A Visibility-based Accessibility Analysis of the Grasp Points for Real-time Manipulation

Han-Young Jang[1], Hadi Moradi[2], Sukhan Lee[2], JungHyun Han[1]

[1]*Department of Computer Science and Engineering*
Korea University
Sungbuk-ku, Seoul, Korea

[2]*School of Information and Communications Engineering*
Sungkyunkwan University
Jangan-ku, Suwon, Korea

***Abstract* – This paper presents a novel approach to accessibility analysis for manipulative robotic tasks. The workspace is captured using a stereo camera, and heterogeneously modeled with the recognized plane features, recognized objects with complete solid models, and unrecognized 3D point clouds organized with a multi-resolution octree. When the service robot is requested to manipulate a recognized object, the local accessibility information for the object is retrieved from the object database. Then, accessibility analysis is done to verify the object accessibility and determine the global accessibility. The verification process utilizes the visibility query, which is accelerated by graphics hardware. The experimental results show the feasibility of real-time and behavior-oriented 3D modeling of workspace for robotic manipulative tasks, and also the performance gain of the hardware-accelerated accessibility analysis obtained using the commodity graphics card.**

Index Terms –accessibility analysis, visibility, 3D workspace modeling, robotic manipulation

I. INTRODUCTION

Accessibility analysis refers to a spatial reasoning activity that seeks to determine the directions along which a tool can access a target object. The traditional application fields include automatic inspection with coordinate measuring machines (CMMs) [1][2], tool path planning for assembly [3], sensor placement for computer vision [4], numerically controlled (NC) machining [5], etc. In recent years, the field of service robots is getting more attention and the real-time accessibility analysis for grasping and delivering objects becomes an important issue.

This paper describes a novel approach for real-time accessibility analysis for manipulative robotic tasks using hardware graphics. The paper presents local and global accessibility analysis of a given object in realistic environments.

II. RELATED WORK

Since most of the work in accessibility has been done in the inspection field, in the following we review a few related studies in this area.

Spyridi and Requicha [12] were the first to incorporate a systematic accessibility analysis for inspected features. They use a computationally intensive method to determine if a point is accessible locally first and then globally considering the entire work piece.

An accessibility analysis approach, for an infinite length probe, based on a ray tracing algorithm was proposed by Lim and Menq [13]. They determine a discrete 3-D accessibility cone which is transformed into a 2-D map in which only the orientation of the probe is expressed by two angles in a spherical coordinate system. A heuristic is used to determine the optimal probe direction for a set of inspected points.

Limeiam and ElMaraghy [11] address accessibility analysis of any point in the 3-D space using the elementary solid modeling operations: intersection, translation and scaling. This method uses CPU time to determine an accessible point, or an accessible surface using an extended version of this method.

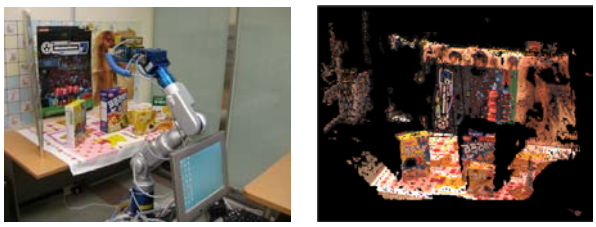
In the studies centered on inspection, it is assumed that the environment is open for a probe's motion and only the work piece itself may collide with the probe. Moreover, all these methods propose algorithms that run mostly off-line to determine the accessibility. However, in real-time manipulation, a fast accessibility of the grasp points on an object is needed. This paper proposes accessibility analysis based on *visibility test*, which has been a fundamental problem in computer graphics since the very beginning of the field. Among the visibility issues, the focus was dominantly on hidden surface removal*. The problem has been mostly solved, and the z-buffer [6] technique dominates for interactive applications. In addition to the z-buffer, current commodity graphics hardware supports an image-space *visibility query* that checks whether a primitive is visible or not. This paper reports an accessibility analysis based on the hardware-accelerated visibility query, and its application to manipulative robotic tasks.

