Robust SLAM based on Segmentation of Dynamic Object's Point Cloud

○ Sun Leyuan, Student Non-member, University of Tsukuba, AIST, leyuansun@outlook.com
Fumio Kanehiro, The Robotics Society of Japan, AIST

At present, the mainstream Simultaneous Localization and Mapping(SLAM) system is mainly suitable for static scenes. However, in real life, dynamic objects in an unknown environment limit the application of the SLAM system. Using the traditional SLAM scheme, if the moving object exists in the sensor field of view, the constructed 3D point cloud map leaves the point cloud of the moving target multiple times, and affects the accuracy of the visual odometry(VO) and closure loop detection. This paper proposes a scheme for SLAM in dynamic environment that can be used to identify dynamic objects and remove its point clouds. The robustness of the system in dynamic environment is verified by the outliers of feature points matching and the point cloud map.

***Key Words*:** Dynamic Object Detection, Point Cloud Segmentation, Feature Points Matching, Robust SLAM

1. Introduction

The process of acquiring external environmental information through sensors in an unknown environment, realizing pose estimation and incrementally constructing an environmental model, and then establishing its own global position is called SLAM. The most critical part of the entire SLAM system is the relative displacement of the two frames based on the position of the associated pixel between two adjacent frames. However, this calculation of the visual odometry is based on a strict condition: the position of the three-dimensional space point of the associated pixel used for the calculation is constant. If the pixels in the field of view are constantly moving such as the all pixels on the human body in Fig.1, and at the same time, these points participate in the pose calculation, then these points will continue to bring errors to the system, eventually leading to VO failure.

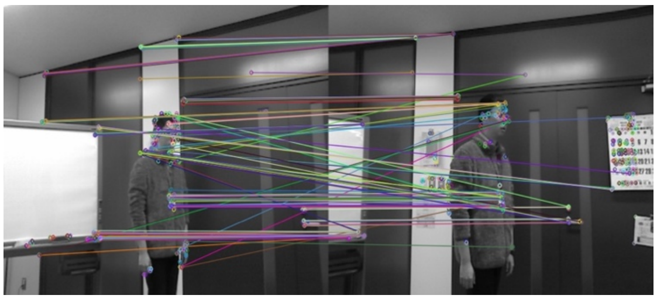


Fig. 1: ORB feature points matching in dynamic environment

Raluca Scona and Mariano Jaimez proposed the StaticFusion[1] to deal with dynamic objects in SLAM system, the point clouds were divided into dynamic foregrounds and static background, only the static background cluster was sent to the back-end of SLAM. The dynamic object in the first frame image of the SF cannot be more than 30% of the image to initialize the map. If it is bigger than it, it becomes difficult to recognize it in the first frame image. If the background object that has been considered to be static is subsequently moved, the algorithm is difficult to detect them. In contrast, the algorithm in this paper does not require the proportion of dynamic objects, and the dynamic objects in the first frame can be identified.

With regard to the humanoid robot's SLAM in dynamic environment, Tianwei Zhang and Emiko Uchiyama proposed the PoseFusion[2] SLAM to deal with the human moving in the static situation, which was based on the deep learning to detect the human joints pose and segment out its point clouds to build the map. The difference with deep learning method to do dynamic object segmentation is that the proposed method does not require the data training, and as long as the depth value of the moving object in the camera view of sight changes, not only the human in Fig.2 but also any kind of dynamic object such as Fig.3 waving tennis racket can be detected.

