Proceedings of 2020 IEEE
International Conference on Mechatronics and Automation
October 13 - 16, Beijing, China

Study on Map Construction of Spherical Robot based on Statistical Filtering

Jian Guo[1] and Xiangyu Chen[1]
[1]*Tianjin Key Laboratory for Control Theory&Applications In Complicated systems and Intelligent Robot Laboratory Tianjin University of Technology Binshui Xidao Extension 391,Tianjin,300384,China*
jianguo@tjut.edu.cn;
794415466@qq.com

Shuxiang Guo[1,2*]
[2] *Department of Intelligent Mechanical Systems Engineering, Faculty of Engineering*
Kagawa University Takamatsu,Kagawa,Japan
*corresponding author : guo@eng.kagawa-u.ac.jp

Jigang Xu[3*]
[3]*Unit68709*
Qinghai Haidong,810700,China
*corresponding author : xujigang216@163.com

***Abstract* – Traditional robots lacked the detection and preservation of environmental information. This paper proposes to apply SLAM to robots. We put a camera on the robot to capture the environment and map it. Traditional SLAM algorithms typically generate point cloud maps, which are bulky and unreadable and cannot be used for robot navigation. Therefore, point cloud map is generally converted into octree map, but due to environmental interference, sensor error and other reasons, point cloud map often contains a lot of noise, which leads to a large error in the generated octree map. In this paper, based on OBR SLAM2 algorithm, proposed a filtering method, through the judgment of point cloud map points automatically select filtering mode, thus generating high quality octree map. Compared with unfiltered maps through experiments, the map generated by this method is smaller volume and more accurate.**

Index Terms – ORB SLAM2, Octomap, The spherical underwater robot

I. INTRODUCTION

Simultaneous localization and mapping (SLAM) is a key technology for mobile robot autonomous exploration in unknown environment. The pioneering work on SLAM was done by r. Smith, m. Self, and p. Cheeseman in 1986 on the representation and estimation of spatial uncertainty. Early SLAM algorithms mainly use laser sensors, infrared sensors and other sensors to sense the environment. As the performance of the camera increases and the size of the camera shrinks, the camera can be used on the mobile robot. In 2003, Andrew Davison used the extended kalman filter (EKF) for the first time to apply the classical non-camera algorithm for SLAM to the single camera scheme for SLAM. In 2007, Georg Klein and David Murray divided tracking and mapping into two separate threads inspired by Nister algorithm of bundle adjustment (BA), which improved the accuracy and operation speed. ORB_SLAM is a mastermind of the keyframe-based SLAM faction. The orb-slam algorithm basically follows the framework of PTAM and adds modules that have been proven to be effective in recent years to create an all-powerful system with high stability and precision, which can be used for indoor/outdoor and small-scale/large-scale scenarios, as well as high quality open source code.[1]-[7]

ORB SLAM has two purposes: to estimate the robot's trajectory and to build the correct map. Maps can be expressed in many ways, such as feature point maps, grid maps, topological maps, and so on. The current map format is mainly point cloud map. In the program, we splice point clouds according to the optimized pose, and finally form a map. This approach is simple, but has some obvious drawbacks: first, the map is not compact. Point cloud maps are usually very large, with a 640×480 image producing 300,000 points of space, requiring a lot of storage. Even after some filtering, the PCD file is large. And importantly, its "big" isn't required. Point cloud maps provide a lot of unnecessary detail. We don't particularly care about wrinkles on the carpet, shadows in the shadows. It's a waste of space to put them on a map. Second, the way of dealing with overlap is not good enough. When constructing the point cloud, we directly put it together according to the estimated pose. When there is an error in the position, the map will overlap obviously. Finally, it is difficult to use for navigation. A point cloud map cannot get navigation information, such as where there are obstacles, where can be passed, etc.[8]-[11]

This paper mainly studied the robot location and map reconstruction when the robot moves on the land. We install a monocular camera in front of the robot to collect images. When the robot is working, the camera will collect images and send them to the controller. After receiving the image, the controller will preprocess the image first, and then identify the feature points and then compare them with the same feature points in the previous image to judge the camera's position and pose, so as to determine the robot's position.

II. THE SPHERICAL UNDERWATER ROBOT

In recent years, many researchers in the world are working on the development and application of mobile robots. As an important tool, mobile robot is used in environment detection. The spherical underwater robot is a kind of amphibious step movement, and also has visual feedback system, which can be

used as a modern high-tech exploration tool for exploring unknown environment. Its structure is shown in Fig.1.

