Cuboid-based Workspace Mapping and Plane Detection using Time-series Range Data

Kimitoshi YAMAZAKI¹, Takemitsu MORI², Takashi YAMAMOTO² Masayuki INABA*¹

Abstract—This paper describes methods of environment map representation and plane detection for daily assistive robots. Under the assumption that 3D information of around environment is constructed by collecting range data gradually measured, we propose an effective map representation named TCCM (Time-series Composite Cuboid Map). The method copes with temporal sequence explicitly, and map updating is efficiently performed. In addition, two plane estimation methods are proposed. Through several experiments in daily environments, we ensured the effectiveness of our method.

I. INTRODUCTION

One of the important functions needed to a daily assistive robot is to figure out the shape of environment where the robot works. This paper describes a map representation method which makes it possible to handle plenty of time-series sensing data captured from Laser RangeFinder. Effective data compression can be achieved with keeping expression power of 3D environment shape as much as possible. In addition, a method to detect ‘planes’ from the map is also proposed.

When we focus on an issue that a robot handles objects placed on a table, one of the feasible approaches is to plan robot motion based on results of range sensing. Recently, several range sensors have succeeded downsizing and weight saving [17] [18], such approach is becoming more common. However, a considerable problem is their limited measurement range. If a set of data which is sufficient to plan the motion of whole body of the robot is needed, a sensor should be moved for measuring over a wide range.

This fact tells us the importance of a framework which is able to cope with time-series range data. In addition, plane detection is feasible application to daily assistive robots.

This paper describes about workspace mapping and plane detection. A map generated by using the representation consists of a number of cuboids all of which are given time-series information. The representation named TCCM (Time-series Composite Cuboid Map) makes it possible to update a map gradually and to detect planar structure effectively.

This paper is organized as follows: next section describes about related work and our contribution. Our map representation TCCM is explained in section III, and the plane detection method is explained in section IV. Section V and VI report some experimental results and section VII concludes this paper.

¹ Department of Mechatronics–Informatics, Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, Japan ⁰² Toyota Motor Corporation {yamazaki, inaba}@jsk.t.u-tokyo.ac.jp

II. RELATED WORK

One of the popular methods to represent 3D map is voxelization. Cartesian space is divided into orderly cubes (voxels), measurement points are voted into them. Using only voxels which have sufficient points are used as environment map. It is simple and easy to implement, but relatively memory-hogging approach. For more efficient processing, octree has been applied to reduce the drawback [3] [8]. Wurm et al. [12] proposed Octomap as compact map representation.

Digital elevation map (often called DEM or 2.D map) has been used for navigation of mobile robots [1] [6]. Horizontal plane is first defined, and it is divided into orderly arranged cells. Height value is input into each of the cells, 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. The map is based on spherical reference surface so that it permits to represent the shape of collapsed buildings with vertical structure. In daily environment, however, a reference plane should be flat because such environment has many horizontal planes where manipulatable objects are placed on.

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 [10]. If measurement points exist in a fitted model, they were replaced with the model. This approach will work well if mostly static and a known environment is targeted, but many points should be just registered in dynamic and unknown environments. It may be relatively hard for a robot software having limited memory and CPU power. In addition, time-series management is comparatively difficult.

As other approaches focusing on lossless compression, Yguel et al.[16] applied wavelet transform. In the viewpoint of memory consumption this approach is useful, but the procedure is relatively complicate and time consuming.

Comparing with these researches, we focus on domestic condition, especially a restricted area where robot arms are able to reach. The contributions of TCCM are as follows:

1) Taking over the advantages of Multi-Level Surface Map proposed by Triebel et al.[11], effective representation can be achieved with a small number of elements and high power of expression.
2) An environment map keeps the compliance of temporal order while it updates its shape by bringing in addition and subtraction operation.
3) As extension of item 1), a plane detection method is proposed.

III. TCCM (TIME-SERIES COMPOSITE CUBOID MAP)

This section describes about 'TCCM'. The word 'cuboid' in this paper is almost the same as 'surface' in MLSM. Basic idea have already been described in [13], but we try to explain it again because there are some extension about bringing in center core.

A. Compositional unit of MLSM and TCCM

TCCM is inspired by MLSM [11]. 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, the number of elements is significantly smaller than voxelization.