This paper reviews 3D workspace modelling in Section III. The accessibility analysis is presented in Sections IV, V and VI. Extensions and discussions are presented in section VII. Section VIII includes the experimental results. Finally, Section IX concludes the paper.

III. 3D WORLD MODELING

Environment modeling is crucial for autonomous mobile robots, especially for intelligent service robots that perform versatile tasks in everyday human life. In this study, we rely on the real-time 3D workspace modeling for robotic manipulation using a *stereo camera* [8], mounted on the robot arm with an *eye-on-hand* configuration (Fig. 1(a)). In this section we review the 3D modeling process briefly.

* Visibility algorithms have recently regained attention because the ever increasing size of 3D datasets makes them impossible to display in real time with classical approaches. For a survey, readers are referred to [7]



(a) (b)

Fig. 1. a) An experimental environment, b) the captured range data

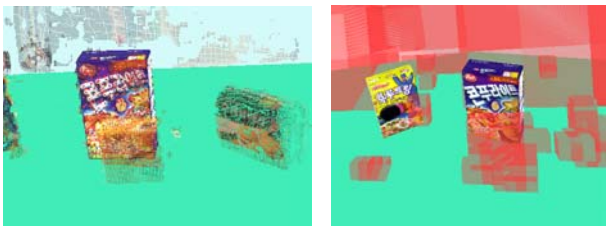
It is assumed that the workspace is textured enough so that the captured range data contains a plenty of 3D points. The range data in the form of 3D *point cloud* including 2D reference image is acquired from the stereo camera on the fly, as shown in Fig. 1(b).

In the 3D modeling process, first the global planar features are extracted. To ensure a fast and robust processing, combined use of the *SIFT* (scale-invariant feature transform) features [9] [10] with the 3D information (stereo-sis SIFT). In the well-textured environment, a meaningful bunch of the SIFT features can be extracted.

All objects to be manipulated (called target objects) have complete *solid model* representations in the object database. In addition, the database is rich enough to contain the SIFT features and *graspability* or *object accessibility* (which will be discussed in Section IV) information of each object.

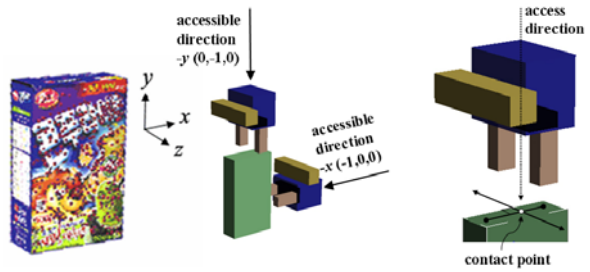
In the second step, the target objects are registered into the scene by matching the SIFT features, and then the 3D point cloud belonging to the target object is discarded, i.e. the point cloud is replaced by the solid model. Fig. 2(a) shows an example of the registered objects in which the extracted planes are also displayed.

The point clouds which are not comprised in the extracted planes and also not recognized as target objects are taken as *obstacles* which should be avoided in manipulating the target object. The obstacles are hierarchically represented in a *multi-resolution octree*. Coarse representations are given to the obstacles which are relatively far away from the target object. In contrast, fine representations are given to the closer obstacles. Such multi-resolution modeling enables efficient collision detection and motion planning. Fig. 2(b) shows the multi-resolution octree representation of the obstacles. Coarse representations lead to bigger octants (cells of the octree) behind the cereal box, and fine representations lead to smaller octants in front of the box.



(a) (b)

Fig. 2. a) Integration of the target object's solid model into the workspace, b) multi-resolution octree representation of obstacles.



(a) (b)

Fig. 3. a) Sample access directions for an object, b) a set of graspability points and an object accessibility direction.