Spherical amphibious robot is a kind of underwater robot. It can not only walk on land, but also perform horizontal movement and rotation in water. At the same time, the robot has a large internal space, which can guarantee a long endurance. Robots are also less noisy and can help the military. Therefore, spherical amphibious robot has wide application prospect and development value. Its gait on land is shown in Fig.2.

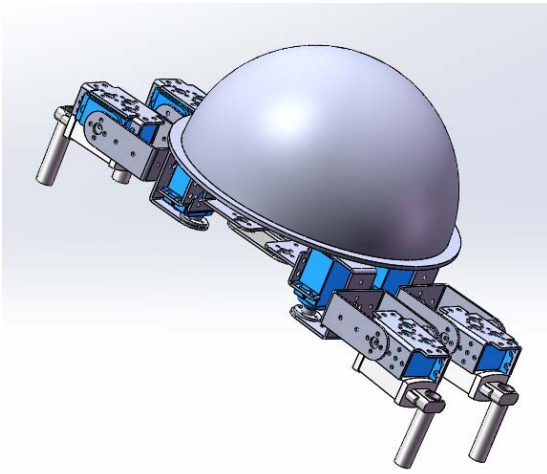

Fig.1 The spherical underwater robot's structure

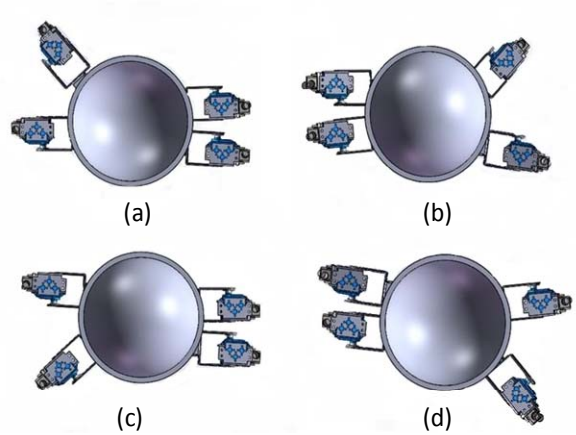

Fig 2 Robot's gait

III. Mapping

A. *ORB SLAM2's Mapping*

ORB SLAM2 is a feature point-based real-time monocular SLAM system that can operate in large-scale, small-scale, indoor and outdoor environments. The system is also robust for vigorous exercise and supports closed-loop detection and relocation of wide baselines, including automatic initialization. The system includes modules common to all SLAM systems: Tracking, Mapping, Relocalization, Loop closing.[12]-[15]

ORB SLAM2 is mainly divided into three threads: Tracking, LocalMapping and LoopClosing:

- Tracking: the main work in this part is to extract the ORB features from the image, make attitude estimation based on the previous frame, or initialize the pose through global relocation, then track the reconstructed local map, optimize the pose, and determine the new key frame according to some rules.
- LocalMapping: this part mainly completes Local map construction, including inserting key frames, verifying the recently generated map points and filtering them, then generating new map points, using Local bundle adjustment (BA), and finally filtering the inserted key frames to remove the redundant key frames.
- LoopClosing: this part is mainly divided into two processes, namely closed-loop detection and closed-loop correction. The closed loop detection was first detected by WOB, and then the similarity transformation was calculated by Sim3 algorithm. Closed-loop correction is mainly about closed-loop fusion and Graph optimization of Essential Graph.

In the previous Tracking we got the new keyframe k_i. The next step is Local Mapping, which includes inserting keyframes, eliminating redundant map points and keyframes, and making Local cluster adjustments.

1. Keyframe insertion

First add the new keyframe k_i as the new node Covibility Graph and update the edges connected to those keyframe nodes that can share map points. At the same time, update the growth tree of keyframe k_i and calculate the word bag BOW that represents the keyframe.

2. Current map point culling

Only map points that can be saved by creating the first three frame constraints of the point cloud are guaranteed to be traceable and not easily subject to large errors during triangulation

3. New map point created

By finding the detected ORB feature points, find the key frame k_c in the Covisibility Graph that is connected to them, make the feature matching, and then triangulation the matched feature points. After triangulation of ORB feature points, check forward depth of field, parallax, backprojection error, and scale consistency to get the new map points. A map point is observed through two keyframes, and it can also be projected to other keyframes connected to it. In this case, the Tracking part can be used to find matches in nearby keyframes to get more map points.

