Interaction in Augmented Reality by Means of Z-buffer Based Collision Detection

Yasuyuki Souma, Hidemi Yamachi, Yasuhiro Tsujimura, Yasushi Kambayashi
Department of Computer and Information Engineering,
Nippon Institute of Technology 4-1 gakuendai,
Miyashiro-cho, Minamisaitama-gun, Saitama 345-8501 Japan
c1065309@cstu.nit.ac.jp, yamachi@nit.ac.jp, tujimr@nit.ac.jp, yasushi@nit.ac.jp

***Abstract*— We propose a new interaction method in AR (augmented reality) using a depth sensor and a collision detection method with the Z-buffer. The method generates 3D models of the real world in real-time and performs interaction between the real and virtual world. In order to provide the real-time collision detection between the models of real world and those of virtual world, we have developed a detection method called *Cyber Radar*. This technique uses two sets of depth values of the Z-buffer that is generated through the orthographic projection. One is obtained through rendering the virtual space from the sensor object toward a target point. This set does not have the depth values of the sensor object. Another one is obtained through rendering only the sensor object in the reverse direction. From these two depth value sets, we obtain the distance between the sensor object and others for each pixel. This technique requires only one or two rendering processes and it is independent from the complexity of object's shape, deformation or motion. In this paper, we evaluate the interaction using this method and report the problems to be solved.**

Keywords-Cyber Radar; depth sensor; Z-buffer; collision detection; object detection

I. INTRODUCTION

For detecting and tracking objects in augmented reality, people use techniques that are based on the recognition of markers or patterns [1]. The virtual objects make action in the relation of those markers. When some other objects hide the markers, the virtual objects can no longer exist in the virtual world [2]. It is impossible for those objects to move around in the virtual and interact with any object in the real world without markers. In addition to that, since the virtual objects are projected on the real world scene, they are never overridden by real objects. These problems of AR techniques give us unnatural feelings when we see the augmented reality scenes.

Hence we need to generate 3D models of the real world to interact virtual objects with real objects in real-time. There are some techniques for generating 3D models of real world. Recently Microsoft released a motion sensing input device named Kinect [3] for their video game machine. Kinect features an RGB camera and a depth sensor. It provides us the depth values of the scene in real-time. Using these depth values, we can construct the 3D models of the real world. Since the models of the real world are just for computing interaction, not for displaying the objects in the real world, it is not necessary to generate exact 3D modes with expensive calculation cost.

In order to generate the interaction between the virtual objects and the 3D models of real world, we need to adopt a collision detection method that does not depend on the number of objects or shapes and motions of them.

The problem of collision detection in cyberspace has been intensively studied in various fields. Most of them have the same feature. They calculate geometrical intersections between each object pair. For reducing the calculation cost, they divide the space or objects into convex polygons and eliminate obviously not colliding parts. The first type of techniques divides whole space into cells and keeps positions of objects using hierarchical trees, such as octree [4], BSPtree [5] and *k*-d trees [6]. The second type of techniques uses bounding volume hierarchies (BVH) which consist from convex hulls such as spheres or boxes. Convex hulls hierarchically cover an object from the whole to each part. AABB [7], Bounding Sphere [8], OBB [9], *k*-DOPs [10] are well known and used for collision detection. Because those techniques detect collisions only one pair of objects at once, they have $O(n^2)$ computational complexity for n objects. They work very well for rigid models that never change their shapes. Once the models change their shapes, however, they have to reconstruct the tree or BVH. The reconstruction procedure takes considerable time and space computational complexities. All of those methods perform one collision test for every pair of objects. If there are several potentially colliding object pairs, they have to perform the collision tests for all of them.

Main contribution: We propose an application of the interactive AR system using a depth sensor and the collision detection method, named *Cyber Radar* [11]. In [12], we showed *Cyber Radar* with the latest hardware showed $O(n)$ performance for n polygons and reading back Z-buffer values from GPU memory is not the bottle neck for our method. *Cyber Radar* detects collisions between each predefined object and other objects. We call the object *sensor object*. *Cyber Radar* uses two sets of Z-buffer values obtained through rendering with the orthographic projection. The viewing volume for the rendering is defined to include the space in which collisions could occur. We also call the viewing volume *target volume*. At first, we set the view point at the sensor object and render the scene toward the end of the viewing volume without the sensor object. If there is no depth value in the Z-buffer less than the maximum depth value, no other object in the target volume and no