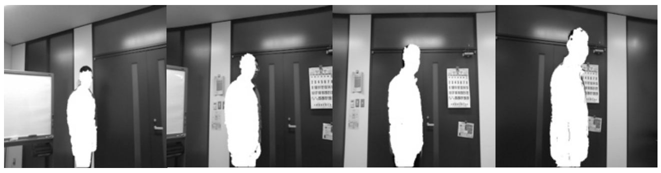


Fig. 2: Continuous walking human segmentation

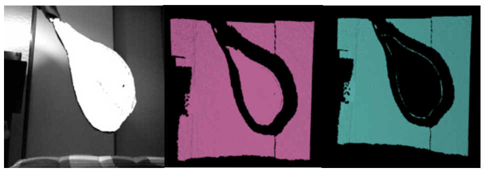


Fig. 3: Point cloud of waving tennis racket segmentation(each picture represents: image after segmentation, original point cloud, point cloud after segmentation)

2. Overview of the Proposed Method

In this report, a detection method based on RGB image in Fig.4 box.1 and depth image in Fig.4 box.5 is proposed. The RGB and depth images are acquired by Kinetic RGB-D camera. As you can see in the flow chart Fig.4, the homography matrix(box.3) between two continuous frames is solved by image matching(box.2), which represents mapping relationship between them. In the binarized differential image(box.4), which combines the depth information(box.5), we can separate the moving object into last and current frame(box.6), which means the image marked with the region in box.7 is obtained.

For dealing with the point cloud, this report uses the marked image(box.7) to do the point cloud clustering segmentation. Combined with the camera model, the 3D point cloud(box.8) is calculated after the camera is calibrated, and filtered by StatisticalOutlierRemoval[3] and downsampled(box.9) to reduce the consumption of computing resources. The clustering algorithm(box.10) is used to segment the point cloud into multiple clusters. After each cluster projected to plane(box.11) , then compared with the moving target region(box.7) to find which the cluster is generated by dynamic object. In the process of removing the point cloud, the cluster point cloud is processed using the boundary extraction algorithm and the prism extraction algorithm(box.12). In the end, we could obtain the static object's RGB image and point cloud(box.13).

In the Fig. 7, the white area is the pixel of the moving object, and the black part represents the static area. However, this image contains the moving objects of the two frames.

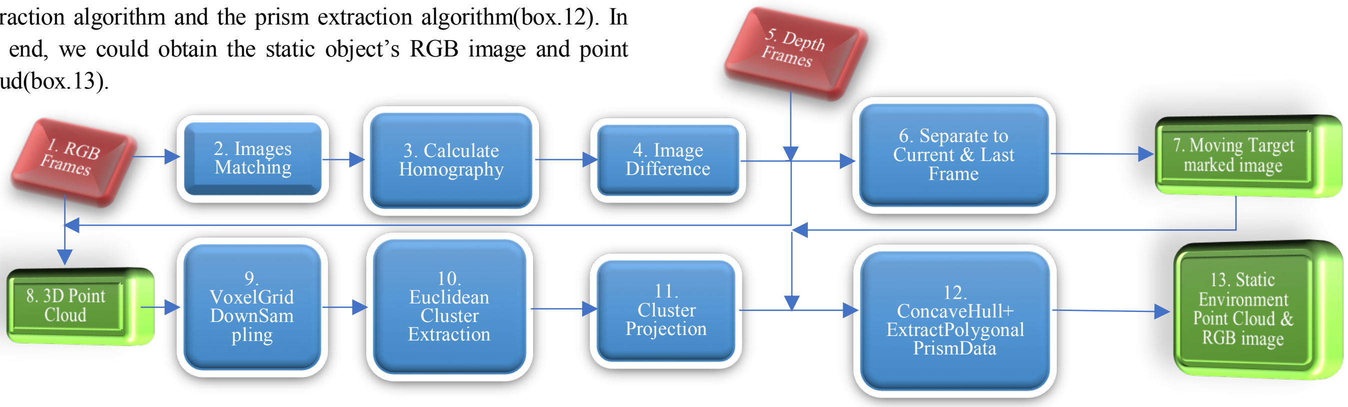
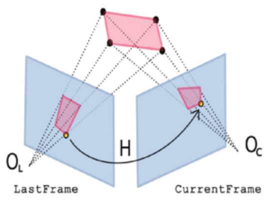