4. Local cluster adjustment

Local cluster tuning optimizes the currently processed keyframe k_i

5. Local keyframe culling

To control the compactly of the reconstruction, LocalMapping detects redundant keyframes and then removes them, which helps control. In this paper, those keyframes that have 90% of the points can be observed by more than three keyframes are considered as redundant keyframes. The algorithm flow is shown in Fig.3.

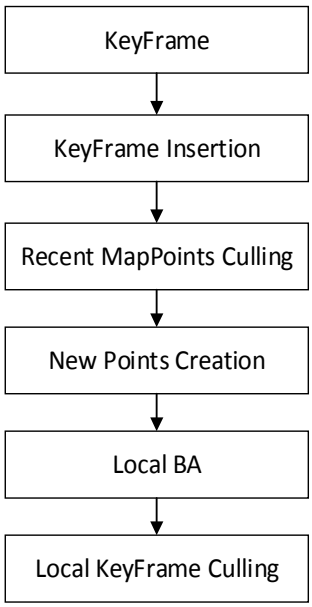

Fig.3 Mapping Process

Finally, at the end of all the steps, the keyframes are logged into the database list.[16]-[25]

B. Octomap

In our system, each keyframe stores a 3D point cloud composed of related feature points. We can generate the map based on the attitude associated with the keyframe and the 3D point cloud. However, the point cloud map is only a coordinate set of points in three-dimensional space, which does not contain volume information and cannot provide information for robot navigation. At the same time, point cloud map cannot distinguish whether an area is unknown or open, and point cloud map is easy to be disturbed by noise. So this paper uses octree map instead of point cloud map. Octree maps use octree memory point clouds to distinguish between unknown and empty areas and eliminate noise when inserting new points. In octree, we use probability to express whether the space is occupied or not during the observation of the environment. Due to the existence of noise, a square may be observed to be occupied sometimes, and after a while, it is not occupied in other squares. This may be due to the dynamic nature of the environment itself on the one hand and noise on the other. According to the derivation of octree, suppose $t=1, \ldots, T$ At time T, the observed data is $z_1, \ldots, z_T$, then the information recorded by the n leaf node is:

$$L(n|z_{1:t}) = L(n|z_{1:t-1}) + L(n|z_t) \tag{1}$$

With

$$L(n) = log[\frac{P(n)}{1-P(n)}] \tag{2}$$

Where n represents the leaf node, and z_t represents the measured value z_t .Therefore, $P(n|z_t)$ represents the probability of the voxel n being occupied at a given measurement value of z_t .

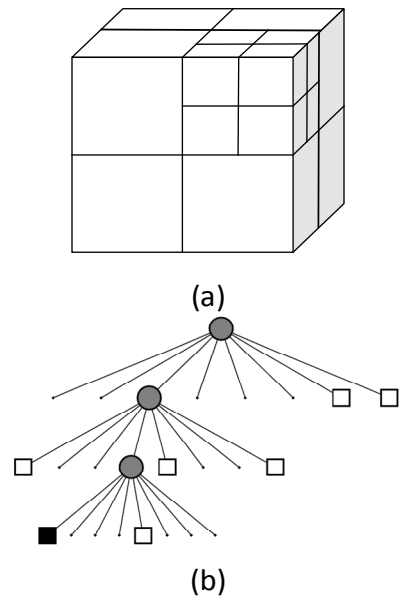

Fig 4 The structure of an octree. Fig.4(a) is the spatial representation of an octree map; Fig 4(b) is the data structure of octree map

C. Point Cloud Image Filtering

Point cloud diagram is the geometry of points in space. Since the point cloud is not a function, it is not defined by any rule or numerical relationship between x, y, and z for complex three-dimensional shapes. So the point cloud can't make the connection between the horizontal and the vertical. And the point cloud is discrete in space. Unlike images, signals are not defined in a certain region and cannot be filtered in the form of a template, so the point cloud does not have such an obvious definition domain as images and signals. At the same time, point clouds are widely distributed in space. The most difficult part is to go through each point in the whole point cloud and establish the relationship between the points. This is different from images and signals.