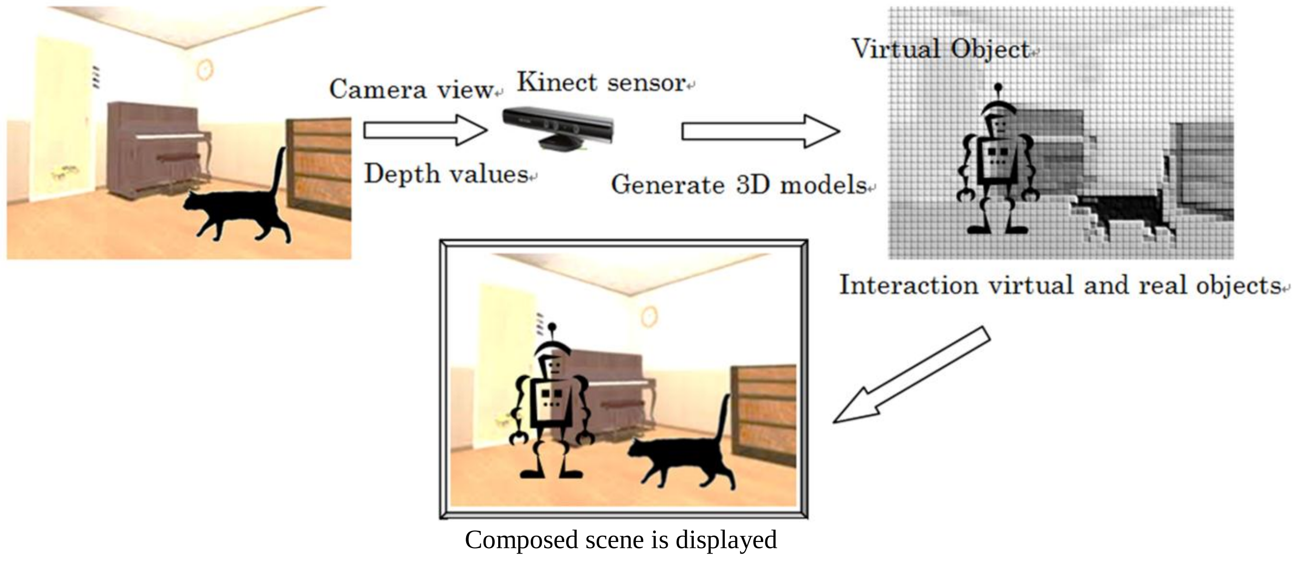


Figure 1. Structure and procedure of the AR system

collision will occur. Otherwise, we set the view point at the opposite position and render only the sensor object, without any other objects. From those two sets of depth values, we can find the distance between the sensor object and other objects in the target volume.

Cyber Radar takes only two renderings for one collision test. Different from other methods, it detects multiple collisions at once without selecting any object pair. Then *Cyber Radar* performs collision detection almost $O(n)$ time, where n is the number of polygons. *Cyber Radar* is also independent from the complexity of an object's shape, deformation or motion. In addition to these advantages, the implementation is simple and easy.

Because of these features of *Cyber Radar*, it is natural for us to apply it for detecting collisions in the AR system with the depth sensor. We have implemented our AR system with *Cyber Radar* and evaluate its effectiveness and investigate problems to be solved.

Organization: The rest of this paper is organized as follows. In the next section, we describe the structure of our AR system. In section III, the algorithm of *Cyber Radar* is described. We show the efficiency and problems of our method through experiments in section IV. We conclude our discussions in section V.

II. System Structure

Our AR system uses Kinect sensor to obtain the RGB camera view and the depth values of the view. Then the system generates the 3D models of the view. After calculating the interaction between the real and the virtual objects, the camera view and the virtual image are superimposed on the PC display. The overall structure and its procedures are depicted in Figure 1.

Aiming for realizing our AR system with real-time interaction, we need to generate 3D models of the real world without intensive computation. Then our AR system generates models of the real world by arranging polygons for each pixel at each corresponding coordinate in the cyberspace.