1. RGB Frames
2. Images Matching
3. Calculate Homography
4. Image Difference
5. Depth Frames
6. Separate to Current & Last Frame
7. Moving Target marked image
8. 3D Point Cloud
9. VoxelGrid DownSampling
10. Euclidean Cluster Extraction
11. Cluster Projection
12. ConcaveHull+ ExtractPolygonal PrismData
13. Static Environment Point Cloud & RGB image

Fig. 4: Flow chart of the proposed method(red boxes 1,5 are the input, green boxes 7,8,13 are the result images and point cloud, the rest blue boxes are image processing)

3. Dynamic Object Filter

3.1 Dynamic pixels identification and separation

Since the Kinect sensor collects information about the unknown environment during motion, we need to find the projection mapping relationship such as Eq. (1) between the image plane of the previous frame and the image plane of the current frame, that is, calculate the homography matrix in Fig.5.



$$\begin{pmatrix} x_C \\ y_C \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_L \\ y_L \\ 1 \end{pmatrix} \quad (1)$$

Fig. 5: Homography matrix

After the two frames(the first line of Fig.6) are smoothed by grayscale and Gaussian filtering, the correspondence between the two frames is found by using the homography matrix. Then all the pixels are traversed in order to calculate the difference and compare with $threshold1$ in Eq. (2) (I_L is the LastImage, I_C is the CurrentImage) to obtain the the first image in Fig.7.

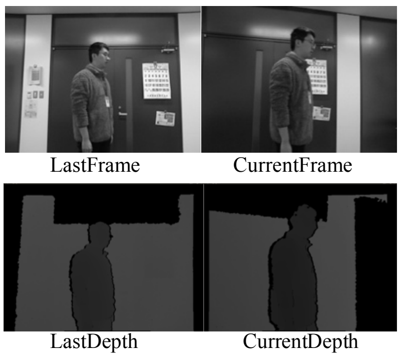


Fig. 6 Input images

$$f(x,y) = |I_L(x_L, y_L) - I_C(x_C, y_C)|$$

$$f(x,y) = \begin{cases} 255, & f(x,y) > threshold1 \\ 0, & f(x,y) \le threshold1 \end{cases} \quad (2)$$

In order to separate them into the second and third image in Fig.7, the proposed algorithm uses the change of the depth values of the corresponding coordinates to compare with the $threshold2$ in Eq. (3) (d_L represents the depth in LastImage). After the separation, traversing all white pixels' coordinates to get the maximum and minimum range of x, y and depth such as the rectangle boundary in the third image of Fig.7.

$$depth(x,y) = |d_L(x_L, y_L) - d_C(x_C, y_C)|$$

$$\begin{cases} (x,y) \in CurrentFrame, depth(x,y) > threshold2 \\ (x,y) \in LastFrame,\ depth(x,y) \le threshold2 \end{cases} \quad (3)$$

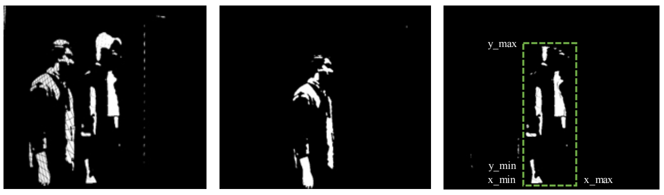


Fig. 7 Dynamic binary pixels separation

The dynamic pixels detection and separate them into Last and Current Frame' s algorithm is as bellowed:

```
Algorithm about how to detect and separate the dynamic object
Known: LastImage,CurrentImage,LastDepth,CurrentDepth,
threshold1,threshold2
1:for each row ∈ [0,LastImage.rows - 1] do
2:    for each col ∈ [0,LastImage.cols - 1] do
3:        xL ← row, yL ← col, z ← 1
4:        x' ← h11 * xL + h12 * yL + h13
5:        y' ← h21 * xL + h22 * yL + h23
6:        z ← h31 * xL + h32 * yL + h33
7:        diff ← abs(Pl(xL,yL) - Pc(xc,yc))
8:            for each row ∈ [0,diff.rows - 1] do
9:                for each col ∈ [0,diff.cols - 1] do
10:                   if diff > threshold1
11:                       x,y ∈ diff ← 255
12:                   else if
13:                       x,y ∈ diff ← 0
14:                   end if
15:                   if dL - dc > T2 &&  (x,y ∈ diff ← 255) then
16:                       LastFrame(x,y) ← 0
17:                       CurrentFrame(x,y)  ← 255
18:                   else if dc - dL > threshold2 && (x,y ∈ diff ← 255)
                          then
```

```
19:                LastFrame(x,y) ← 255
20:                CurrentFrame(x,y)  ← 0
21:            else if
22:                LastFrame(x,y) ← 0
23:                CurrentFrmae(x,y) ← 0
24:            end if
25:        end for
26:      end for
27:   end for
28:end for
```