In summary, point cloud filtering is only similar to signal and image filtering in an abstract sense. Because the filtering function is to highlight the need for information. For the point cloud image, the filter can be used to eliminate outliers or gross errors caused by measurement errors. It works by making a statistical analysis of the neighborhood of each point and calculating its average distance to all neighboring points. If the result is a gaussian distribution whose shape is determined by the mean value and standard deviation, then points with an average distance outside the standard range (defined by the global distance mean value and variance) can be defined as outliers and removed from the data. First, the point cloud is traversed to calculate the average distance between each point and its nearest K neighbor points. Then, calculate the mean value μ and standard deviation σ of all mean distances, then the distance threshold $d_{\max}$ can be expressed as $d_{\max} = \mu + \alpha \times \sigma$, α is a constant Finally, the

point cloud is traversed again, eliminating points whose mean distance to K neighboring points is greater than $d_{\max}$.

III. EXPERIMENTAL TEST AND RESULT ANALYSIS

A. Run TUM Dataset to Get Data

The Technical University of Munich (TUM) is the standard data set of SLAM algorithm, which includes video images in different scenes, depth information, camera parameters, etc. It can support monocular, binocular and RGB-D camera simulation operation. This article simulates the room in the TUM data set. The data set room is a scene in which the camera moves 360° inside the room. The first is the sparse point cloud image generated by the original ORB-SLAM2.

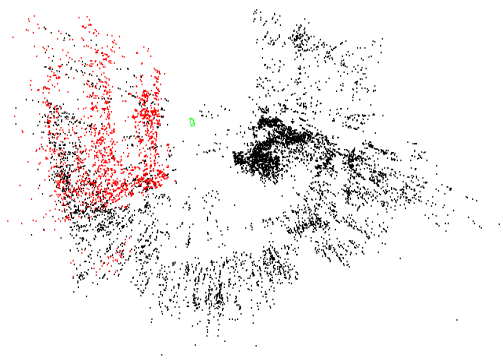

Fig 5 Sparse point cloud map

In Fig.5, the red dots represent the map points being observed by the camera, and the black dots represent the previously saved map points. As can be seen from the figure, the sparse point cloud map can only track the trajectory of the camera and cannot bring any effective environmental information and navigation information to the robot.

The Fig.6 shows the dense point cloud map generated by the improved ORB SLAM2 algorithm:

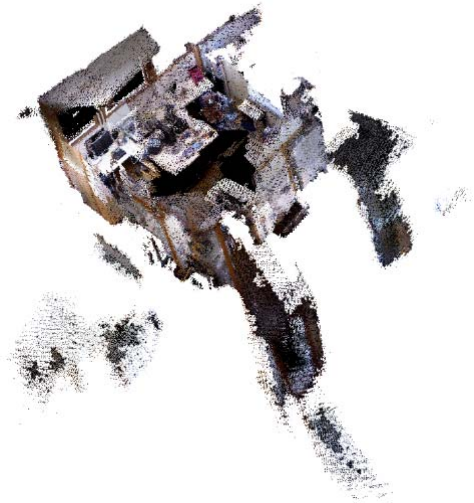

Fig 6 Dense point cloud map

As can be seen from the figure above, compared with the sparse point cloud map, the dense point cloud map can show the real environment, but due to the noise interference, there are many meaningless map points in the point cloud map. Therefore, filtering is required. The point cloud map after filtering is shown in Fig.7.

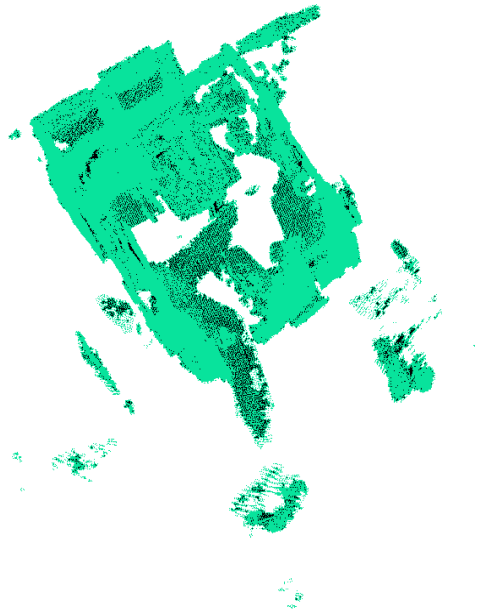

Fig 7 Point cloud map after filtering

It can be seen clearly that the discrete points outside the map are greatly reduced, and the information effectiveness of the filtered point cloud map is improved, which makes the subsequent octave map more concise and effective.