III. Mechanism of *Cyber Radar*

In order to detect other objects in the target volume, we set the view point at the sensor object (Figure 2(a)). By defining the orthogonal view volume that has the same depth as the length of the sensor object and the distance to detect collision, as the target volume, the scene is rendered without the sensor object (Figure 2(b)). We call the obtained the Z-buffer value set *Range Distance Scope* denoted as **R**. If there are any pixels in the *Range Distance Scope* that have the depth values less than the target volume distance ***d***, there are some objects in the target volume.

If some objects are detected in the target volume, the collision detection is performed. The view point is set at the other side of the target volume with the direction toward the sensor object. Then only the sensor object is rendered to obtain the depth value set. We named the Z-buffer value set *Mask Distance Scope* denoted as **M**. With these two depth value sets, we can obtain the distances between the sensor object and other objects. We named this distance value set *LookAt Distance Scope* denoted as **L**.

Before we describe the detail algorithm of *Cyber Radar*, we define the following notations.

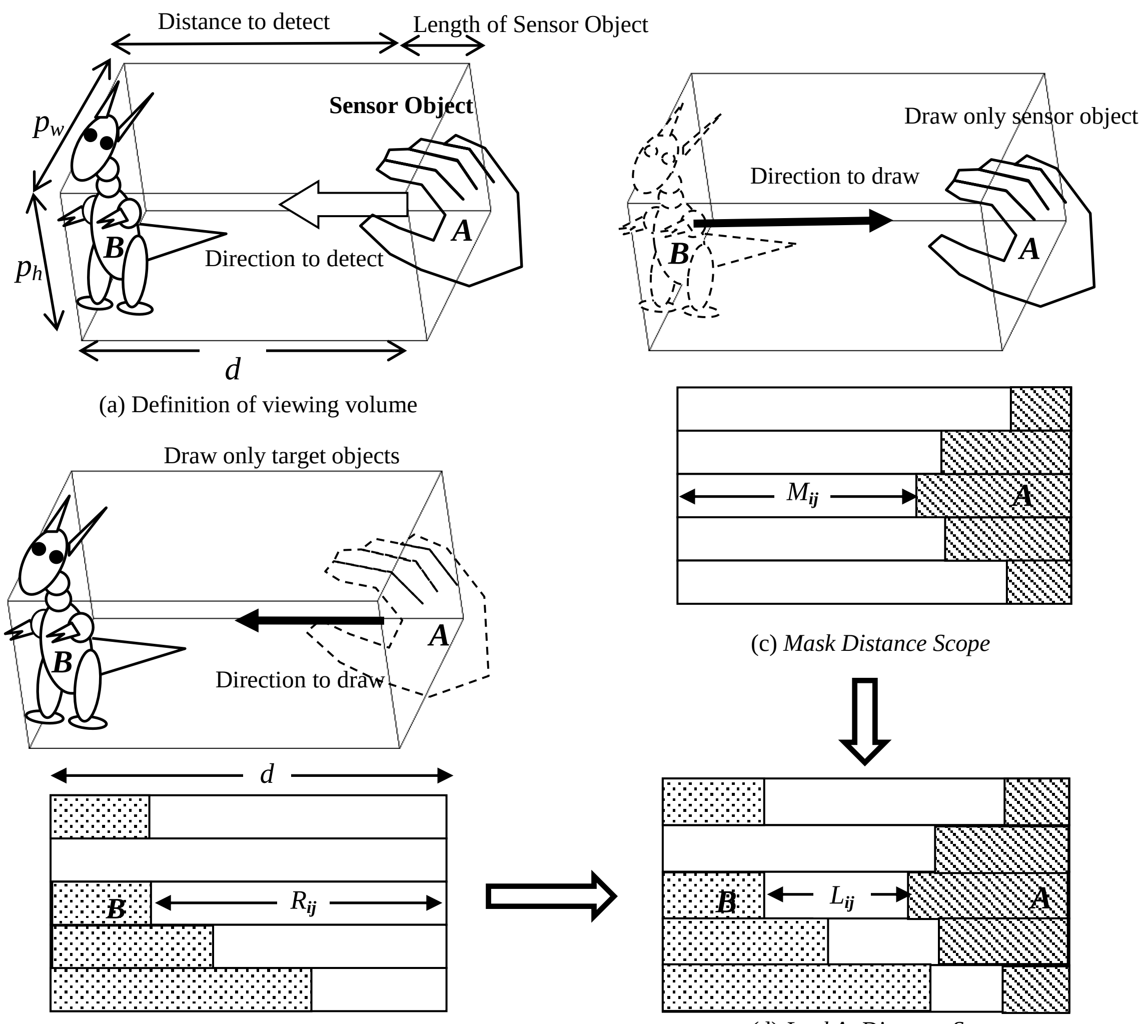
p_w, p_h : Pixel width and height of depth values $i = 1,2,\dots, p_w$, $j = 1, 2, \dots, p_h$

