Dense RGB-D SLAM for Humanoid Robots in the Dynamic Humans Environment

Tianwei Zhang , Emiko Uchiyama , and Yoshihiko Nakamura

SLAM Problems for Humanoid Robot

Dynamic environment:

1. Humans are often considered as moving obstacles in the humanoids working spaces, which results the dynamic environment problem.
2. The disturbances caused by the executions of biped locomotion and falling down

Previous works

1. SLAM for Humanoids

   A recent work of Scona fusing the Valkyrie humanoid motion information into EF framework, which achieves local loop closing in slightly dynamic environments.

   Problem: It cannot deal with full dynamic unknown environment

Previous works

2. Dynamic SLAM Methods

    ElasticFusion(EF):It can handle the small-scale environment changing since it benefits from the deformation graph based non-rigid module fusion. [deformation graph: use a part of points to represent whole points. Like max pooling in deep learning]

    Co-Fusion(CF): It’s based on EF. At first CF has to first reconstruct the map in a static environment, and then they enable the dynamic object detection and tracking abilities within that reconstructed map

    StaticFusion (SF): segmenting point clouds into a number of clusters to divide static and moving objects and only use static backgrounds.

POSEFUSION FOR HUMANOID

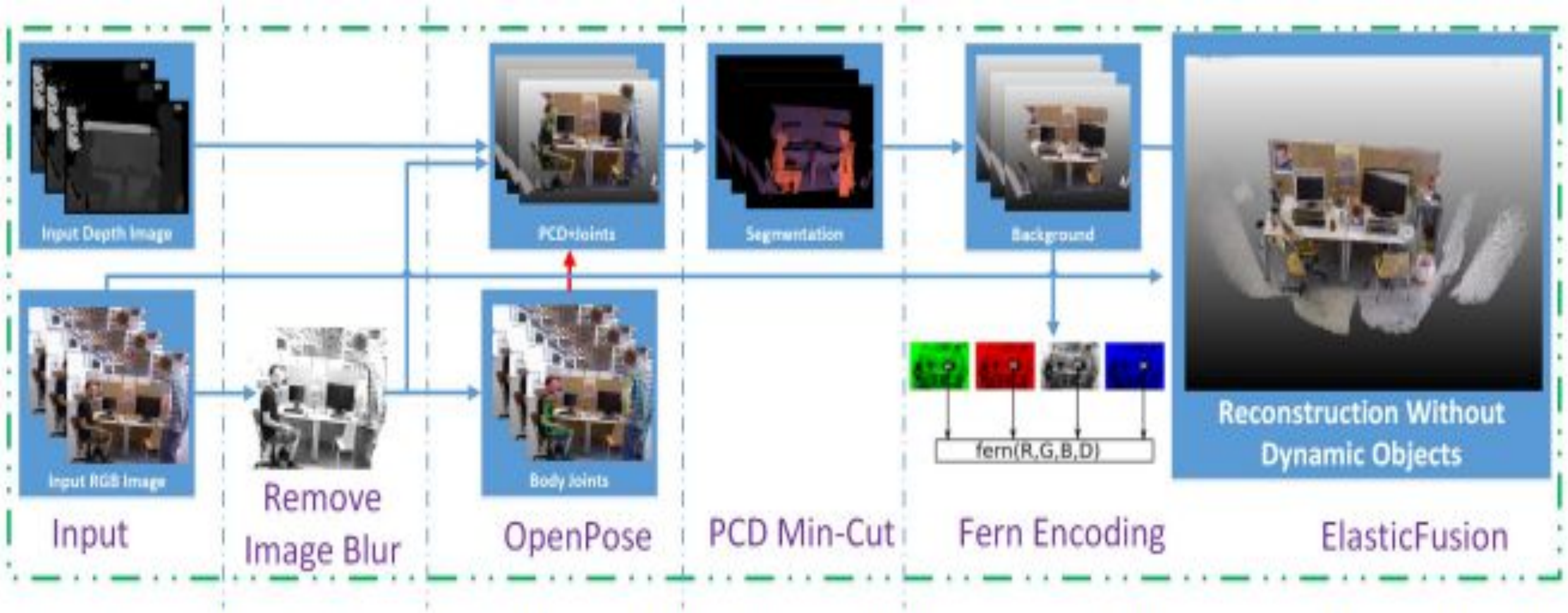


Fig. 3: Dynamic Humans Detection and Removal: PFH firstly input RGB images to OpenPose to detect human joints. Then we project joint points to the PCD and followed by foreground removal. Finally, static backgrounds are reconstructed.

POSEFUSION FOR HUMANOID

1. Remove the Image Blur From Robot Motion Shaking

