Online Collision and Occlusion Detection Based on 3D Cuboid Maps for a Daily Assistive Robot with a Tilting LRF

Kimitoshi Yamazaki and Masayuki Inaba
Grad. School of Info. Science and Tech.
The University of Tokyo
Tokyo, Japan
{yamazaki,inaba}@jsk.t.u-tokyo.ac.jp

Takemitsu Mori and Takashi Yamamoto
Toyota motor corporation,
543 Kirigabora, Nishi-hirose,
Toyota, Aichi, Japan
{takemitsu_mori, tyamamoto}@mail.toyota.co.jp

***Abstract*— This paper describes about online collision and occlusion detection for a daily assistive robot. It is assumed that 3D map about around environment is constructed by collecting datasets gradually measured using a Laser RangeFinder mounted on a tilting platform, we develop an approach to detect collision and occlusion based on the 3D map while a robot tries to manipulate objects in daily environment. As the map representation, Time-series Composite Cuboid Map is proposed. It copes with temporal sequence explicitly, and map updating is efficiently performed. Several experiments on the assumption of daily assistance by a robot, we show the effectiveness of our approach.**

Index Terms **– TCCM, collision and occlusion detection, daily assistive robot**

I. Introduction

Robots working on daily scene should have many of abilities related to environment recognition. For instance, when we consider about robotic object manipulation such as picking up an object placed on a table, not only to estimate the pose of the target object but also to figure out the condition of around environment should be needed because furniture and movable objects as people may exist near the robot. The purpose of this research is to develop a function to detect collision and occlusion of moving robot arms by using a Laser RangeFinder(LRF) mounting on tilting platform. In our approach 3D point cloud captured from the LRF is translated into light-weight map representation, collision and occlusion are able to be checked in online.

In the present, there are few sensors which provide robots with 3D information over wide-range with sufficient accuracy in a single measurement. Because sensors as LRF can only have a limited measurement range, they are often combined with movable platform to extend its measurement range. Considering about guarantee of robotic motion in online, this fact tells us the importance of handling of data with temporal sequence. In addition, effective data compression without loss of expression power is suitable for real-time application.

From these reasons, we propose a novel map representation which explicitly considers temporal alternation of measurement data. The method named TCCM (Time-series Composite Cuboid Map) is inspired from MLSM (Multi-Level Surface Map) proposed by Triebel et al. [10]. We extent MLSM for complying with issues described above and refer practical usages of the representation, especially about online collision and occlusion detection. The proposed method was examined using a tilting platform shown in Fig.1,which is mounted on the head of a dual-arm robot.

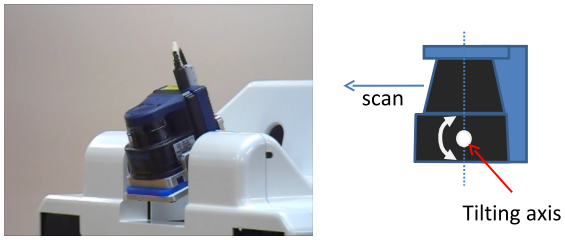

Fig. 1. A tilting LRF mounted on a robotic head

This paper is organized as follows: section II shows related work and contribution of our approach. Section III explains the map representation named TCCM (Time-series Composite Cuboid Map), and section IV indicates some applications to robotic task. Section V and VI shows some experiments related to daily assistance. Section VII presents our conclusion.

II. Related work

Because we aim to create 3D map which is useful for detecting collision and occlusion while at a robotic manipulation, mapping region is only an area where robot arms are able to reach. When a LRF mounted on movable platform is used, we should care online changing of environmental shape.

One of the popular methods to represent 3D map is voxelization. Cartesian space is divided into orderly cubes, measurement points are voted into voxels. Using only voxels which have sufficient points are used as environment map. It is simple and easy to implement, but relatively memory-hogging approach. Octree has been applied to reduce the drawback [3] [8]. Wurm et al. [11] proposed Octomap as compact map representation.

