nd} joint becomes ineffective to the possible collision, *i.e.,* the VSJ may not absorb the impact energy. On the other hand, as the angle approaches to 90°, the VSJ becomes effective, *i.e.,* the stiffness of the colliding cylinder decreases or soft collision is possible.

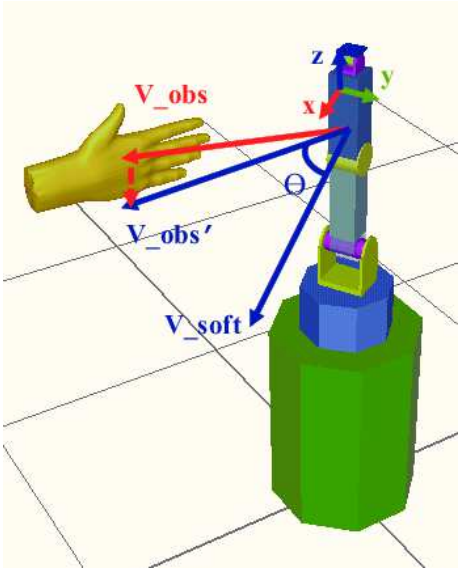

Figure 10: Geometry and symbols for the derivation of preparatory term.

Let V_{obs} be the unit vector from the 2^{nd} cylinder of the robot to an obstacle, and $V_{obs'}$ be the unit vector of the projection of the V_{obs} onto the x-y plane of the body-fixed coordinate system of the cylinder. V_{soft} is perpendicular to the axis of rotation of the VSJ. Then the objective of the preparatory term is to align the two vectors, $V_{obs'}$ and V_{soft}. The angle between the two vectors, θ is given by

$$\theta = \begin{cases} \theta' + 180°, & \text{if } -180° \le \theta' < -90°; \\ \theta', & \text{if } -90° \le \theta' < 90°; \\ \theta' - 180°, & \text{if } 90° \le \theta' < 180°, \end{cases} \quad (4)$$

where $\theta' = \arctan 2(V_{soft}, V_{obs'})$. Hence the preparatory term is

$$\delta q^p = (0, \theta, 0, 0, 0, 0, 0)^\top, \quad (5)$$

where the superscript p means 'preparatory'. (See Figure 10.)

4 Implementation and Demonstration

4.1 Implementation

To demonstrate the effectiveness of the reactive control algorithm, we implemented an experimental setup: we use a quad-core PXI controller (NI 1042Q) for the low-level motor control of the SS-Arm I and a Windows 7 PC with a 2.70 GHz Core i7 processor for the proximity sensing with a Kinect™ and the calculation of repulsive and preparatory terms (Kinematics). The PXI and PC communicate with each other using wired TCP/IP protocol. The joint angles for the SS-Arm I manipulator to move is transmitted at about every 33 msec, because the Kinect™ sensor has 30 FPS on average.

Table 3: Projection by Newton-Raphson Method

project_to_constraint_manifold(q)	
1	$e \leftarrow g(q)$
2	**while**($\|e\| < \varepsilon$)**do**
3	$q \leftarrow q - J(q)^+ e$
4	$e \leftarrow g(q)$
5	**end**

Since the reactive control algorithm transmits $\delta q = \delta q^r + \delta q^p$ for the SS-Arm I manipulator to move for 33 msec, the hardware controller (PXI) should generate smooth joint trajectories for the robot to move without noticeable discontinuities or vibration. The hardware controller generates the trajectories using a cubic polynomial for each joint, because there are 4 boundary conditions; the position, velocity, acceleration, and the corresponding value in the δq for the joint.

4.2 Demonstration: Reactive Control Under Task Constraints

In this demonstration, SS-Arm I is required to hold a glass with some beverage straight upward while avoiding collision with obstacles and preparing for possible collision. To satisfy the task constraint, we include a procedure to project an arbitrary set of joint angles onto the constraint manifold, $\mathcal{M}$. (See [Um *et al.*, 2010] and [Suh *et al.*, 2011] for details.)

If the constraint manifold is locally parameterized implicitly as $g(q) = 0$, a simple optimization procedure is to define an error vector

$$e = g(q), \quad (6)$$

so that if q lies on the constraint manifold, $e = 0$, and nonzero otherwise. For q sufficiently close to $\mathcal{M}$, $\|e\|^2$ is a valid distance function that can be easily minimized via a Newton-Raphson root-finding procedure for $g(q)$. Table 3 summarizes the projection algorithm.

Let V_{tar} and V_{cur} be vectors of the z axis of the target orientation and current orientation, respectively. Then, the task constraint of holding the cup straight upward is

$$g(q) = 1 - (V_{tar} \cdot V_{cur}). \quad (7)$$

The set of joint angles of the SS-Arm Iconsidering the collision avoidance and preparation is

$$q = q_c + \delta q = q_c + \delta q^r + \delta q^p, \quad (8)$$

where q_c is the current set of joint angles. Then, q is projected onto the task constraint manifold with the Newton-Rhapson method. Figure 11 shows that the reactive control for safety successfully operates with the task constraint satisfaction algorithm. The SS-Arm I not only avoids obstacles but also prepares safe collision while satisfying the task constraint.