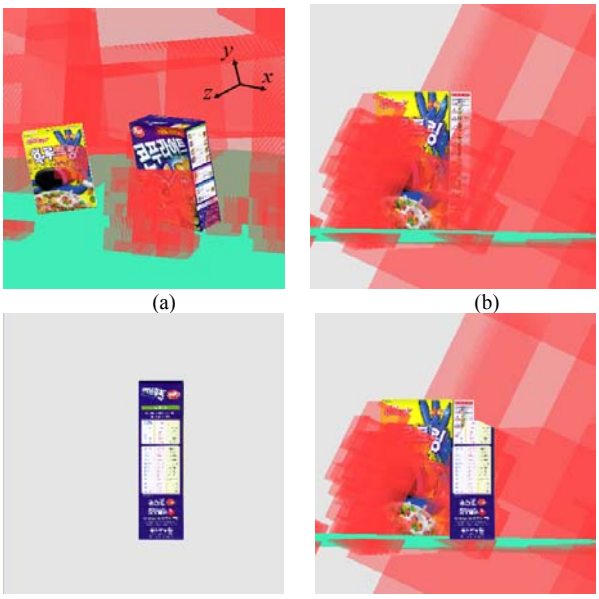
IV. GRASPABILITY

Regardless of the environment in which an object is placed, the object can be stably grasped and manipulated at certain areas of the whole object body. These graspable points can be determined based on the object's weight, dimensions and material as well as the robot's gripper characteristics. As discussed in the previous section, all target objects have complete solid models in the database, and the database contains *graspability* or *object accessibility* information of each object. Fig. 3 shows the graspability representation for a cereal box. With a robot arm of a *small* gripper, it is reasonable to define 4 access directions: $\pm x$ and $\pm y$ with respect to the local coordinates of the cereal box. Fig. 3(a) shows $-x$ and $-y$ object accessibility directions.

To grasp an object, an access direction should be determined first. For an access direction, there can be infinitely many *contact points*. As illustrated in Fig. 3(b), a contact point is defined as the intersection between the object surface and the gripper axis when the gripper moves toward the object along the access direction. Such infinitely many contact points can be represented as a line with two end points. For an arbitrarily-shaped object, however, the line should be generalized into a curve, and the curve may be 3D. Therefore, we call the collection of the contact points a *contact curve*. Object accessibility is then represented as 'access direction + contact curve.'

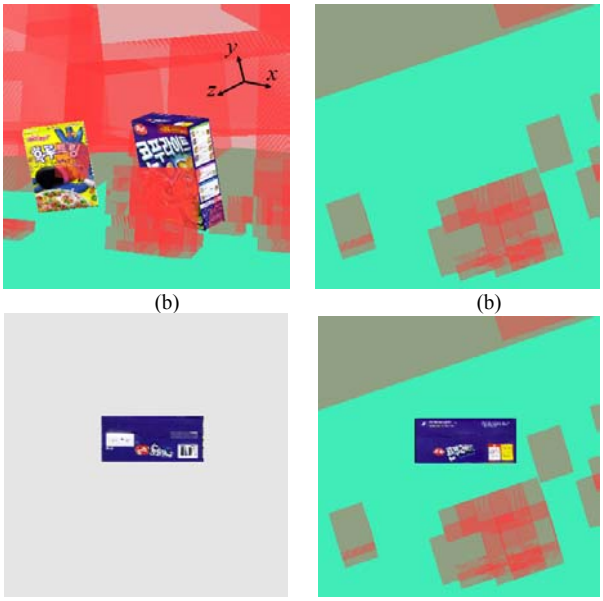
Given a contact point on the contact curve, the orientation of the gripper is determined. As illustrated in Fig. 3(b), the gripper should be aligned with the *normal* vector of the contact curve at the contact point. The access direction and the normal vector fix the initial orientation of the gripper. Together with the initial position, it defines the initial pose of the gripper. Then, the gripper simply translates towards the contact point with the fixed orientation, as will be discussed in the next section.