Digital elevation map (often called DEM or 2.5D map) has been used for navigation of mobile robots [2] [6]. A horizontal plane is first defined, and it is divided into

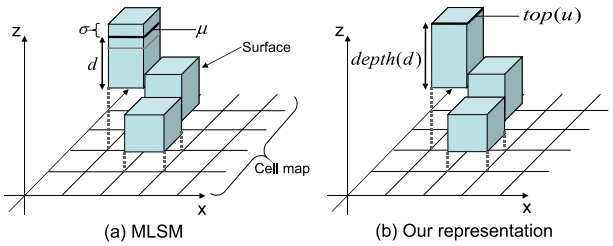

Fig. 2. Map representation

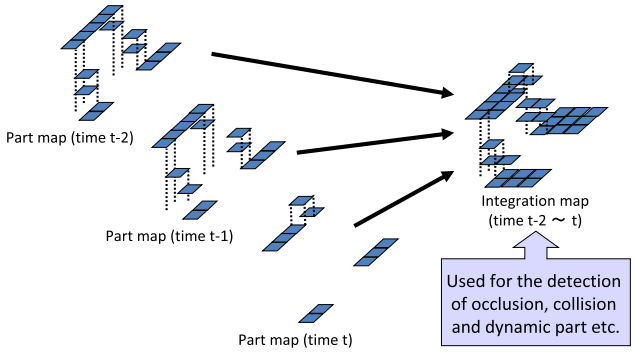

Fig. 3. The concept of TCCM

orderly arranged cells. Height value is input into each cell, environment shape can be represented with low memory use. However, it has poor expression power in regard to vertical structure because one cell can have only one height value. As an extension of DEM, Sphere-DEM [5] was proposed. Although it permits to represent the shape of collapsed buildings which often consist of vertical structure, the map is based on spherical reference surface. Because daily environments have many horizontal planes where manipulatable objects are placed on, a reference plane should be flat.

Some other researchers coped with measurement points directly [7]. In their approach, an environment map is basically represented as pointcloud. Parts of the points were able to remove by combining with 3D model fitting [9]. If measurement points exist in a fitted model, they were replaced with a simple primitive. This approach will work well if mostly static and known environment is targeted, but many points should be just registered in dynamic and unknown environment. It may be relatively hard for a robot software having limited memory and CPU power.

Comparing with above researches, our approach contributes to online mapping about the following items:

- Taking over the advantages of Multi Level Surface Map, low memory use and high power of expression are able to be achieved.
- The compliance of temporal order of the environment map was kept according to addition and subtraction operation. The effective procedure of the operation is also proposed.

III. TCCM (Time-series Composite Cuboid Map)

This section describes about our proposed representation 'TCCM'. The word 'cuboid' in this paper is similar to 'surface' in MLSM.

A. Compositional unit of MLSM and TCCM

TCCM is inspired by MLSM [10]. MLSM represents environment shape by a group of 'surfaces' which are defined as the element of a cell map. The cell map is defined on a horizontal $X-Y$ plane, one cell can have several surfaces. Unlike 2.5D representation, MLSM is able to represent vertical structure. Moreover, memory use is significantly lower than voxelization.

Fig.2(a) shows basic structure of multi-level surface. If a measurement point (3D point $\mathbf{x} = (x, y, z)$) exists inner a cell, a surface is generated with setting the z coordinate as μ and adding measurement error as σ. In the case of MLSM, if the top of a surface is near to the bottom of another surface, they are fixed each other and a depth value d is defined. However, our cuboid representation consists of two parameters (u, d) as shown in Fig.2(b). Cuboids are not bound unless two cuboids have overlapped part. That is, we cope with such connection severer than MLSM.