Fig.8 and 9 show the effect of the octree map generated by the original map and the filtered map:

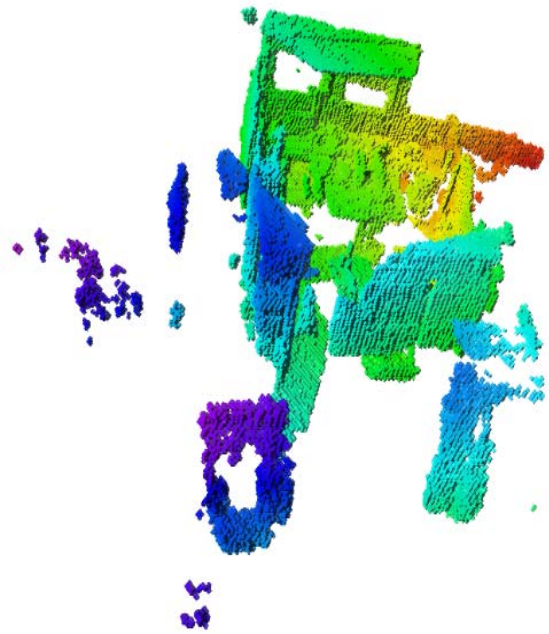

Fig 8 Unfiltered octree map

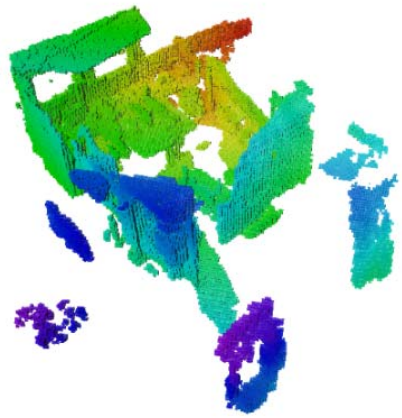

Fig 9 Octree map after filtering

It is obvious that there are fewer outliers on the west and south sides of the room, which allows the map to have less map data without losing existing information, making the file easier to store and calculate.

Table I is the comparison of the file sizes of each point cloud map and octree map:

TABLE I
FILE SIZE COMPARISON

Map Type / Attribute	Point cloud map	Octree map before filtering	Octree map after filtering
Points	39480	39480	36201
Size	6.3MB	50.1KB	48.1KB

B · Experiments with The Real Environment

We will place an RGB-D camera on the spherical underwater robot. Using raspberry PI 4B as a processor to run programs and process data. In this experiment, we use the Astra Pro depth camera. The size of it is 165×40×30mm and the maximum power is 2.5W via USB, which enables it to run on the robot for a long time. The image resolution of the camera was set to 640×480 and the frequency was set to 30 images per second in the experiment. Depth images have the same resolution as frequency and image acquisition. The robot moves around the box in a circle around the room so that the camera can capture parts of the room and map it. Fig.10 shows the indoor real environment, and Fig.11 shows the dense point cloud map obtained by the robot in the indoor environment:

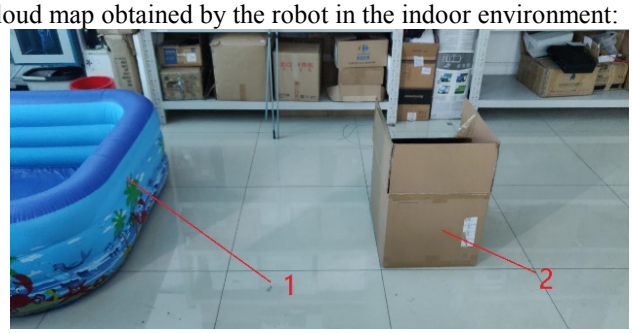

Fig.10 Indoor real environment

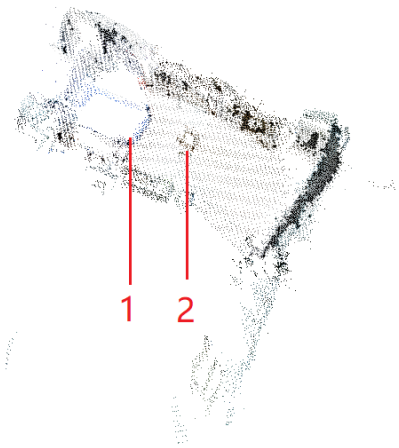

Fig 11 Indoor point cloud map

Where 1 is the indoor swimming pool and 2 is a box. We can see the RGB-D camera can correctly model the surrounding environment. However, due to noise interference and other factors, some useless map points appear on the point cloud map. After filtering, most invalid map points can be eliminated, as shown in Fig.12:

