A Hardware-based Accessibility Analysis for Real-time Robotic Manipulation

Han-Young Jang[1], Sukhan Lee[2], Daesik Jang[2], Eunyoung Kim[2], Hadi Moradi[2], and JungHyun Han[1,**]

[1] Department of Computer Science and Engineering, Korea University, Korea
[2] School of Information and Communications Engineering, Sungkyunkwan University, Korea

Abstract. This paper presents a new approach to real-time accessibility analysis for manipulative robotic tasks using graphics hardware. 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 multiresolution 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 local 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.

1 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.

The majority of the work in accessibility analysis has been done in the inspection field. Spyridi and Requicha [6] 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 workpiece.

An accessibility analysis approach, for an infinite length probe, based on a ray tracing algorithm was proposed by Lim and Menq [7]. They determine a discrete 3D accessibility cone which is transformed into a 2D map where 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 [8] address accessibility analysis of any point in the 3D space using elementary solid modeling operations: intersections, translation

** Corresponding Author.

and scaling. The method relies on CPU power to determine an accessible point, and an extended version of the method can be used for surface accessibility.

In the studies centered on inspection, it is assumed that the environment is open for a probe's motion and only the workpiece itself may collide with the probe. Moreover, virtually all methods propose algorithms that run mostly off-line to determine the accessibility. However, in real-time manipulation, a fast accessibility analysis 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[3]. The problem has been mostly solved, and the z-buffer [9] 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.

2 Workspace Modeling

Environment modeling is crucial for autonomous mobile robots, especially for intelligent service robots that perform versatile tasks in everyday human life. However, *real-time* workspace modeling in a *cluttered environment* is a difficult problem, and few research results have been reported.

Fig. 1. Experimental environment with an eye-on-hand configuration

The authors of this paper have tackled the problem of real-time 3D workspace modeling for robotic manipulation using a *stereo camera*, and has proposed a new approach based on recognition of global spatial features and target objects to be manipulated. In a home service robot scenario, the robot is requested to manipulate (for example, grasp and move) an object. A stereo camera is mounted

[3] 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 [10].

on the end effecter of the robot arm with an *eye-on-hand* configuration, as shown in Fig. 1.

(a) 3D point cloud (b) reference image

Fig. 2. Captured range data from a stereo camera

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. 2. (The 3D points are displayed with the combined colors of the corresponding pixels in the reference images, just for the visualization purpose.)

The approach to workspace modeling using the 3D point cloud consists of three stages: (1) plane extraction, (2) target object recognition, and (3) obstacle representation. The following subsections summarize these stages.

2.1 Plane Extraction

Fig. 3. The scene with planes extracted

In a home service robot scenario, it is reasonable to assume that the requested object lies on a planar surface such as the top plane of a table. Under the assumption, the global planar features are first extracted. To ensure a fast and robust processing, we resort to the combined use of the *SIFT*(scale-invariant feature transform) features [11][12] with the 3D information (stereo-sis SIFT).

In the well-textured environment, a meaningful bunch of the SIFT features can be extracted.

The 3D points comprised in the extracted planes are replaced by the planes in the world model. Fig. 3 shows the extracted planes and the remaining points outside the planes.

2.2 Object Recognition

(a) object model in database (b) object model in the workspace

Fig. 4. Object recognition

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 *local accessibility* (which will be discussed in Section 3) information of each object. Fig. 4-(a) shows some examples of object models used in this paper, illustrated with the SIFT features.

If the target object is recognized, its solid model is plugged into the workspace by the quaternion-based method of Horn [13], 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. 4-(b) shows an example.

2.3 Obstacle Representation

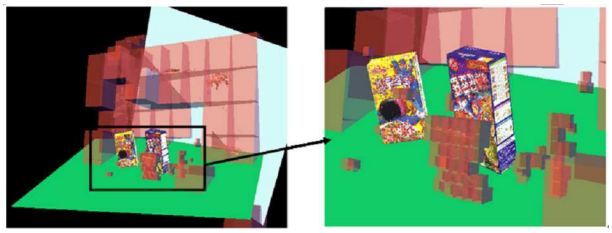


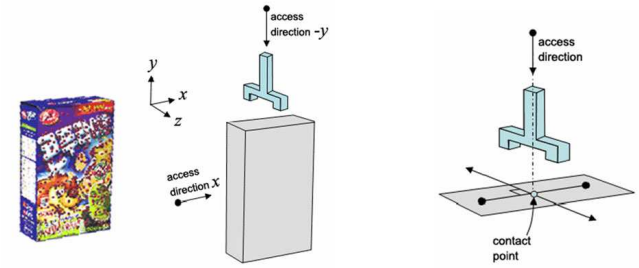
Fig. 5. Multi-resolution octree representation of obstacles

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 an *octree*. In the proposed approach, the *multi-resolution modeling* is applied to the octree construction. 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. 5 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.

3 Local Accessibility

Regardless of the environment in which an object is placed, the object might be stably grasped and manipulated at certain areas of the 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 *local accessibility* information of each object. Fig. 6 shows the local accessibility representation for a cereal box. With a robot arm of a *small* gripper (shown in Fig. 1), it is reasonable to define 4 access directions: $\pm x$ and $\pm y$ with respect to the local coordinates of the cereal box. (In Fig. 6-(a), only $+x$ and $-y$ are illustrated.)