A procedure of updating of cuboid is as follows: first, a belonging cell of a measurement point $\mathbf{x}_i = (x_i, y_i, z_i)$ is specified. One cuboid which has the z_i value on u is defined, and pre-defined d value is set to the cuboid. Next, if another measurement point $\mathbf{x}_j$ exists inside or near the cuboid, it is updated relying on below two patterns:

$$(u,d) = \notin \begin{cases} (z_i, d + z_i - z_j), & if\ z_i \geq z_j \\ (z_j, d + z_j - z_i), & if\ z_i < z_j\ and\ z_i \geq (z_j - d) \end{cases} \quad (1)$$

If $z_i < (z_j - d)$, a new cuboid is generated with setting the top value as z_j. Although the variable d is constant in our assumption, it can also be decided depending on interval of sensor measurement.

B. A concept of TCCM

Fig.3 shows the concept of TCCM. Using a group of measurement points at time t, a map m_t is generated. Meanwhile, by gathering and combining n number of time-series maps, a map M_t can be generated. In the rest of this paper, we call them 'part map' and 'integration map' respectively. The integration map can be written as $M_t = \sum_{i=t-n+1}^{t} m_i$.

According to C++ style, the structure of a cuboid is defined as follows:

```
struct TCCM {
  int pnum, col, dyn;
  float u, d;
  list < int > msnlist;
};
```

where *pnum* is a variable which registers the number of points voted. *col* is a variable representing 'all clear', 'collision' and 'occlusion', *dyn* is a variable representing degrees of reliability about whether or not the cuboid belonging to dynamic part. *msnlist* is an integer list only for integration map. Serial number of part map whose cuboids are utilized to construct the integration map are registered to the list in

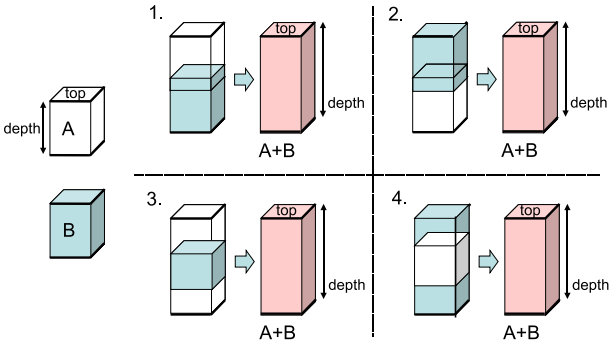

Fig. 4. Addition of cuboids

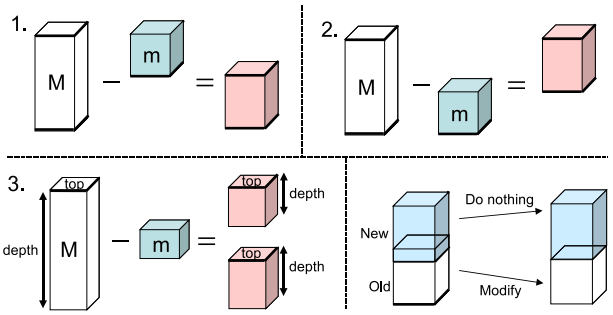

Fig. 5. Subtraction of cuboids

order. When we need to know the organization of a part of a cuboid in the integration map, this structure enables us to search an original cuboid from part maps. This is why our representation permits strictly management of clock time related to the timing of sensor measurement.

As mentioned above, volume of map information is able to significantly be reduced because one measurement result is translated into one cuboid map (part map). Moreover, as described in next subsection, this strategy enables efficient update of an integrated map.

C. Constitution method of an integration map M_t

Two approaches are proposed: (i)**addition of all part maps**: n successive part-maps are simply added to an integration map, (ii)**replacement of old map by new map**: after a latest map m_t is added to an integration map, an oldest map m_{t-n} is subtracted from it.

1) Addition of all part maps: One simple approach making an integration map is an addition of several part maps. This is easy to implement, but the processing time is relatively long because n times addition is needed every measurement cycle.

Now the addition process is explained by using two part maps. First, cuboids included in both part maps are copied into cell map which is for an integration map. If two cuboids $C_A : (u_A, d_A)$ and $C_B : (u_B, d_B)$ have overlapped part, they are integrated into one cuboid and are registered to the integration map. There are 4 patterns of duplication and the way for integration. (See Fig.4).