In the example of Fig. 3, four instances ($\pm x$ and $\pm y$) of object accessibility are stored in the database, and *priorities* are given based on the intuitive human graspability preferences. It is also possible to have a set of pre-determined priorities and change the weights on the fly considering the environment. For instance, how much the gripper jaw gets in contact with the object can be used as a quality measure for accessibility direction and can change the priority weights (Section VII-A).



(a) (b) (c) (d)

Fig. 4. Object visibility test along –x: a) The target object in the 3D model, b) the rendered environment, c) back face of the target object, d) rendered environment including the target object.



(b) (b) (c) (d)

Fig. 5. Object visibility test along –y: a) The target object in the 3D model, b) the rendered environment from top, c) top of the object, d) rendered environment including the target object from above.

V. Object Visibility

Given object graspability, it should be verified by considering the global environment. When verified, it is called *global accessibility*. As the object accessibility is encoded as 'access direction + contact curve' and priorities are given to the instances, the algorithm starts with the highest accessibility direction and then tested for global accessibility. If the test succeeds, a contact point is returned, the robot arm moves toward the contact point, and grasps the object. If the test fails, the object accessibility instance with the next-priority is selected, and the same process is repeated.

The global accessibility is verified through *visibility*, which is classified into *object visibility* and *gripper visibility*. In order for an object to be retrieved, the object should be *fully visible*. The visibility test is done using *hardware visibility query* supported by contemporary graphics hardware, which scan-converts the object and checks if the depth of any pixel is changed. The visibility query returns the number of fragments (pixels) that have passed the depth test. We have implemented the hardware visibility query using DirectX 9.0 IDirect3DQuery9 interface. When the query type D3DQUERYTYPE_OCCLUSION is given to IDirect3DQuery9, the number of pixels visible on the screen is returned. As it is done in hardware, the query is executed extremely fast.

Two types of projection are supported by graphics hardware: *orthographic* and *perspective*. We use the orthographic projection in which the viewing direction is set to one of the object's access direction starting with the access direction with the highest priority. For instance, let's assume the priority for the cereal box is set to $\pm x$ over $\pm y$, for the object in Fig. 3.

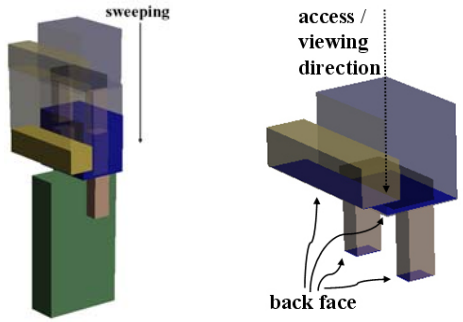
The surrounding environment of the cereal box is shown in Fig. 4(a). Let us discuss the visibility test with the access/viewing direction of $-x$.

First, the surrounding environment which include octrees, planes, and recognized objects except target object are rendered, as illustrated in Fig. 4(b). Second, the depth-test condition is set to 'greater-than' so as to make only the farther (not nearer) pixels can pass the depth-test, and then the visibility query is issued with the target object. Finally, we render the target object. If the target object passes the depth test, then the visibility query returns the number of rendered pixel. If the object is fully visible, returned number of visibility test is zero.

For the depth test with the target object, we need to consider only the back faces. We can draw the back faces of object to control the culling mode using DirectX SetRenderState interface. More specifically, we invoke SetRenderState (D3DRS_CULLMODE, D3DCULL_CW). Then, the front faces are culled, and only the back faces are rendered. Fig. 4(c) shows the back faces of the cereal box, and Fig. 4(d) is the rendered back face of the target object by using back face culling. As it shown in the figure, several pixels of target object's back faces are rendered, and so the target object is determined to be not retrievable along $-x$.

Partially invisible objects can also be grasped and moved, but it requires *motion planing*. Currently, motion planning is simplified and every object is assumed to be moved along the opposite of the access direction. In other words, we assume that the robot will grasp the object at the given point and move it along a line determined by the accessibility direction (a normal to the grasp point on the robots surface).