Fig.1(a) shows basic structure of multi-level surface. If a measurement point (3D point \( x = (x, y, z) \)) exists inner a cell, a surface is generated with setting the \( z \) coordinate as \( \mu \) and adding measurement error as \( \sigma \). 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. Meanwhile, our cuboid representation consists of two parameters \( (u, d) \) as shown in Fig.1(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 \( x_i = (x_i, y_i, z_i) \) 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 \( x_j \) exists inside or near the cuboid, it is updated relying on below two patterns:

\[
(u, d) = \begin{cases} 
(z_i, d + z_i - z) & \text{if } z_i \geq z_j \\ 
(z_j, d + z_j - z_i) & \text{if } z_i < z_j \text{ and } z_i \geq (z_j - d)
\end{cases}
\]

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.2 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 \( M_t = \sum_{i=-n+1}^{t} m_i \).

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

```cpp
struct TCCM { 
  int pnum, cls; 
  float u, d; 
  list < int > msnlist; 
  float p[3], P[3][3], s[3], S[3][3]; 
};
```

where \( pnum \) is a variable which registers the number of points voted. \( cls \) is a class label explained at section IV. \( u \) and \( d \) represent height and depth of a cuboid respectively. 4 floating arrays are used for registering center cores described in next section. \( msnlist \) 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 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, the number of map elements is able to 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 \( M_t \)

To achieve efficient map renewal process, an approach of cuboid replacement is adopted [13]. After the latest map \( m_t \) is added to an integration map, the oldest map \( m_{t-n} \) is subtracted from it.

1) Replacement of an old map and a new map: 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 older part maps (preliminary processing): 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 \( M_1 \) and the latest part map \( m_l \). If such parts were found, existing part maps which compose the relevant cuboids are specified by referring \( msnlist \) of each cuboid belonging to \( M_1 \). After that, cuboids in the existing part maps are shortened by using the information of latest cuboids. Lower right figure in Fig.3 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.

Renewal of an integration map: After the preliminary processing mentioned above, a latest map \( m_l \) is added into the integration map \( M \). Because there are no duplication parts between part maps, the latest map is simply combined with original integration map.

Next, the oldest map is subtracted from the integration map. The process 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.3, 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 \( ^2d_M = (u_m - d_m) - (u_M - d_M) \).

### IV. PLANE DETECTION

A. Horizontal Plane Detection

In daily environment, many of objects to be handled exist on horizontal plane. Detection function of such plane is useful for assistive robots assumed to do pick-and-place task. Planes having same slope with basic plane of TCCM is easy to find. We adopt a clustering algorithm which is based on recursive search to cuboids belonging to neighbor cells, and then a group of cuboids which have almost same \( u \) value is regarded as constructing one horizontal plane.

Basic procedure is as follows: Starting in a certain cell, 8 neighbors of the cell are referred. If the difference of \( u \) values between cells is under predefined threshold, same class label is given to the cells. The variable \( cls \) described in section III.B is used for this clustering.

B. Center core of cuboid

This section describes about center core which is additional information into cuboid. It is preliminary knowledge for detecting planes without relying on its slope.

Center core is equal to long axis of ellipsoid estimated from points in a cuboid (See Fig.4). The axis is able to be calculated by eigenvalue decomposition of covariance matrix related to 3D position of the points. Because this representation builds up map resolution, it is useful for a cuboid which stores biased distribution of points.

The calculation of average and covariance of points is as follows:

\[
\mathbf{p}_N = \frac{\mathbf{s}_N}{\mathbf{N}}, \quad \mathbf{P}_N = \frac{\mathbf{s}_N + \mathbf{p}_N\mathbf{s}_N^T}{\mathbf{N}}, \quad (2)
\]

where \( T \) denotes transpose of a matrix and a vector. \( N \) is the number of points. \( s \) and \( S \) are

\[
\mathbf{s}_N = (N - 1)\mathbf{p}_{N-1} + \mathbf{x}_N, \\
\mathbf{S}_N = \mathbf{S}_{N-1} + \mathbf{x}_N\mathbf{x}_N^T, \quad (3)
\]

where \( \mathbf{x} \) denotes 3D position of a point. By using these equation, average and covariance are sequentially calculated.