R : *Range Distance Scope* $\mathbf{R} \in \{R_{ij}\}$

M: *Mask Distance Scope* $\mathbf{M} \in \{M_{ij}\}$

L : *LookAt Distance Scope* $\mathbf{L} \in \{L_{ij}\}$

d : Target volume distance



(a) Definition of viewing volume

(b) *Range Distance Scope*

(c) *Mask Distance Scope*

(d) *LookAt Distance Scope*

Figure 2. Collision detection by *Cyber Radar*

The distance between view point and the detected objects is $d – R_{ij}$ and the distance from the end of the target volume to the sensor object is $d – M_{ij}$. Using these values, the pixel value of *LookAt Distance Scope* becomes as follows.

$$L_{ij} = d – (d – R_{ij}) – (d – M_{ij}) = R_{ij} + M_{ij} – d \qquad (3)$$

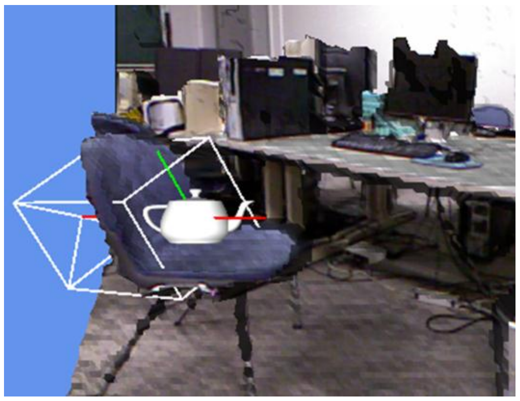
If *min*(L), the minimum value of L, is greater than zero or distance to move, collision will not occur. Otherwise collision will occur at the pixel(s) that have the minimum value. If objects are moving, the time of collision is calculated from *min*(L) and the velocity of objects.

The algorithm of *Cyber Radar* is described as follows.

```
1. Set up the rendering conditions (Figure. 2(a))
      Set view point at the sensor object
      Determine rendering direction
      Set width and height of the target volume
      $d$ := Target volume distance
      $p_w, p_h$ := Projection width and height
2. Render scene without the sensor object (Figure. 2(b))
3. Obtain R (Range Distance Scope)
4. if min(R) = $d$
         return false (no object)
5. Set the rendering conditions (Figure 2(c))
      Set view point at the end of the target volume
      Determine rendering direction toward sensor object
6. Render scene only sensor object
7. Obtain M (Mask Distance Scope)
      Swap right and left pixel values
8. Calculate L (LookAt Distance Scope)
      L := R + M – $d$
9. if min(L) > 0 or distance to move
      return false (no collision)
   else
      return true (collision is detected)
```

IV. IMPLEMENTATION RESULTS AND ESTIMATION

We have implemented the AR system to estimate its performance. Figure 3(a) shows the scene that the real world with the virtual object. The teapot is the virtual object. Figure


(a) Scene of real world and virtual tea pot

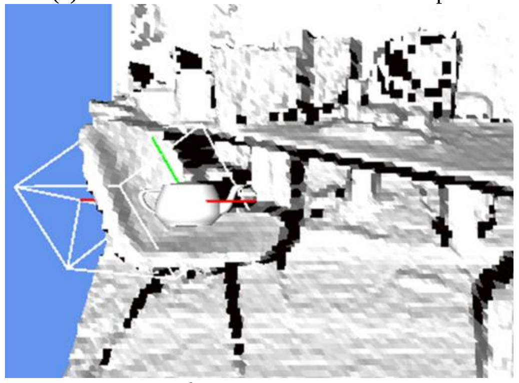

(b) 3D models of the real world and virtual teapot

Figure 3. The scene of the AR system

3(b) shows the 3D models of the same scene that is generated to detect collision between the real objects and the virtual teapot. The white rectangular frame at the teapot is the target volume of *Cyber Radar*. The resolution of the depth sensor of Kinect is 640x480 pixels. In order to generate models of the real world, we put polygons at each coordinate of pixels by considering their real sizes and depth values. The number of polygons is 37,842 for every scene of the models of the real world while 6, 6,256 and 181,412 are for three sensor objects. The computing environment for the experiment is Mac Pro with Quad-Core Intel Xeon 3500 2.66GHz and NVIDIA GeForce GT 120.