In Fig. 4, we have shown that the cereal box is not retrievable along $-x$. The same geometric reasoning along the access direction x shows that the box is not retrievable either. Then, the next-priority access directions, i.e. $\pm y$, are investigated. Due to the presence of the plane feature, the access direction y is immediately rejected. Fig. 5 shows the three steps of the object visibility test along $-y$.



(a) (b)

Fig. 6. Gripper visibility test using back faces

First, we render the environment. Fig. 5(b) illustrates the environment looking down along –y direction. Then the visibility test is performed using the back faces of the object as shown in Fig. 5(c). In this case, there is no pixel to pass the depth test. Therefore the cereal box is determined to be fully visible. Fig. 5(d) shows the cereal box does not collide with obstacles.

VI. Gripper Visibility

Object visibility is just the necessary condition for object manipulation. The sufficient condition is that the gripper should be able to access the object and grasp it. If the gripper can translate towards the target object *without colliding* with the other primitives in the scene to obtain the configuration in Fig. 6(a), where the gripper contacts the target object, the object is determined to be *globally accessible*.

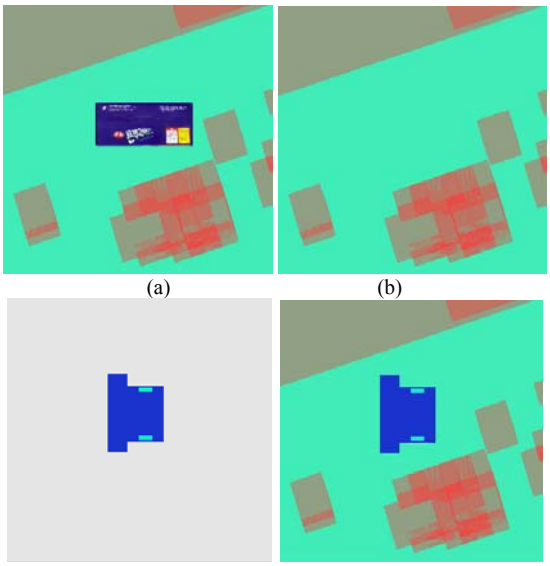
In principle, the global accessibility test requires the *swept volume* of the gripper to be tested for collision with the scene primitives. The swept volume is generated by linearly connecting the gripper of the initial pose and that of the final pose, as illustrated in Fig. 6(a). Sweeping and collision detection are not cheap operations. Fortunately, they can be replaced by *gripper visibility test*.

Given the viewing direction (access direction of the local accessibility instance) of the orthographic projection, the boundary faces of the gripper are classified into front and back faces. We need to consider only the back faces, shaded in Fig. 7(c). If the back faces (at the final pose) are all visible along the access direction, it is concluded that the sweeping gripper does not collide with the scene primitives.

The gripper's visibility test goes through the process similar to that of object visibility test. First, the surrounding environment is rendered. Second, the condition of depth test is set, and finally the back face of gripper is rendered using visibility query. The gripper is fully visible, the sweeping gripper does not collide with the scene primitives, and consequently the target object is determined to be globally accessible. Fig. 7(d) shows gripper final pose with environment. The gripper is fully visible.

Recall that there are (almost always) infinitely many contact points[†]. Note that the gripper visibility can be verified for some contact points while it may not be for others. Fig. 8 shows an example. We then have to be able to

[†] Some objects may have a single contact point. A good example is a cola can when the access direction is equal to the can axis.



(a) (b)

(c) (d)

Fig. 7. Gripper visibility test along -y

decide if the gripper-visible contact points exist, and to select the best/optimal one among them, if any. For the purpose, the contact curve is sampled.

The sampled points are given priorities. It can be either computed on the fly or pre-determined and stored in the database. Currently, it is computed on the fly: The points at the center of the contact curve are given the higher priorities. Starting from the highest-priority sampled point, the gripper visibility test is performed. If verified, the contact point is determined. Otherwise, the nest-priority sampled point is tested. If all sampled points are rejected, the object is determined to be not accessible.