1) If $u_A \geq u_B$ and $d_A < u_B$, d_A is changed as $d_A = (u_A - u_B) + d_B$. After that, C_B is deleted.
2) If $u_A < u_B$ and $d_B < u_A$, d_B is changed as $d_B = (u_B - u_A) + d_A$. After that, C_A is deleted.
3) If $u_A > u_B$ and $d_A \geq d_B$, C_B is deleted.
4) If $u_A < u_B$ and $d_A \geq d_B$, C_A is deleted.

Before the cuboid elimination, a remained cuboid resets *pnum* and *msnlist* of the deleted one.

2) Replacement of an old map and a new map: For more efficient updating than the simple addition described above, we adopt subtraction procedure. Basically, an integration map is generated through two stage procedure, but deformation of existing maps is needed as preliminary processing because it is essential to guarantee of time-series consistency. The procedure is explained below.

Deformation of part maps (Preprocessing): A cuboid in an integration map may consist of several cuboids in part maps. In such case, excessive subtraction occurs when an old part map is subtracted from the integration map. This is because cuboids which include 3D points measured at later time are eliminated by old cuboids.

To cope with such excessive subtraction, intersection of cuboids are removed in advance. First, intersection parts are investigated between the integration map M_t and the latest part map m_t. If such parts were found, existing part maps which compose the relevant cuboids are specified by referring *msnlist* of each cuboid belonging to M_t. After that, cuboids in the existing part maps are shortened by using the information of latest cuboids. Lower right figure in Fig.5 shows the deformation example.

According to above procedure, part maps are improved not to have any duplication part for integration map generation. This deformation causes inconsistent condition at each part map because some of existing parts of cuboids are removed. However, integration map does not have any excess reduction, there is no problem when collision check or other process is performed by using the integration map.

Addition of the latest part map: Following the same procedure as III.C.1).A, latest map m_t is added into the integration map M_t

Subtraction of the oldest part map: The oldest map is subtracted from the integration map. Because of preprocessing, there are no duplication parts between part maps. So the subtraction is easily achieved by removing part of the integration map with the same shape of the oldest part map.

As shown in 1. to 3. in Fig.5, there are 3 subtraction patterns between a cuboid $C_M = (u_M, d_M)$ in the integration map and a cuboid $C_m = (u_m, d_m)$ in a part map:

1) If $u_M = u_m$,
   $u_M = u_M - d_m$, and $d_M = d_M - (u_m - d_m)$.
2) If $d_M = d_m$, $d_M = d_M - (u_m - d_m)$.
3) If $u_M > u_m$ and $d_M < d_m$ are satisfied, the cuboid is divided into two parts. The upper part needs modification as ${}^1u_M = u_M$ and ${}^1d_M = u_M - u_m$. The lower part needs modification as ${}^2u_M = u_m - d_m$ and ${}^2d_M = (u_m - d_m) - (u_M - d_M)$.

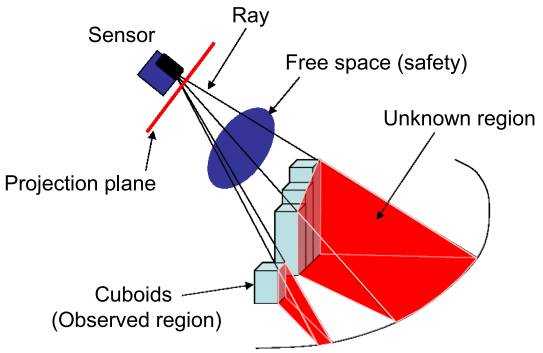

Fig. 6. Discrimination of free space and occluded (unknown) space. Regions where cuboids do not exist is divided two types; free space and unknown space. Although the robot moves freely in the former region, latter region may include unknown obstacles. So projection 3D scene into 2D plane is performed. Red line near to the sensor indicates a plane used for Z-buffer.

This procedure is more effective than the approach described in C.1). The reason is that it limits the searching area to investigate overlapping cuboids. This characteristic arises an advantage that the difference of the number of part maps has little influence on the processing time of updating an integration map (see section V.B).