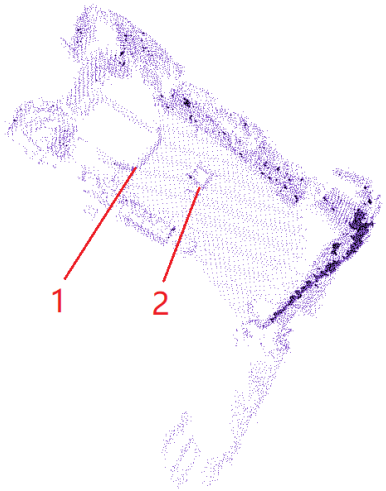

Fig.12 Indoor point cloud map after filtering

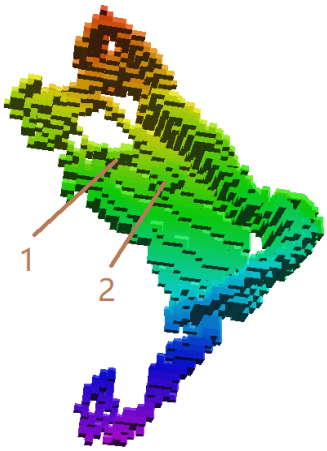

Fig.13 Octree map after filtering

After filtering, most invalid map points on the map are deleted, which will reduce the data processing and improve the map effect. Fig.13 is the transformation of the filtered dense point cloud map into an octree map. The error between the generated point cloud map and the actual environment is within 0.01m.

Table II is the comparison of the file sizes of each point cloud map and octree map:

TABLE II
FILE SIZE COMPARISON

Map Type / Attribute	Point cloud map	Octree map before filtering	Octree map after filtering
Points	21480	18320	17941
Size	4.3MB	35.1KB	32KB

As can be seen from Fig.13, the octree map quantifies the surrounding environment so that the robot can distinguish the height of the surrounding environment. At the same time, it can be seen that the map effect generated by the RGB-D camera on the robot is worse than that generated by the TUM data set, because the inevitable vibration of the walking robot in the process of moving leads to image blurring, resulting in errors.

V. CONCLUSIONS AND FUTURE WORK

In this paper, the image information is obtained by installing RGB-D camera on the spherical amphibious robot, and then the image information is converted into point cloud map. Since the point cloud map is bulky, noisy and cannot be used for navigation, we filter the point cloud map to remove the noise, and then convert the point cloud map into an octree map that is easy to store and navigate. From the data of TUM dataset and the results of actual camera operation, the filtered map has smaller volume and more effective information. However, due to the complexity of the algorithm, it is difficult to generate octree map in real time according to the point cloud map. The future work is to improve the algorithm to generate the map in real time and navigate according to the map.

ACKNOWLEDGMENTS

This research is supported by National Natural Science Foundation of China (61703305), Key Research Program of the Natural Science Foundation of Tianjin (18JCZDJC38500) and Innovative Cooperation Project of Tianjin Scientific and Technological (18PTZWHZ00090).

REFERENCES

[1] Locks Floriani, N. Palomeras, L. Weihmann, H. Simas, and P. Ridao, "Model-based underwater inspection via viewpoint planning using octomap,", pp. 1-8, March, 2017.

[2] M. Drwiega, "Efficient Integration of Octree Based Maps in Multi-Robot System,", pp. 487-492, 2018.

[3] M. Compagnoni, A. Pini, A. Canclini, P. Bestagini, F. Antonacci, S. Tubaro, and A. Sarti, "A Geometrical–Statistical Approach to Outlier Removal for TDOA Measurements," IEEE T. Signal Proces., 65, pp. 3960-3975, September 2017.

[4] Y. Yue, D. Wang, P. G. C. N. Senarathne, and D. Moratuwage, "A hybrid probabilistic and point set registration approach for fusion of 3D occupancy grid maps,", pp. 001975-001980, September 2016.

[5] H. Wang and Y. Liu, "A low-cost autonomous navigation system for a quadrotor in complex outdoor environments," International Journal of Advanced Robotic Systems, 17, p. 172988142090515, May 2020.