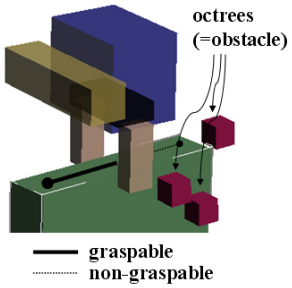


Fig. 8.Visible and invisible points

VII. Extensions and Discussions

A. *Quality of an Accessibility Direction*

Among all accessible directions that passed the visibility test, the motion planning module needs to select one. Consequently a priority or quality assigned to an accessible direction would be desirable to help selecting an accessible direction. The priorities can be either defined off-line and stored in the object database or determined on-line considering both off-line and real-time priorities/qualities.

A simple and intuitive quality measure is based on the gripper-object overlap in which the more the gripper and the object overlap, the higher the friction, the better the grasp.

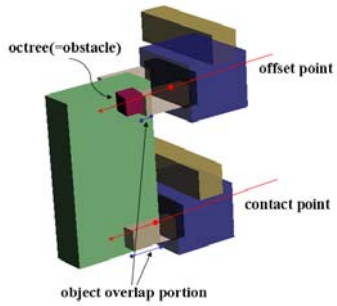


Fig. 9. Gripper-object overlap depends on the obstacles around the object.

The gripper-object overlap relies on measuring the maximum overlap between the gripper jaws and object before hitting an obstacle. Figure 9 shows two accessible directions in which one has 50% gripper-object overlap and the other has only 15% overlap.

The gripper-object overlap is also determined using the visibility test. Basically if the gripper cannot reach the contact point completely, then the visibility test is applied in fixed length considering the gripper with some offset from the contact point on the object.

B. *Local Accessibility vs. Global Accessibility*

As mentioned earlier, it is assumed that the object would be grasped and removed along the accessibility direction. In other words, there should be no object along the given direction and the robot does not have any limit manipulating the object along the accessibility direction. However, in real environment, an object may need to be grasped at the given grasped point, lifted and moved along another direction due to manipulator's limits or obstacles in the environment. This means that the local accessibility, a limited area around the target object, needs to be tested rather than the global accessibility. The limited area around the target object is represented by a sphere located at the center of the object with a given *locality radius* (Fig. 10). The locality radius can be as small as the length of the gripper plus half of the object's width. Figure 10 shows the minimum locality radius with which a gripper can slide in and grasp the target object. The red blocks show the obstacles that have been rendered and considered in the accessibility analysis. In contrast, the orange blocks are the obstacles that are outside the locality sphere and have not been considered in the accessibility analysis.

The proposed method has been modified to determine the accessibility given the locality radius of local accessibility around the target object. The modified version allows the test for both local and global accessibility by changing the locality radius.

C. *Far vs. Near Objects*

When a service robot is far away from a target object, it is not important to know all accessible directions for each contact curve. Rather it is important to know if a contact curve is accessible in one of the sample points. However, when the robot gets closer and needs to manipulate the object, then it needs to know all possible accessible directions to efficiently determine the path and perform the task. Consequently, when the robot is far away from an object, the accessibility analysis on each contact curve would return as soon as one sample point is acceptable. This shows that the contact curve under study can be accessed at least at one sample point.

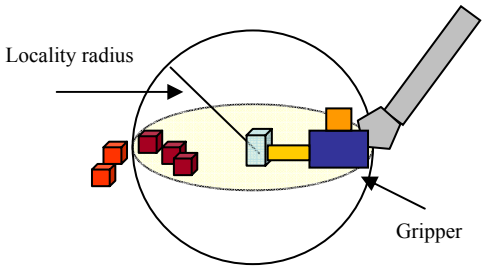


Fig. 10. Local accessibility based on the locality radius.