Estimate the blur score by Laplacian method for each RGB input image and remove the image pairs with serious motion distortion. The image blur level can be reflected by the image Laplace variations. The Threshold is experimentally adjusted

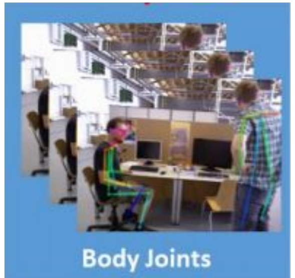
From http://www.aishack.in/tutorials/sobel-laplacian-edge-detectors/

POSEFUSION FOR HUMANOID

2. Use OpenPose to Detect Humans' joint

When the input image f is given, the feature map is extracted via CNN network and then output data: $[h \times w]_f$

in which h is the detected human bodies, at a maximum 15, which means OpenPose can detect at maximum 15 humans within one frame. The w is the list of estimated joints, it presents a probability map on the RGB image plane which indicates the existence likelihood of at maximum 18 human joint.



POSEFUSION FOR HUMANOID

3. Use Min-cut to remove dynamic object

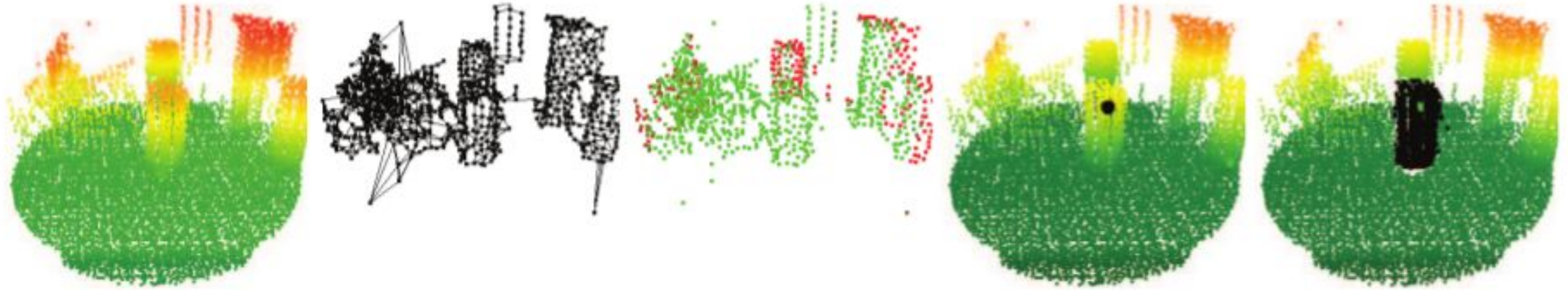
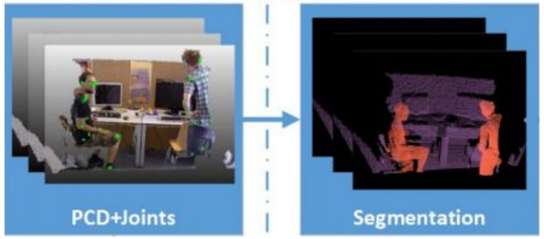

(a) Input (b) Graph (c) Background Penalty (d) Foreground Constraint (e) Result

Figure 2. Overview of our system. (a) The system takes as input a point cloud near an object location (in this case, a short post). (b) A k-nearest neighbors graph is constructed. (c) Each node has a background penalty function, increasing from the input location to the background radius (visualized with color turning from green to red as the value increases). (d) In the automatic version of our algorithm, a foreground point is chosen as a hard constraint (in the interactive mode, the user chooses hard foreground and background constraints). The resulting segmentation is created via a min-cut (e).