IV. Extension of TCCM for Collision and Occlusion Detection

Measurement data captured by a LRF mounted on a robot is translated into TCCM. If we utilize this map to decide robot behavior, some additional functions are needed, for instance, 1) collision and occlusion detection between the robot model and cuboids, 2) perception of dynamic region and so on. In this research, a robot body is represented as a group of primitives. The shape of robot arms are approximately modeled by using rectangular box, cylinder, capsule and sphere.

A. Collision detection

Collision detection is achieved by investigating intersection between primitives of the robot model and cuboids in the environment map. One of the approaches to efficient processing is to restrict map areas where disputable cuboids exist. To achieve it, the robot model is first projected into a basic plane of a map, and cuboids existing on the area are selected as collision candidates.

B. Occlusion detection

In order to ensure a safe behavior of the robot, only doing collision detection is insufficient. As shown in Fig.6, it is precisely needed to distinguish 'real free space' and 'unknown space'. Because TCCM permits to represent vertical structure, occlusion detection has important role to check the condition that the robot extends its arm to behind of some objects.

We apply Z-buffer which is a well-known method in the field of computer graphics. Z-buffer is a planar frame with a virtual camera. At the beginning, infinite value is set to all of pixels in the frame, and the pixels are overwritten if the frame captures some objects near to the position of the virtual camera. In our case a virtual camera is set to the same position of a sensor as shown in Fig.6, both primitives and cuboids are projected into the plane. If cuboids are depicted over a part of robot primitives, it is judged as occlusion.

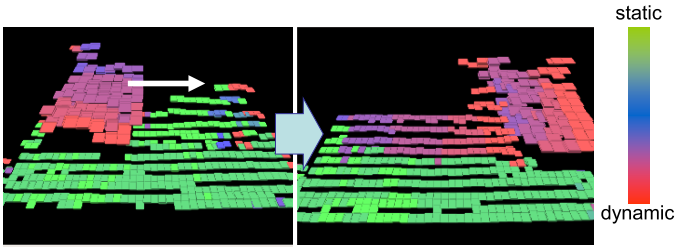

Fig. 7. Application to dynamic region perception: cuboids colored green indicates that they are belonged to static region, and red cuboids are regarded as dynamic region.

C. Dynamic region estimation

It is helpful that an environment map is able to include the information of dynamic or static region from time-series data. Although the map representation described above sections permits to keep temporal sequence, the length of time reserved in an integration map is only about 1 seconds in our case. This is too short to detect dynamic obstacles.

In order to deal with data at more long period, a variable for representing survival times is introduced as *dyn*. This variable is updated whenever new measurement data is mapped inner or near cuboids which are already existing. The updating is performed by the following procedure. Now we consider two integration maps M_{t-n-1} and M_t; the former consists of part maps from $m_{t-n-n'}$ to m_{t-n-1} and latter consists of maps from m_{t-n} to m_t, where $n > 0$, $n' > 0$. If M_{t-n-1} and M_t have cuboids which are almost same position, the cuboids in M_t accept handover of *dyn* variable in M_{t-n-1} after increment it.

Fig.7 shows the example in which a tilting LRF was directed to a top board of a dining table. In this experiment, only the table existed at the beginning, but a person who grasped A4 paper held out the paper between the sensor and the table in the middle of the measurement. After that, while he moved the paper from side to side, cuboids derived from the moving paper and also cuboids occluded by the paper were belonged to dynamic region.

In other words, dynamic region is a region where the measurement has been insufficient for regarding as static one. As such region has higher possibility of changing than static regions, this information is useful for invoking careful motion planning.