[6] J. Lunenburg, R. van de Molengraft and M. Steinbuch, "A representation method based on the probability of collision for safe robot navigation in domestic environments," Auton. Robot., 42, pp. 601-614, May 2018.

[7] Y. Li, F. Zhun, Z. Guijie, L. Wenji, L. Chong, W. Yupeng, and X. Honghui, "A SLAM with simultaneous construction of 2D and 3D maps based on Rao-Blackwellized particle filters,", pp. 374-378, 2018.

[8] E. Pedrosa, A. Pereira and N. Lau, "A sparse-dense approach for efficient grid mapping,", pp. 136-141, September 2018.

[9] J. Zhang, S. Liu, B. Gao, and C. Zhong, "An Improvement Algorithm for OctoMap Based on RGB-D SLAM,", pp. 5006-5011, 2018.

[10] R. Lavrenov, A. Zakiev and E. Magid, "Automatic mapping and filtering tool: From a sensor-based occupancy grid to a 3D Gazebo octomap,", pp. 190-195, March 2017.

[11] D. Yang, S. Bi, W. Wang, C. Yuan, W. Wang, X. Qi, and Y. Cai, "DRE-SLAM: Dynamic RGB-D Encoder SLAM for a Differential-Drive Robot," Remote Sens.-Basel, 11, p. 380, January 2019.

[12] J. Chen and S. Shen, "Improving octree-based occupancy maps using environment sparsity with application to aerial robot navigation,", pp. 3656-3663, September 2017.

[13] C. Zhao, L. Sun, Z. Yan, G. Neumann, T. Duckett, and R. Stolkin, "Learning Kalman Network: A deep monocular visual odometry for on-road driving," Robot. Auton. Syst., 121, p. 103234, March, 2019.

[14] K. Liu, Z. Fan, M. Liu, and S. Zhang, "Object-aware Semantic Mapping of Indoor Scenes using Octomap,", pp. 8671-8676, May 2019.

[15] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: an efficient probabilistic 3D mapping framework based on octrees," Auton. Robot., 34, pp. 189-206, March 2017.

[16] R. Mur-Artal, J. M. M. Montiel and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," IEEE T. Robot., 31, pp. 1147-1163, March 2015.

[17] H. L. G. W. Qiang LV, "ORB-SLAM-based Tracing and 3D Reconstruction for Robot using Kinect 2.0," in 2017 29th Chinese Control And Decision Conference (CCDC), January 2017.

[18] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," IEEE T. Robot., vol.33, pp. 1255-1262, September 2017.

[19] L. Sun, Z. Yan, A. Zaganidis, C. Zhao, and T. Duckett, "Recurrent-OctoMap: Learning State-based Map Refinement for Long-Term Semantic Mapping with 3D-Lidar Data," January 2018.

[20] L. Zhang, L. Wei, P. Shen, W. Wei, G. Zhu, and J. Song, "Semantic SLAM Based on Object Detection and Improved Octomap," IEEE Access, 6, pp. 75545-75559, March, 2018.

[21] Shuxiang Guo, Yanlin He, Liwei Shi, Shaowu Pan, Rui Xiao, Kun Tang, Ping Guo, " Modeling and experimental evaluation of an improved amphibious robot with compact structure," Robotics and Computer-Integrated Manufacturing, Vol.51, pp. 37-52, July 2018

[22] Shuoxin Gu, Shuxiang Guo, Liang Zheng, Ruochen An, "Turning Locomotion Analysis and Performance Evaluation for a Spherical Underwater Robot", IEEE International Conference on Mechatronics and Automation, pp. 2507-2511, July 2019

[23] Jian Guo, Chunying Li, Shuxiang Guo, "Path Optimization Method for the Spherical Underwater Robot in Unknown Environment", Journal of Bionic Engineering, vol.15, pp.1-30, July 2018

[24] Shuoxin Gu, Shuxiang Guo, Liang Zheng. "A highly stable and efficient spherical underwater robot with hybrid propulsion devices", Autonomous Robots, vol.61, pp. 122-215, May 2019

[25] Jian Guo, Chunying Li, Shuxiang Guo, "A Novel Step Optimal Path Planning Algorithm for the Spherical Mobile Robot Based on Fuzzy Control", IEEE Access, vol. 8, pp. 1394-1405, May 2020