(a) access directions (b) contact point

Fig. 6. Local accessibility

For an access direction, there can be (infinitely) many *contact points*. As illustrated in Fig. 6-(b), the 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*. Local accessibility is then represented as 'access direction plus contact curve.'

Given a contact point on the contact curve, the orientation of the gripper is determined. As illustrated in Fig. 6-(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.

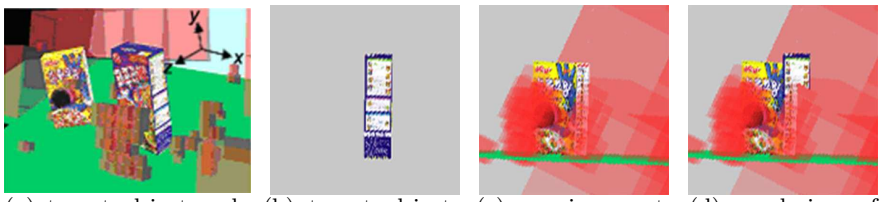
In the example of Fig. 6, four instances ($\pm x$ and $\pm y$) of local accessibility are stored in the database, and *priorities* are given based on the intuitive human graspability preferences. It is also possible to have a set of pre-determined priorities and change the priorities on the fly. Suppose that both $+x$ and $-x$ are given the first priority and that both $+y$ and $-y$ are given the second priority. Such priorities tell the robot to try either $+x$ or $-x$ first, and then try either $+y$ or $-y$ when the first try fails.

4 Global Accessibility

Given local accessibility, it should be verified by considering the global environment. When verified, it is called *global accessibility*. Geometric reasoning is required to convert the local accessibility into global accessibility.

Recall that local accessibility is encoded as 'access direction plus contact curve' and priorities are given to the instances. The algorithm starts with the highest accessibility direction and then tests it for global accessibility. If the test succeeds, a contact point is returned, which is guaranteed to be optimal for the current obstacle configuration. Then, the robot arm moves toward the contact point and grasps the object. If the test fails, however, the next-priority (local accessibility) instance is selected, and the same process is repeated.

The following subsections show that the global accessibility is verified through *visibility*, which is classified into object visibility and gripper visibility.



(a) target object and its environment (b) target object rendering (c) environment rendering (d) rendering of both target object and its environment

Fig. 7. Object visibility test along $-x$

4.1 Object Visibility

In order for an object to be grasped, 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. 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 orthographic projection, and its viewing direction is set to the access direction. Suppose the priority of $\pm x$ over $\pm y$, for the object in Fig. 6. The surrounding environment of the cereal box is shown in Fig. 7-(a). Let us discuss the visibility test with the access/viewing direction of $-x$. First, the visibility query is issued with the target object only. Obviously, the target object is fully visible, as shown in Fig. 7-(b)[4], and the number of pixels n occupied by the object is returned and recorded. Second, the depth-buffer is cleared, and the environment is rendered except the target object, as shown in Fig. 7-(c). Finally, the visibility query is issued again with the target object only, and the number of pixels m is returned. If $n > m$, the target object is partially or completely invisible. As shown in Fig. 7-(d), the cereal box is partially invisible due to some obstacles represented in octree cells. It is found by comparing n and m. As $n > m$ along the access direction $-x$, the object is determined to be *not* accessible along $-x$. Partially invisible objects can also be grasped and moved, but it requires *motion planning*. Currently, motion planning is not addressed, and every object is assumed to be moved along the opposite of the access direction. Of course, our work will be integrated with motion planning in the future.

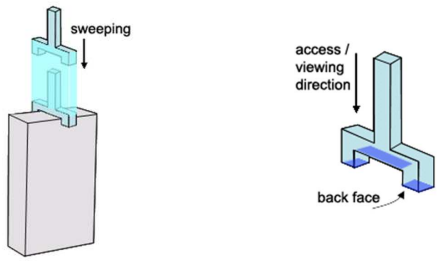
In Fig. 7, we have shown that the cereal box is not accessible along $-x$. The same geometric reasoning along the access direction x shows that the box is not accessible either. Then, the next-priority access directions, i.e. $\pm y$, are investigated. Due to the presence of the plane feature, the access direction y is immediately rejected. Finally, the geometric reasoning will prove the cereal box is accessible along $-y$.

It is worth mentioning that the first step of the object visibility test (rendering of the target object) can be performed efficiently. Note that only the *front faces* may contribute to the final image (and also depths) and the graphics hardware internally performs *back-face culling*. In the cereal box example, only the top face is used for the visibility query when testing the access direction $-y$.

4.2 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* other primitives in the scene to obtain the configuration in Fig. 8-(a), where

[4] The rendered images are provided just for easy understanding, and are not really used for geometric reasoning. Only the *hardware visibility query* is issued.