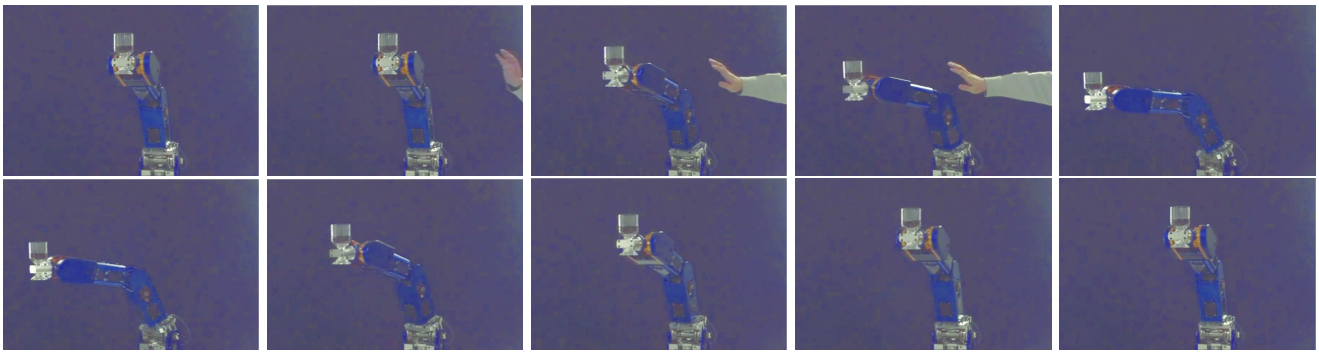

Figure 11: Obstacle avoidance and preparation with task constraints. The manipulator returns to its initial pose when the obstacle disappears.

5 Conclusion

In this paper, we introduce a new approach to safe manipulation; collision avoidance and preparation for safe collision. There exists gap between *active* and *passive* safety strategies; the former is to avoid obstacles as far as possible using various sensors and motion planning, and the latter is to limit impact and damage under unavoidable collision using passive safety components such as joint torque limiters or variable stiffness joints. Therefore, the manipulator should always prepare for safe collision by changing its pose while avoiding obstacles, because those safety components mentioned above may loose their functionality when the robot collides with obstacles in such a way that the impulse of the collision is not transmitted to the components.

We develop a new proximity sensing algorithm for serial robot manipulators using a low-cost and robust RGBD sensor, and also propose a new reactive control algorithm that fills the gap between the active and passive safety strategies by introducing preparatory movement in addition to repulsive motion. We implement the algorithm to a manipulator that has safe-joints and demonstrate that the algorithm works as we expect.

However, there are a few limitations and issues that are not addressed yet in this work:

- We have to increase the accuracy of proximity sensing. Because there exists error in the measurement of relative pose of the RGBD sensor with respect to the robot, we set the radii of cylinders to be slightly larger than their minima required. Because of the margin, the algorithm fails to detect small obstacles which are very close to the manipulator.
- There is occlusion problem. When a small object approaches to the robot on the opposite side of the RGBD sensor, the proximity sensing algorithm detects only a small number of points that are visible to the sensor. The algorithm cannot estimate the proper size of the object yet. This can be overcome by adding more RGBD sensors, or improving obstacle detection algorithm to cluster points that have the same or similar velocity vectors, regard them as a single object, and estimate the size of occluded part of the object.
- More elaborate evaluation of the collision avoidance. Because the overall performance depends on the physical specifications of robot manipulator, we are developing a framework to evaluate the collision avoidance that considers the acceleration, velocity, joint limits, and inertia of the robot, *etc.* We have a few preliminary results obtained by Monte Carlo simulations, and are analytically investigating them.
- The algorithm to calculate the preparatory term for soft (safe) collision is not general. The algorithm should be modified and tuned to every specific kinematic structure. We are working on this issue to generalize the method.
- The repulsive control can be improved by considering the velocity and acceleration of the obstacle detected. The robot should react more sensitively to obstacles approaching to it, and less sensitively to those moving away from the robot.

We are currently working on the issues mentioned above to improve the algorithm, and re-implementing the algorithm using ROS [ROS, 2012], which incorporates OpenNI and point cloud library. The new environment also has built-in calibration capability that addresses the correspondence between RGB and depth data.

Acknowledgments

This work was supported by the Intelligent Robotics Development Program, which is one of the 21^{st} Century Frontier R&D Programs funded by the Ministry of Knowledge Economy of Korea. The authors thank Dong-Eun Choi for preparing CAD illustrations in this work.

References

[Flacco *et al.*, 2012] F. Flacco, T. Kröger, A. De Luca and O. Khatib. A Depth Space Approach to Human-Robot Collision Avoidance. in *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 338–345, 2012.

[Gaytle *et al.*, 2007] R. Gaytle, K. R. Klingler and P. G. Xavier. Lazy Reconfiguration Forest(LRF)—An Approach for motion planning with multiple tasks in dynamic environments. in *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1316–1323, 2007.