On the other hand, as shown in Fig.4, deformation of the distribution can be performed obeying the following rules:

- Addition of cuboids \( C_1 \) (the distribution is \( N(\mathbf{p}_1, \mathbf{P}_1) \)) and \( C_2 \) (the distribution is \( N(\mathbf{p}_2, \mathbf{P}_2) \)):
also proposed [2] [9]. In this approach, a pair of neighbor points are used for estimating parameters of plane equation, and other points are used for validating the parameters by means of least median square [14]. Meanwhile, region growing was also proposed [2] [9]. In this approach, a pair of neighbor points are generated, and planar regions is expanded from the pair. Both of these approaches need 3D points as initial state.

Because all of cuboids have center core information in our method, more effective plane detection can be achieved according to the following procedure: first, l number of center cores are randomly selected, and following calculation is performed by using two of them.

\[
a = \frac{j p - i p}{j p - i p} \tag{6}
\]

where \( n \) indicates a vector directed to center core, \( i = 1, ..., l \) and \( j \) is the same but \( i \neq j \). If the two cores lie in the same plane, it means that \( \mathbf{b}^i \) and \( \mathbf{b}^j \) exist with same or opposite direction. Fig.6 shows the relation of two center cores.

The pairing and fitting are performed against all of possible pairs from \( l \) number of center cores. If all of cross products \( \mathbf{b} \) satisfy the following condition, we regard that a plane is found.

\[
\left\{ \cos^{-1} \frac{i \mathbf{b} \cdot j \mathbf{b}}{|i \mathbf{b}|^2 |j \mathbf{b}|} \text{ or } \cos^{-1} \frac{-i \mathbf{b} \cdot j \mathbf{b}}{|i \mathbf{b}|^2 |j \mathbf{b}|} \right\} < \text{threshold} \tag{7}
\]

If this criteria passes, the average of \( k \mathbf{b}(k = 1, ..., l) \) is calculated and then it is substituted into \( \mathbf{b}^{\mathbf{p}} \mathbf{p} + d = 0 \) for calculating \( d \), we finally get 4 parameters of plane equation. In our implementation, \( k \mathbf{p} \) is a center position of the longest cuboid.

After estimating the plane parameter, all of center cores are checked whether or not they belong to the plane. Two criteria are used as follows:

- angular threshold: the direction of \( n \) is almost vertically from \( \mathbf{b}^{\mathbf{p}} \).
- distance threshold: \( d \) value of plane equation is sufficiently-small with substituting \( \mathbf{b}^{\mathbf{p}} \).

Finally, three eigenvalues \((\lambda_1, \lambda_2, \lambda_3)\) provided from a covariance matrix which is calculated from all of center cores are used to judge qualities of the results. That is,

\[
\text{val} = \frac{\lambda_1 \lambda_2}{\lambda_3} \tag{8}
\]

where \( \lambda_1 \geq \lambda_2 \geq \lambda_3 \).

If necessary, simple clustering can also be applied to the selected center cores. A criterion of grouping the cores is that the distance between center cores must be in the threshold. The clustering procedure is better able to be streamlined.
V. PROOF EXPERIMENTS FOR TCCM GENERATION

A. Settings

A LRF (UBG-04LX-F01 [17]) was installed on tilting platform, and the platform was mounted on the head of a life-sized robot. The joint configuration of the robot is 7 DOFs arms, 1 DOF waist and 3 DOF neck. The height of the tilting platform was 1520 mm when the robot stood erect.

Fig. 7 (A)-1,2 and (B)-1,2 shows two examples of experimental conditions.

(A): A dining table with several objects: A dining table the height of which was 700 mm was set on the front of the robot. Several small objects were placed on the table. The robot looking obliquely downward measured its front environment by the LRF tilted +/- 30 degrees.

(B): A shelf stored several objects: A shelf the size of which was 900 width, 300 mm depth and 1200 mm height was used. Several small objects were stored in 3rd drawer. The robot standing in the front of the shelf tried to open the drawer, and measured the content of it. Waist and head joints not only arm joints were moved, inner of the drawer was scanned by the LRF tilted +/- 30 degrees.

Following results show performances on offline processing by using 2.24 GHz CPU, Linux OS.

B. Processing time of map generation

5000 number of range data measured at daily environment shown in Fig.7(A)-1 were used for generating TCCM. The graph depicted in Fig.8 shows the number of range data for an integration map vs. processing time. 4 patterns cell size 10mm; 20mm; 30mm and 40mm are examined.