3.2 Dynamic Object Point Cloud Clusters Selection

Using VoxelGrid[4], under the premise of ensuring the shape of the point cloud, the number will be downsampled to ensure the efficiency of the algorithm. Before cloud clustering[5], since both moving objects and static objects are above the ground plane, depending on the scene, it is sometimes necessary to remove the ground planar[6] in order to achieve a better clustering effect. Such as in Fig. 8, the downsampled points cloud contains 1 dynamic cluster(human) and 2 static cluster(walls). After clustered, the boundary of each cluster is obtained by the projection formula of the camera pinhole model, and compared with the range of the dynamic object pixel points in the previous step, finally which cluster belongs to dynamic objects can be decided.

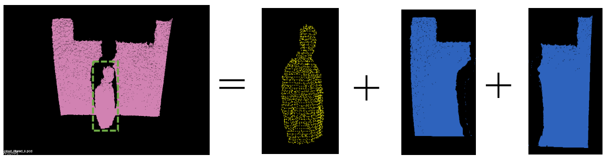


Fig. 8 Downsampled point clouds including 3 clusters

The dynamic cluster judgement algorithm is as bellowed:

```
Algorithm about which cluster is belong to the dynamic object
Known: count is the number of cluster, fx, fy, cx, cy are the camera model,
threshold3
1:for each i ∈ [0, count - 1] do
2:    for each j ∈ [0,cluster.size-1] do
3:        u0 ← (x' * fx)/z' + cx
4:        v0 ← (y' * fy)/z' + cy
5:        Boundary ← (u_min,u_max,v_min,v_max,
                      depth_min,depth_max)
6:    end for
7:end for
8:point_num ← 0
9:for each row ∈ [0,CurrentImage.rows - 1] do
10:    for each col ∈ [0,CurrentImage.cols - 1] do
11:        if CurrentImage(row,col) == 255 then
12:            if row > u_min a && row < u_max && col > v_min &&
               col <v_max then
13:                if cluster(row,col)depth_min && cluster(row,col) <
                   depth_max then
14:                    num = num + 1
15:                end if
16:            end if
17:        end if
18:    end for
19:end for
20:if num > threshold3 then
21:return true
22:end if
```

3.3 Dynamic Object Point Cloud Clusters Segmentation

Once we have determined the dynamic cluster such as Fig.11, we need to remove it in the original point cloud instead of the downsampled one. First, we perform a concave hull algorithm[8] on the projection of downsampled dynamic object cluster, in order to obtain its boundary, such as the continuous blue points in Fig. 9. By stretching the concave hull in the original point cloud, the dynamic object point cloud in the original point cloud is included in the concave polygonal prism[8]. For example, stretching the concave hull to form a polygonal prism, the red lines in Fig. 9 represents the stretch direction, as we can see, the original human point cloud is inside the prism. After removing them, the remaining point cloud is static scenes like Fig.10. In the end, projecting it to the 2D plane, and set the removed pixel value to 0.