From paper: Min-Cut Based Segmentation of Point Clouds

POSEFUSION FOR HUMANOID

3. Use Min-cut to remove dynamic object



1. Choose joint point as foreground point location.
2. Connect k-nearest neighbors to construct a node graph.Each point cloud is a vertex and each two points will be connected with edge with cost C and background penalty P;
3. Cut the graph with a threshold

σ is a user defined parameter; (x, y) is neighbor point of Joint (Jx, Jy, Jz)

$$C = e^{-\left(\frac{len}{\sigma}\right)^2} \qquad P = \frac{\sqrt{(x-J_x)^2+(y-J_y)^2}}{r}$$

POSEFUSION FOR HUMANOID

4. Relocalization by Random Ferns

The underlying concept of relocalization approach is based on the ability of compact code generation for camera frames and efficient evaluation of code similarities between different frames

Each frame has a code block instead of a feature bag so the idea is to store compact codes which allow efficient keypoint recognition instead of encoding whole image frames. At test time, the conservatory of ferns is utilized as a classifier in order to find putative matches between incoming frames and a learned keypoint database

POSEFUSION FOR HUMANOID

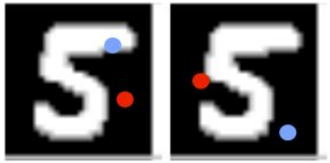
4. Relocalization by Random Ferns

- A Fern applies a series of S binary tests to the input vector **I**

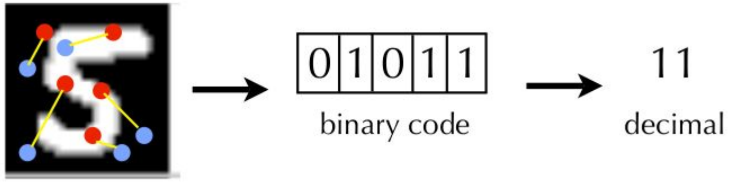
  e.g relative intensities of a pair of pixels:

  $f_1(\mathbf{I}) = I(x_a, y_a) > I(x_b, y_b)$ -> true

  $f_2(\mathbf{I}) = I(x_c, y_c) > I(x_d, y_d)$ -> false



- This gives an S-digit binary code for the feature which may be interpreted as an integer in the range [0 … 2^S-1]