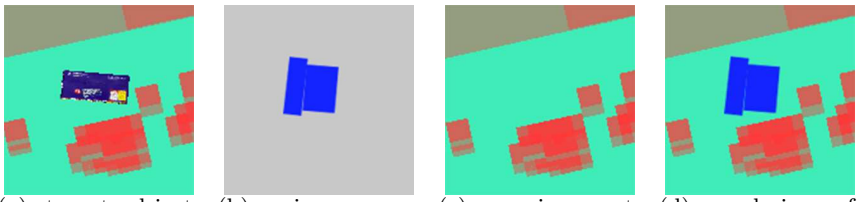
(a) sweeping gripper (b) gripper back faces

Fig. 8. Gripper visibility test with back faces

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 (discussed in Section 3) and that of the *final pose*, as illustrated in Fig. 8-(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. 8-(b). 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.

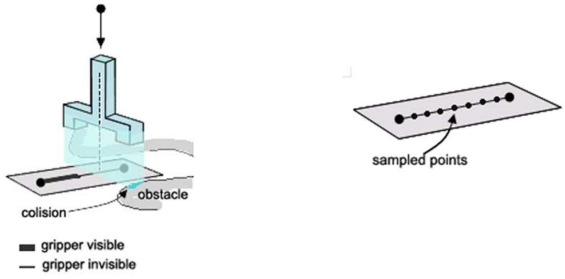


(a) target object and its environment (b) gripper rendering (c) environment rendering (d) rendering of both gripper and the environment

Fig. 9. Gripper visibility test along $-y$

The gripper visibility test goes through the process similar to that of object visibility test. First, the visibility query is issued only with the back faces of the gripper at the final pose. It is fully visible as illustrated in Fig. 9-(b)[5]. The

[5] In the current implementation, the wire part of the gripper is not considered when its geometric model is constructed. Only the gripper body and the stereo camera



(a) mixture of visible and invisible points

(b) point sampling

Fig. 10. Contact point determination

return value (number of visible pixels) n is recorded. Second, the depth-buffer is cleared, and the whole environment is rendered, as shown in Fig. 9-(c). Finally, the visibility query is issued again with the back faces of the gripper with the same pose, and the number of pixels m is returned. Only when $n = m$, the gripper is fully visible, the sweeping gripper does not collide with the scene primitives, and the target object is determined to be globally accessible. Fig. 9-(d) shows the final image.

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. 10-(a) shows an example. We then have to be able to 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, as shown in Fig. 10-(b).

4.3 Local Visibility vs. Global Visibility

As mentioned earlier, it was assumed that the object would be accessed and removed along an accessibility direction. However, in a real environment, an object may need to be accessed along a direction, grasped at a contact point, and then lifted and moved along another direction due to manipulator limits or obstacles in the environment. This means that a local visibility does not necessarily have to be converted into a global visibility. The proposed method based on the graphics hardware shows one of the simplest manipulation method, which does not actually need motion planning.

5 Conclusion

This paper presents a novel approach to accessibility analysis for manipulative robotic tasks: visibility-based geometric reasoning. The accessibility anal-

---

are modeled. The wire part is on the opposite side of the gripper tips, and therefore does rarely cause problems. However, for complete analysis, it will be modeled, but the proposed algorithm will not need to be changed at all.

ysis 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. The determined accessibility direction is used by the motion planning system to generate a collision free path to grasp and lift the object. 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 planners and grasp analysis.

Acknowledgements

This paper was performed for the Intelligent Robotics Development program, one of the 21st century frontier R&D programs funded by the Ministry of Commerce, Industry and Energy of Korea. This work is partly supported by the science and technology program of Gyeonggi Province in Korea.

References

[1] ANSI, "Dimensioning and Tolerancing," *ANSI Y14.5M-1982*, American Society of Mechanical Engineers, February 1982.

[2] S. N. Spitz, and A. A. G. Requicha, "Accessibility Analysis using Computer Graphics Hardware," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 6, No. 3, pp. 208-219, July-September 2000.

[3] R. H. Wilson, "Geometric Reasoning about Assembly Tools," *Artificial Intelligence*, Vol. 98, Nos. 1-2, pp. 237-279, January 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, April 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] 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.

[7] 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.

[8] A. Limaiem and H. A. ElMaraghy, "A General Method for Accessibility Analysis," *Proc. International Conference on Robotics & Automation* (*ICRA97*), Albuquerque, New Mexico, April 1997.

[9] E. Catmull, *A Subdivision Algorithm for Computer Display of Curved Surfaces*, Ph.D. Thesis, University of Utah, December 1974.

[10] D. Cohen-Or, Y. Chrysanthou, and C. Silva, "A Survey of Visibility for Walkthrough Applications," *EUROGRAPHICS00 Course Notes*, 2000.

[11] S. Se, D. Lowe and J. Little, "Vision-based Mapping with Backward Correction," *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.

[12] M. A. Garcia and A. Solana, "3D Simultaneous Localization and Modeling from Stereo Vision," *Proc. International Conference on Robotics & Automation* (*ICRA04*), New Orleans, LA, April 2004.

[13] P. Besl and N. McKay, "A Method for Registration of 3D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14, pp. 239-256, 1992.