[LaValle and Kuffner, 2001] S. LaValle and J. Kuffner. Randomized kinodynamic planning. *Int. Journal on Robotics Research*, 20(5):378–400, 2001.

[Ferguson *et al.*, 2006] D. Ferguson, N. Kalra and A. Stentz. Replanning with RRTs. in *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1243–1248, 2006.

[Stentz, 1995] A. Stentz. The focussed D* algorithm for real-time replanning. in *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, 1995.

[Yang and Brock, 2010] Y. Yang and O. Brock. Elastic Roadmap—Motion generation for autonomous mobile manipulation. in *Autonomous Robots*, 28(1):113–130, 2010.

[van den Berg *et al.*, 2011] V. van den Berg, J. Snape, S. Guy and D. Manocha. Reciprocal collision avoidance with acceleration-velocity obstacles. in *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 3475–3482, 2011.

[Fiorini and Shiller, 1998] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *Int. Journal on Robotics Research*, 17(7):760–772, 1998.

[Haddadin *et al.*, 2010] S. Haddadin, H. Urbanek, S. Parusel, D. Burschka , J. Robmann, A. Albu-Schaffer and G. Hirzinger. Real-time reactive motion generation based on variable attractor dynamics and shaped velocities. in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3109–3116, 2010.

[Haddadin *et al.*, 2008] S. Haddadin, A. Albu-Schaffer, A. De Luca and G. Hirzinger. Collision detection and reaction: A contribution to safe physical human-robot interaction. in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3356–3363, 2008.

[Choi *et al.*, 2011] J. Choi, S. Hong, W. Lee, S. Kang and M. Kim. A robot joint with variable stiffness using leaf springs. *IEEE Transactions on Robotics*, 27(2):229–238, 2011.

[Kim and Song, 2010] B. Kim and J. Song. Hybrid dual actuator unit: A design of a variable stiffness actuator based on an adjustable moment arm mechanism. in *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1655–1660, 2010.

[Lee *et al.*, 2009] W. Lee, J. Choi and S. Kang. SpringClutch: A safe torque limiter based on a spring and CAM mechanism with the ability to reinitialize its position. in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 5140–5145, 2009.

[Park and Song, 2010] J. Park and J. Song. Safe joint mechanism using inclined link with springs for collision safety and positioning accuracy of a robot arm. in *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 813–818, 2010.

[Wolf *et al.*, 2011] S. Wolf, O. Eiberger and G. Hirzinger. The DLR FSJ: Energy based design of a variable stiffness joint. in *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 5082–5089, 2011.

[Ge and Cui, 2002] S. S. Ge and Y. J. Cui. Dynamic motion planning for mobile robots using potential field method. *Autonomous Robots*, 13(3):207–222, 2002.

[Hwang and Ahuja, 1992] Y. K. Hwang and N. Ahuja. A potential field approach to path planning. *IEEE Transactions on Robotics and Automation*, 8(1):23–32, 1992.

[Novak *et al.*, 1992] J. L. Novak and J. T. Feddema. A capacitance-based proximity sensor for whole arm obstacle avoidance. in *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1307–1314, 1992.

[Brock *et al.*, 2002] O. Brock, O. Khatib and S. Viji. Task-consistant obstacle avoidance and motion behavior for mobile manipulation. in *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 388–393, 2002.

[Kuhn and Henrich, 2007] S. Kuhn and D. Henrich. Fast vision-based minimum distance determination between known and unknown object. in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2186–2191, 2007.

[Shotton *et al.*, 2011] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. in *Computer Vision and Pattern Recognition (CVPR)*, pages 1297–1304, 2011.

[Greuter *et al.*, 2011] M. Greuter, M. Rosenfelder, M. Blaich, and O. Bittel. Obstacle and game element

detection with the 3d-sensor kinect. in *Research and Education in Robotics - EUROBOT*, pages 130–143, 2011.

[Csiszar *et al.*, 2012] A. Csiszar, M. Drust, T. Dietz, A. Verl and C. Brisan. Dynamic and interactive path planning and collision avoidance for an industrial robot using artificial potential field based method. in *Mechatronics*, (R. Jablonski and T. Brezina, Eds.), Springer, part 4, pages 413–421, 2012.

[Microsoft, 2012] Microsoft Corporation. Develop for Kinect^{TM}, 2012. URL `http://www.microsoft.com/en-us/kinectforwindows/develop/`.

[Kavraki *et al.*, 1996] L. Kavraki, P. Svestka, J. C. Latombe and M. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[Um *et al.*, 2010] T. T. Um, C. Suh, B. Kim, H. Noh, M. Kim and F. C. Park. Tangent Space RRT: A randomized planning algorithm on constraint manifolds. in *Advances in Robot Kinematics* (J. Lenarcic and M. Stanisic, Eds.), Springer, part 4, pages 251–260, 2010.

[Suh *et al.*, 2011] C. Suh, T. T. Um, B. Kim, and F. C. Park. Tangent Space RRT with lazy projection: an efficient planning algorithm for constrained motions. in *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 4968–4973, 2011.

[ROS, 2012] Willow Garage. Robot Operating System URL `http://www.ros.org/`.