V. Performance Verification

Using a Laser RangeFinder (UBG-04LX-F01 [13]) mounted on a tilting platform, we validated our method. The sensor system was equipped on a head of a life-sized robot which consists of a 3-DOF head, 7-DOF arms and a 1-DOF

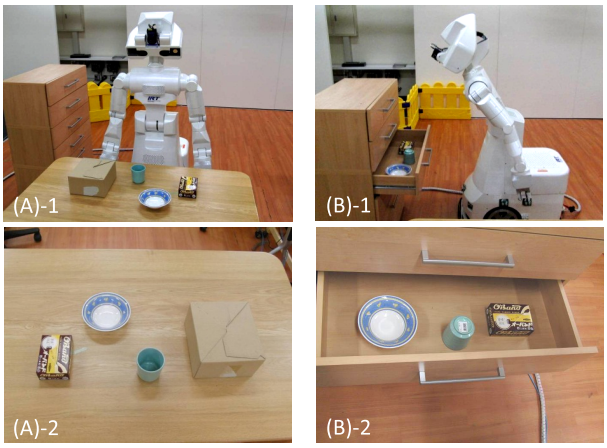

Fig. 8. Examples of experimental situation

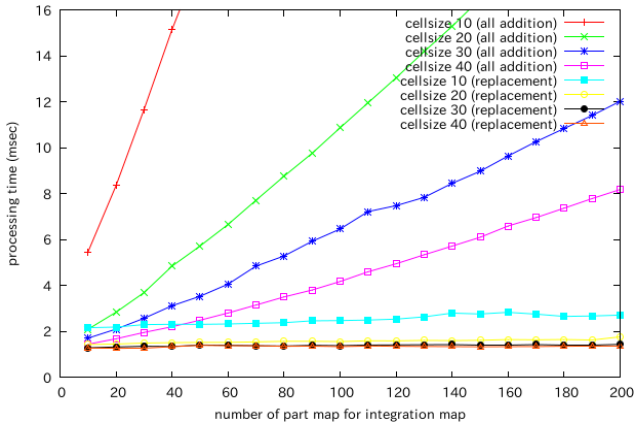

Fig. 9. Processing time for map generation. The horizontal axis indicates the number of part maps used for generating integration map, and vertical axis indicates processing times needed for updating 1 scan data.

torso. The center position of tilting was 1515 [mm] height when the robot faces front.

A. Experimental conditions

Fig.8 shows two examples of experimental situation. Left column shows a situation that several small objects were placed on a dinning table. The robot stood on the front of the table, and measured them. In this case, the center of tilting was set to 45 [degrees] with downward angle and its amplitude was set to 30 [degrees]. Right column shows a shelf which sizes 900 [mm] width, 300 [mm] depth and 1200 [mm] height. The robot opened the drawer by itself, and then the center of tilting is turned on the drawer by moving pitch joints of the waist and the head.

From the viewpoint of obstacle detection for safety task execution, the time for one tilting was set to 1575 [msec]. The reason was that this setting makes it possible to find an object with 50 [mm] width in a sphere with radius 1 [m]. This area covered the workspace of the robot. In this setting, 35 to 40 scans were acquired in one tilting motion.