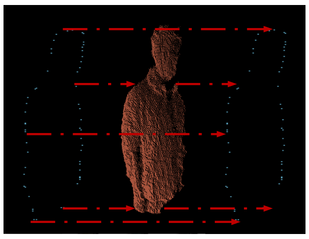


Fig. 9 Original point cloud inside the polygonal prism

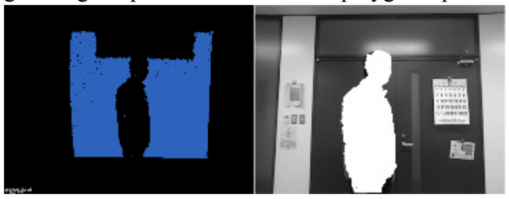


Fig. 10 Static scene point cloud and RGB image

4. Experiment Results

The dynamic filter that we propose is used to send filtered information to RTAB[9] SLAM's mapping node. At the same time, the matching of the feature points between original–original, filtered-filtered have been test in Fig. 14, 15. However, in the both filtered situation, the white pixels around the boundary of dynamic object were very similar when detect ORB feature which produces large outliers in matching Fig. 15.

In order to solve this problem, we use the dynamic object position information to make a bool type matrix mask. The details are in the Fig. 11. First step is traversing all the pixels in order to segment white pixels and non-white pixels, then extract contours. Using the convex hull to make the mask, in this mask matrix, the black represents 0 and the others represents 1, the region of 1 will be considered as the ROI(region of interest) in Fig. 12. So the mask has been added to the filtered image, the ORB feature points has been detected only in the ROI. The left image in Fig.13 is without the ROI mask, the right is after the mask.

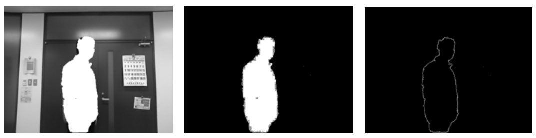


Fig. 11 Dynamic Object Region Mask

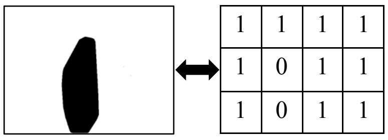


Fig. 12 Bool Type Matrix

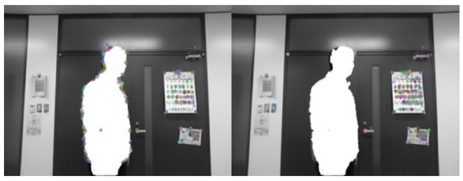


Fig. 13 Detect ORB with ROI and without ROI

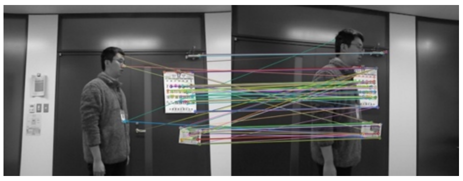


Fig. 14 Matching in original image

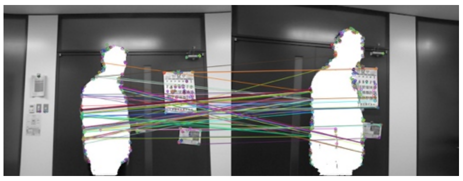


Fig. 15 Matching without ROI mask

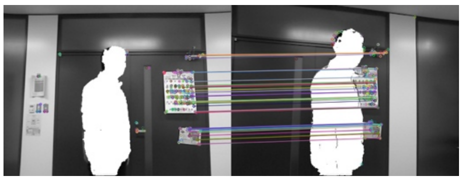


Fig. 16 Matching with ROI mask

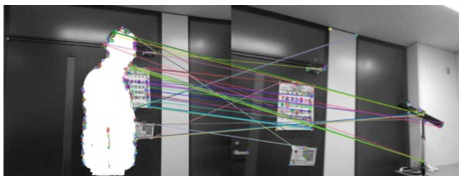


Fig. 17 Matching with static scene(without mask)

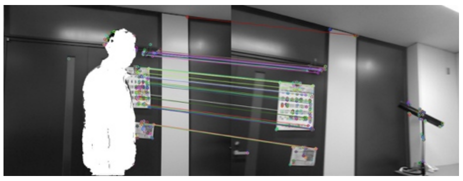


Fig. 18 Matching with static scene(with mask)

As we can see, the filtered image with mask has the better performance in reducing the outliers in Fig. 14-16. Besides, the Fig. 17,18 represent the last frame with dynamic scene, but the dynamic object disappeared in the current frame. With the mask, the outliers are less in the comparison. All the matchings are under the same Hamming distance constrain condition.