The frame rates with collision detection by *Cyber Radar* are in Table 1. As the number of polygons of the sensor objects increases, the frame rate decreases noticeably. According to our ongoing experiments, it seems that the modeling procedure causes the low frame rate. We need to analyze the bottle neck of the system exactly and devise the solution for increasing the frame rate.

V. CONCLUSION

We proposed a new interaction method in AR using a depth sensor and a collision detection method using Z-buffer named *Cyber Radar*. *Cyber Radar* detects collision independent from the complexity of object's shape, deformation or motion. Through the experiment, our AR system demonstrates that it achieves the required performance for real-time interaction. Through the experiment, we found some problems to be solved.

Table 1. Frame rate with collision detection

Number of polygons of sensor object	fps
6	13.9
6,256	12.5
181,412	9.4

There may be hidden surfaces in each scene of the real world. There are no polygons at those parts. If the virtual object moves into those parts, it falls into behind the real world. To avoid this problem, we may need to add polygons at the edges of those parts toward the polygons of their neighbor pixels. This means that closer objects in the real world are embossed.

To solve this problem, we need to develop the method for modeling of the real world. While static objects in the real world can be modeled by scanning with the depth sensor in advance, we need to generate models of dynamic objects in real-time.

We expect to reduce the number of polygons of the virtual objects for rendering according to the resolution of the depth sensor. We also expect that this method reduces the computation time for collision detection.

ACKNOWLEDGMENT

This work is partially supported by Japan Society for Promotion of Science (JSPS), with the basic research program (C) (No. 23510178), Grant-in-Aid for Scientific Research.

REFERENCES

[1] G.Klein and D.Murray, "Parallel Tracking and Mapping on a camera phone," ISMAR 2009. Proceedings of 8th IEEE International Symposium on Mixed and Augmented Reality, pp. 83-86 2009

[2] H.Uchiyama and H.Saito, "Random dot markers," IEEE Virtual Reality Conference 2011, pp. 35-38 2011

[3] http://research.microsoft.com/en-us/um/redmond/projects/kinectsdk/ <retrieved: September, 2011>

[4] H. Samet, "Spatial Data Structures: Quadtree, Octrees and Other Hierarchical Methods," Addison Wesley 1989

[5] B. Naylor, J.A. Amatodes and W. Thibault, "Merging BSP trees yields polyhedral set operations, " Computer Graphics (SIGGRAPH '90 Proc.), vol. 24, pp. 115-124 1990

[6] M. Held, J.T. Klosowski and J.S.B. Mitchell, "Evaluation of collision detection methods for virtual reality flythroughs," Proc. 7th Canada Conf. Comput. Geom., pp. 205-210 1995

[7] M. Ponamgi, D. Manocha and M. Lin, "Incremental algorithms for collision detection between solid models," IEEE Transactions on Visualization and Computer Graphics, vol.3(1), pp. 51-67, 1997.

[8] P.M. Hubbard, "Approximating Polyhedra with Spheres for Time-Critical Collision Detection", ACM Transactions on Graphics, vol. 15(3), pp. 179-210, 1996.

[9] S. Gottschalk, M. Lin and D. Manocha, "OBBTree: A Hierarchical Structure for Rapid Interference Detection", Proc. of ACM Siggraph '96, pp. 171-180 1996

[10] G. van den Bergen, "Efficient Collision Detection of Complex Deformable Models using AABB Trees." Journal of Graphics Tools, vol.2(4), pp. 1-14 1997

[11] H. Yamachi and Y. Shindo, "A Technique for Object and Collision Detection by Z-buffer," Transactions of Information Processing Society of Japan vol. 43(6), pp. 1899-1909, 2002 (in japanese)

[12] H.Yamachi, Y.Souma, Y.Kambayashi, Y.Tsujimura and T.Iida, "Evaluation of a Technique for Collision and Object Detection with the Z-buffer in Cyber Space," Cyberworlds 2011, pp. 26-92 2011