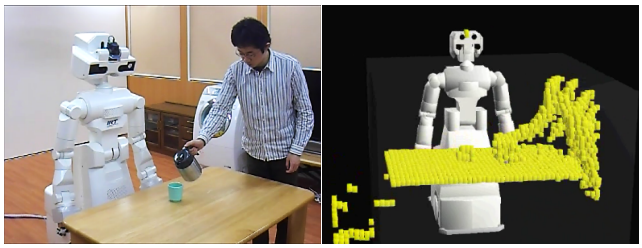

Fig. 10. Mapping result in pouring motion

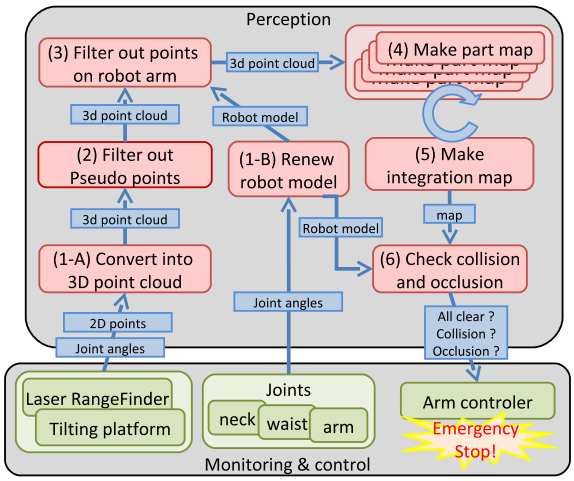

Fig. 11. Software system for collision and occlusion detection

B. Processing time

Fig.9 shows processing times on two types of map generation method described in section III.C under the condition of Fig8, (A). The horizontal axis indicates the number of part maps used for integration map generation, and vertical axis indicates processing times needed for one updating of the map. In this experiment, the average number of cuboids was 1073 under the condition that $1500 \times 2000[mm]$ area was divided into $20 \times 20[mm]$ cells. Comparing with all addition approach, replacement approach kept constant processing time. It took $1.35[msec/scan]$ at 2.1 GHz single CPU when an integration map generated from 40 part maps. One interest point in replacement approach was that processing time for the updating probably had only few changes depending on the number of range data. In general, such a characteristic is useful for generating a detail map because a lot of range data may be needed to fill small size of cells.

In addition to examples shown in Fig.8, comparable performance was also achieved when we tried at other types of conditions. For instance, position of objects was changed, a person extended his arm to the front of the robot and robot tried to grasp an object on a table. Fig.10 shows one mapping result of them.

VI. Collision and Occlusion Detection

Experiments for detecting obstacles were performed. Fig.11 shows the software system we implemented.

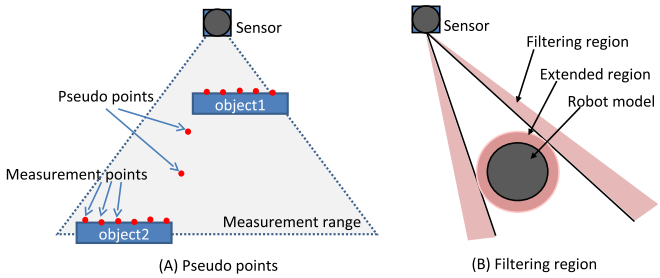

Fig. 12. Filtering needless points. Left: Pseudo points could be found between two objects. Right: Gray circle shows a cross-section view of robot model and pink triangle area indicates a domain of points elimination. pick circular area is also a target of the elimination.

A. *Filtering out measurement points (Fig.11(2),(3))*

Left figure of Fig.12 shows a problem which is peculiar to a LRF. When two objects are placed in different distance from the sensor, pseudo points often arise between them. This phenomenon becomes prominent if objects are placed on near to the LRF. To cope with this, a point whose nearest neighbors are separated from it is not used to mapping.

On the other hand, although an environment map should be generated from measurement points which pick up only environment shape, actual points can include from a part of robot body because the robot moves its arms in the environment. To eliminate such points, the shape of the body was approximated by 26 primitives. If some points existed inner or near the primitives, they were not used in the rest of processes. Purple rectangle area shown in Fig.12 is also targets to point elimination.

B. *Experimental setting*

In order to effectively detect collision and occlusion, below two rules were applied:

- Primitives of the robot model is virtually expanded. If they overlap a part of cuboid, it is judged as collision.
- In addition to present pose, a future pose several hundreds [msec] later is also targeted for checking.

In this experiment, the expansion ratio was set to 1.2, and a 500 [msec] later pose was also checked.

C. *Experimental results*

Three types of obstacles: a pole, human arm, and a rectangular box shown in Fig.13 were used. While the robot were stretching its arm on a table, an obstacle was suddenly placed on about a front of 150mm away from the moving end-effector (for collision) or was inserted between the sensor and the end-effector (for occlusion). The motion was immediately stopped whenever the robot detected them.

Three series of speed 100mm/sec, 250mm/sec and 400mm/sec were set to the end-effector. 10 times of experiments were tried at each combination of obstacles and end-effector speed. Although tilting mechanism moving with cycle T had a constraint that the time observing same place again could take $3/4T$ in the worst case, no collision was observed through all 90 trials.