On the average, the accessibility analysis for faraway objects can be 50% faster than near objects, considering the above method. A heuristic, to further improve the accessibility analysis for faraway objects, is to start from top of the contact curve. It would reduce the analysis time up to 90% because the objects are normally sitting on a flat surface and they are accessible from the top.

VIII. Experimental Results

The above method was implemented considering the following scenario: a service robot is requested to grasp a pre-selected object. Then the robot captures images from the environment, makes the 3D model, recognizes the object, determines the accessibility directions, does motion planning based on a feasible accessible direction, and moves the manipulator to grasp it (Fig. 11).

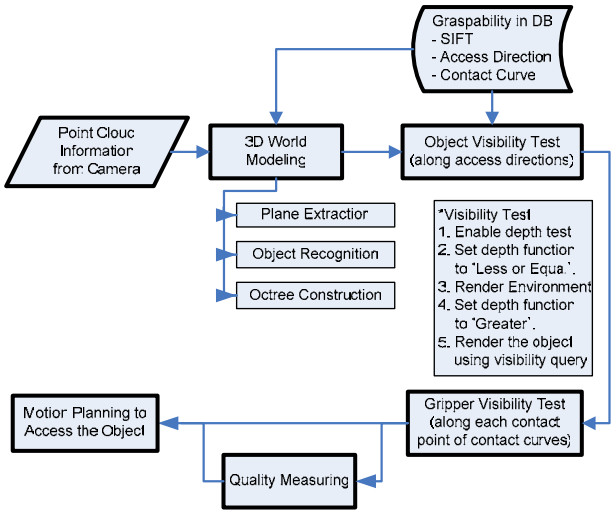


Fig. 11: Object manipulation flowchart

The algorithms were implemented on a modest PC (Pentium 4 2.8 GHz, and NVIDIA Geforce 6600GT graphics card) and many experiments have been conducted. Table I shows the measured results of system performance for a sample environment. The accessibility test has been performed on a 60x60cm^2 area with 0.05cm accuracy. It should be mentioned that the number of objects in the database does not affect the process of registering a pre-selected object. However, if all the objects in the database

are required to be registered in the environment, then the object recognition time will increase linearly based on the number of objects in the database.

TABLE I
MEASUREMENT OF SYSTEM PERFORMANCE

	Average time (ms)
SIFT feature calculation	188
Plane extraction	19
Object recognition (3 object models)	350
Multi-resolution octree construction	60
Accessibility test without quality measuring	13
Accessibility test with quality measuring	20

Currently the manipulation is done using the 3D world model which is not supposed to be updated. The experiments have shown that the smaller the object, the harder to accurately determine the pose of the object. Consequently, a continuous 3D modeling is under investigation to improve the accuracy of the 3D model progressively.

Fig. 12 shows a sample manipulation task in which the robot is requested to grasp a Sunkist container inside a refrigerator. As can be seen from the pictures, the 3D model only includes a small part of the refrigerator and does not model the whole refrigerator due to the small camera view angle.

IX. CONCLUSION

This paper presents a novel approach to accessibility analysis for manipulative robotic tasks: visibility-based geometric reasoning. The accessibility analysis process utilizes the visibility query, which is accelerated by graphics hardware. The performance and robustness of the proposed approach are evaluated in cluttered indoor environments experimentally. The experimental results demonstrated that the proposed methods are fast and robust enough to manipulate 3D objects for real-time robotic application. Currently, the arm follows a predefined path, but it is to be planned. Moreover, the minimum clearance needed for a robot gripper to access a grasp point should be determined. Therefore, future research plan includes the integration with motion planner and grasp analysis.

ACKNOWLEDGMENT

This paper was performed for the Intelligent Robotics Development Program, one of the 21st Century Frontier R&D Programs funded by the Ministry of Science and Technology of Korea. The authors like to thank Ms. Suyeon Hong and Mr. Dohyong Lee for helping with experiments. The authors also like to thanks Ms. Eunyoung Kim for her help and support in using the 3D modelling module for the accessibility analysis.