The following figures are the point cloud map and Octomap comparison in Fig. 19,20. Inside the red line, there are the dynamic object parts. Both of right figures are after the dynamic filter effect.

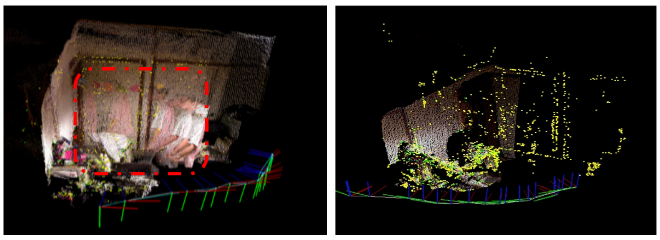


Fig. 19 Point cloud map comparison

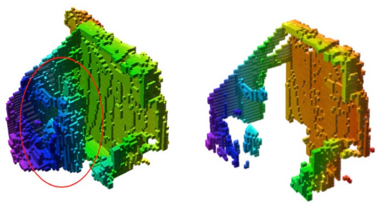


Fig. 20 Octomap comparison

Table 1: Thresholds list and time consuming.

Name	Threshold	Name	Time
Image differential binarization	T1=40	Calculate homography matrix	0.067182s
Depth threshold	T2=0.2m	Use depth information to construct point cloud	0.029468s
Point cloud statistical outliers removal filter	T3=0.06m	Image differential binarization to find out dynamic pixels	0.714532s
Point cloud clustering distance threshold	T4=0.1m	Downsampling point cloud	0.026421s
Downsampled voxel grid resolution	T5=0.02m	The removal of ground plane	0.05677s
Pointing cluster boundary extraction angle	T6=8 degree	The removal of dynamic object point cloud cluster	0.454950s
3D moving target detection point number threshold	T7=3000	Total time	1.34933s
The RTAB system adds the key-frame each second, so in general, add the 2 keyframes need 2 seconds, and calculate the 2 frames need 1.3 seconds) The running virtual machine information: ROM:32GB CPU: Intel® Core™ i7-4930K CPU @ 3.40GHz × 12			

5. Conclusions

The dynamic object detection and point cloud removal algorithms proposed in this paper have better effects when the depth is constantly changing. Table 1 is the time consuming of each part. But it is sensitive to a large number of manually set thresholds in Table 1. In addition, the point cloud map reconstruction which is blocked by dynamic objects is sparse in Fig. 18. In the future, the quantified VO results are necessary for explaining the dynamic filter performance in SLAM.

REFERENCES

[1] R. Scona, M. Jaimez, Y. R. Petillot, M. Fallon, and D. Cremers, “Staticfusion: Background reconstruction for dense rgb-d slam in dynamic environments.” Institute of Electrical and Electronics Engineers, 2018.

[2] T. Zhang and Y. Nakamura, “Posefusion: RGB-D SLAM in dynamic human environment,” 2018, international Symposium on Experimental Robotics.

[3] http://pointclouds.org/documentation/tutorials/statistical_outlier.php#statistical-outlier-removal

[4] http://wiki.ros.org/pcl_ros/Tutorials/VoxelGrid%20filtering

[5] http://pointclouds.org/documentation/tutorials/cluster_extraction.php#cluster-extraction

[6] http://wiki.ros.org/pcl_ros/Tutorials/SACSegmentationFromNormals%20planar%20segmentation

[7] http://pointclouds.org/documentation/tutorials/hull_2d.php#hull-2d

[8] http://wiki.ros.org/pcl_ros/Tutorials/ExtractPolygonalPrismData%20segmentation

[9] Mathieu Labbé ,François Michaud, “Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation”, IEEE Transactions on Robotics, June 2013.