In this experiment, the time for one tilting cycle was set to 1575 msec. This was because a robot should sense an object which is bigger than 50mm in its workspace (within a radius of one meter). Because the scanning time of this LRF was 28msec/scan, 35 to 40 scans could be acquired while one tilting cycle.

As shown in Fig.8, the processing time of map updating was sufficiently short in all of the cases that 40 number of scans were used to generate an integration map. For example, only 1.35msec was needed for a map updating the size of the cell which was 20mm. Another interest point was that processing time for the updating a map probably had only few changes when the number of scan was increased.

In the case of Fig.7(A)-3, 569 x 40 points were translated into 1300 cuboids. So the reduction rate of the number of map elements was seventeenth. Comparable performance was also achieved when we tried at other types of conditions, for instance, the 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.

VI. PROOF EXPERIMENTS FOR PLANE DETECTION

A. Horizontal plane detection

Fig.9 shows a measurement result of the top board of a table which was 1200mm width. Despite horizontal plane was measured, we can observe about 20mm error in height value. Because such error may cause of calibration error of
difference of could be removed. In this experiment, threshold related to plane. Top board of the table and bottom plate of the drawer maps after removing cuboids corresponded to horizontal detection described in section IV.A. The lowest row shows tilting platform and measurement accuracy of the LRF, it is difficult to completely remove them.

Fig. 7 (A)-3, 4 and (B)-3, 4 shows before and after the plane detection described in section IV.A. The lowest row shows four types of situations shown in Fig. 10 were used for evaluation of the center core based plane detection. Using a set of measurement data generated from tilting LRF mounted on a robot head, processing time and estimation accuracy on plane detection were evaluated. In these simulation, position error obeying to normalized distribution $N(0, \sigma^2)$ was added to each measurement points.

For the purpose of comparison, plane detection program included in PCL (Point Cloud Library) [19] was used. In this program, a plane parameter and points belonging the plane is estimated based on RANSAC algorithm. On the other hand, proposed plane detection was performed 50 or 10 trials, and then the parameter which indicated the largest value at equation (8) was selected.

TABLE I shows some of the evaluation results. Each cell indicates average and standard deviation when 20 times testing in each case. The position of the top of the table was 700 mm, and that of the front board was 600 mm. TABLE II shows processing time. Although the proposed method has disadvantage at the point of repetitive accuracy of the estimated position, processing times were 10 times faster than PCL.

C. Experiments using real sensor data

Fig. 11 shows center cores from a dataset measured at the front of a shelf shown in Fig. 7 (B). Left figure shows elevation view and right figure shows side view. Near $y = 0$, $z = 1000$ where vertically long center cores can be found is the place where many number of cuboids were combined. Because each drawer had a small knob, the range data included some projection portion. Moreover, a floor and the front of the shelf did not form horizontal and vertical plane because of small inclination of the robot’s wheelbase. The use of center core adequately represented such quality of the data.
TABLE III shows other detection results under the condition that the original 3D points were variously rotated and translated. This was an example for investigating the applicability of our method for finding planes with arbitrary inclination angle. Although it was sometimes hard to get long center cores from rotated data, the results shown in the TABLE III indicate that planes was able to be found in all trials. In our experience, plane candidates having \(\text{val} \) more than 0.15 was sufficient to identify as reliable results.

Fig.12 shows examples of other experiments. In the left column, a person having white plate moved it to the front of the robot. On the other hand, a package box was placed on the table in the right column. Both of two cases, we could find 2 planes as shown in the middle and the lowest rows in the Fig.12. These figure depicted remaining points which were regarded as structural element of estimated planes. From these results, we can conclude that center core based approach can be used for finding planes with arbitrary inclination angle, not only vertical planes.

VII. CONCLUSION

The contributions of this paper were summarized as follows:

- we explained a novel map representation which was suitable for time-series of range data.
- We also proposed two plane detection methods. One of them, the concept using center core defined inner cuboid was proposed, and a detection method based on random sampling of the center cores was represented.
- Through simulation and real experiments assuming that a robot performed daily assistance, the performance of our method was investigated.

Future work, we apply this method to picking and placing task by a robot. One of the applications is that the robot does the work without collision in environments including dynamic existence like people.

REFERENCES

[18] MESA Imaging, SR4000: http://www.mesa-imaging.ch
[19] Point Cloud Library: http://www.pointclouds.org