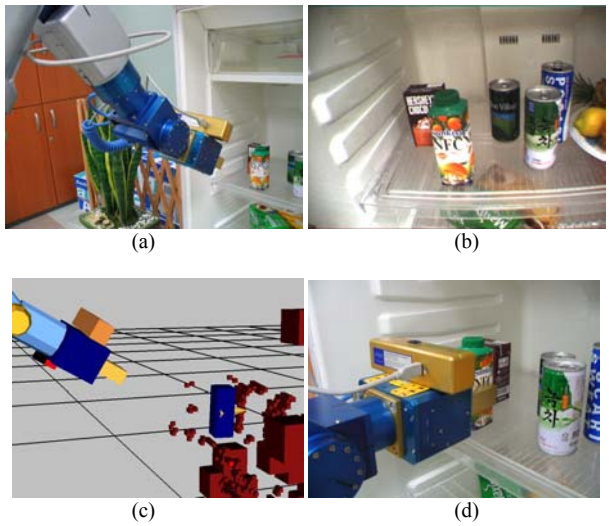


Fig. 12. a) Robot's manipulator is positioned to capture the scene, b) the scene from camera point of view, c) the 3D modeled environment represented in the simulation showing the target object in blue and the obstacles in red (octree cells), d) the robot has grasped the object after accessibility analysis.

REFERENCES

[1] ANSI, "Dimensioning and Tolerancing," Am. Nat'l Standard ANSI Y14.5M-1982, Am. Soc. Mechanical Engineers, United Eng. Center, Feb. 1982.
[2] Steven N. Spitz, Aristides A.G. Requicha, "Accessibility Analysis Using Computer Graphics Hardware," IEEE, 2000.
[3] R.H. Wilson, "Geometric Reasoning about Assembly Tools," Artificial Intelligence, vol. 98, nos. 1-2, pp. 237-279, Jan. 1998.
[4] E. Trucco, M. Umasuthan, A. Wallace, and V. Roberto, "Model-Based Planning of Optimal Sensor Placements for Inspection," IEEE Trans. Robotics and Automation, vol. 13, no. 2, pp. 182-193, Apr. 1997.
[5] P. Gupta, R. Janardan, J. Majhi, and T. Woo, "Efficient Geometric Algorithms for Workpiece Orientation in 4- and 5-Axis NC Machining," Computer-Aided Design, vol. 28, no. 8, pp. 577-587, 1996.
[6] Edwin E, "A Subdivision Algorithm for Computer Display of Curved Surfaces," Catmull, University of Utah, December 1974.
[7] D. Cohen-Or, Y. Chrysanthou, and C. T. Silva, "A survey of visibility for walkthrough applications," Proc. of EUROGRAPHICS'00, course notes, 2000.
[8] S. Lee, D. Jang, E. Kim, S. Hong, and J. Han, "Stereo vision based real-time workspace modeling for robotic manipulation," IROS 2005.
[9] S. Se, D. Lowe and J. Little, "Vision-based mapping with backward correction," 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002.
[10] D. Lowe. "Object recognition from local scale invariant features," In proceedings of the Seventh International Conference on Computer Vision (ICCV'99), pages 1150–1157, Kerkyra, Greece, September 1999
[11] A. Limaiem and H. A. ElMaraghy, "A General Method for Accessibility Analysis," International Conference on Robotics & Automation (ICRA'97), Albuquerque, New Mexico, April 1997.
[12] A.J. Spyridi and A.A.G. Requicha, "Accessibility Analysis for Polyhedral Objects", in S.G. Tzafestas, ed., Engineering Systems with Intelligence: Concepts, Tools and Applications, Dordrecht, Holland: Kluwer Academic Publishers, Inc. pp. 317-324, 1991.
[13] C.P. Lim and C.H. Menq, "CMM feature accessibility and path generation," International Journal of Production Research, vol. 32, pp. 597-618, March 1994.