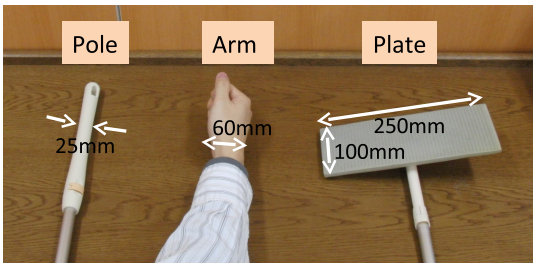

Fig. 13. Obstacles for experiments

VII. Conclusion

We proposed a novel map representation which explicitly considers temporal alternation of measurement data. Applications to collision and occlusion detection were experimented by using a real robot. These results showed the effectiveness of our approach, and also indicated the possibility of fast updating and high power of expression of the map.

Future work, we will try to combine this approach with other types of method, for instance, plane detection and segmentation of moving obstacles and so on. Combination with reactive motion planning is also interesting.

References

[1] C. Ye and J. Borenstein: "A new terrain mapping method for mobile robots obstacle negotiation," In Proc. of the UGV Technology Conf. at the 2002 SPIE AeroSense Symposium, pp.21–25, 2002.

[2] J. R. Collins, Y. Tshin and A. Lipton: "Using a dem to determine geospetial object trajectories," in DARPA Image Understanding Workshop, pp. 115–122, 1998.

[3] N. Fairfield, G. Kantor and D. Wettergreen: "Real-time SLAM with octree evidence grids for exploration in underwater tunnels," Journal of Field Robotics, pp.3–21, 2007

[4] M. Hevert, C. Caillas, E. Krotkov, I. S. Kweon and T.Kanade: "Terrain Mapping for a Roving Planetary Explorer," Proc. of IEEE Int'l Conf. on Robotics and Automation, vol.2, pp.997–1002, 1989.

[5] K. Nagatani, H. Ishida, S. Yamanaka and Y. Tanaka: "Three-dimensional Localization and Mapping for Mobile Robot in Disaster Environment," Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems, pp.3112–3117, 2003.

[6] F. Nashashibi, P. Fillatreau, B. DacreWright, T. Simeon: "3-D Autonomous Navigation in a Natural Environment," Proc. of IEEE Int'l Conf. on Robotics and Automation Vol.1, pp.322–439, 1994.

[7] A. Nuchter, K. Lingemann, J. Hertzberg and H. Surmann: "6D SLAM – 3D mapping outdoor environment: Research articles," Journal of Field Robotics, Vol.24, No. 8–9, pp.699–722, 2007.

[8] P. Payeur, P. Hebert, D. Laurendeau, C.M. Gosselin: "Probabilistic Octree Modeling of a 3-D Dynamic Environment," Proc. of IEEE Int'l Conf. on Robotics and Automation, pp.1289–1296, 1997.

[9] R. B. Rusu, N. Blodow, Z. C. Marton and M. Beetz: "Close-range Scene Segmentation and Reconstruction of 3D Point Cloud Maps for Mobile Manipulation in Domestic Environment," Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems, pp. 1–6, 2009.

[10] R. Triebel, P. Pfaff, W. Burgard: "Multi-Level Surface Maps for Outdoor Terrain Mapping and Loop Closing," Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems, pp. 2276–2282, 2006.

[11] K. M. Wurm, A. Hornumg, M. Bennewitz, C. Stachniss and W. Burgard: "Octomap: A Probabilistic, Flexible and Compact 3D Map Representation for Robotic Systems,",2010

[12] M.Yguel, C. Keat, C. Braillon, C. Laugier and O. Aycard: "Dense Mapping for Range Sensors: Efficient Algorithm and Sparse Representation," in Proc. of Robotics: Science and Systems, pp.129–137, 2007.

[13] Scanning range finder (SOKUIKI sensor): http://www.hokuyo-aut.jp/02sensor