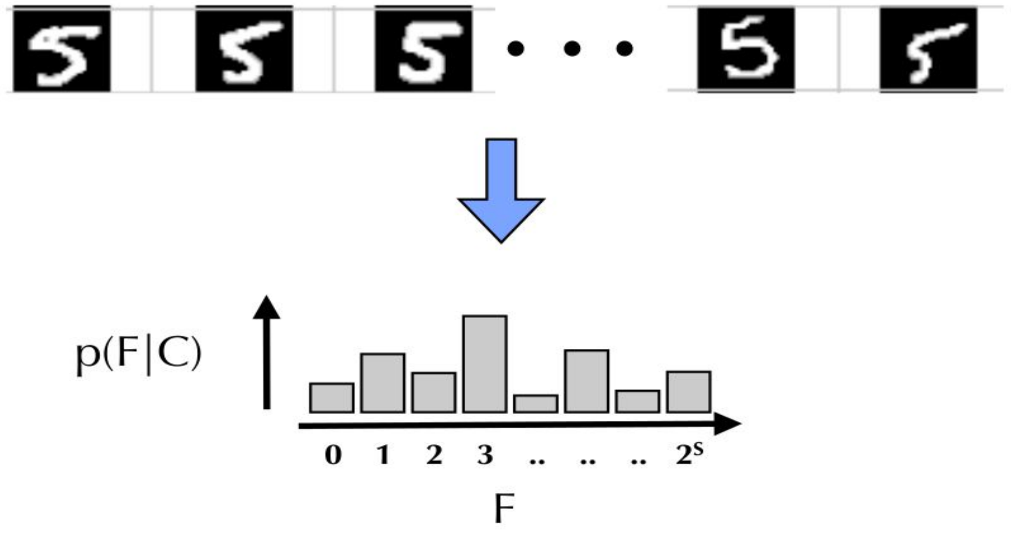


Use binary code instead of features to represent an image and get a score for each image

From [Random Forests and Ferns]

POSEFUSION FOR HUMANOID

4. Relocalization by Random Ferns

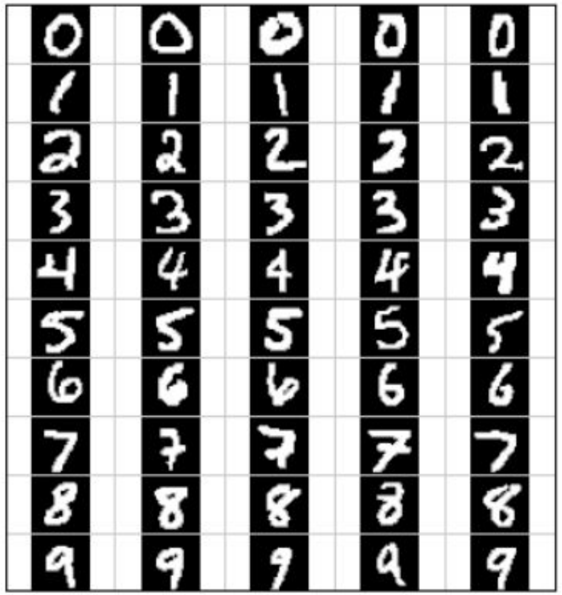


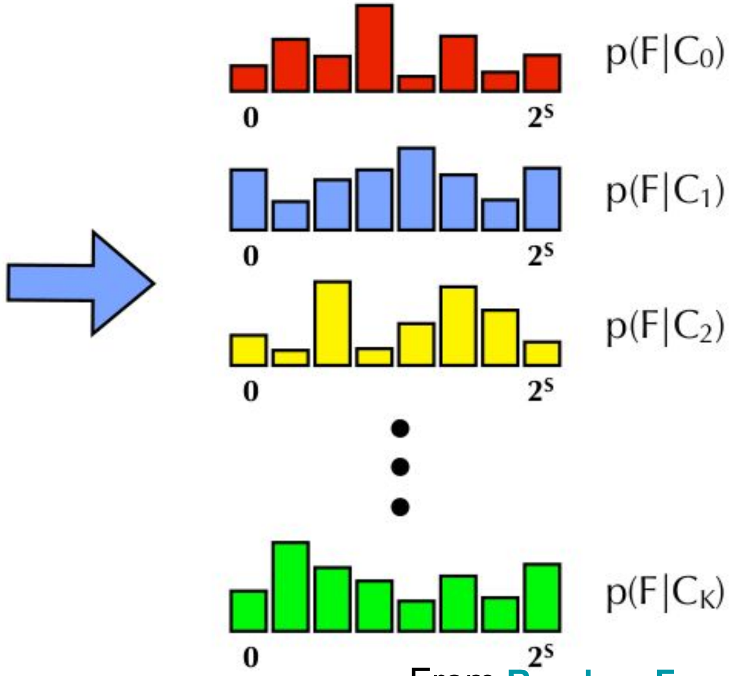
For each class we can generate a distribution

From Random Forests and Ferns

POSEFUSION FOR HUMANOID

4. Relocalization by Random Ferns

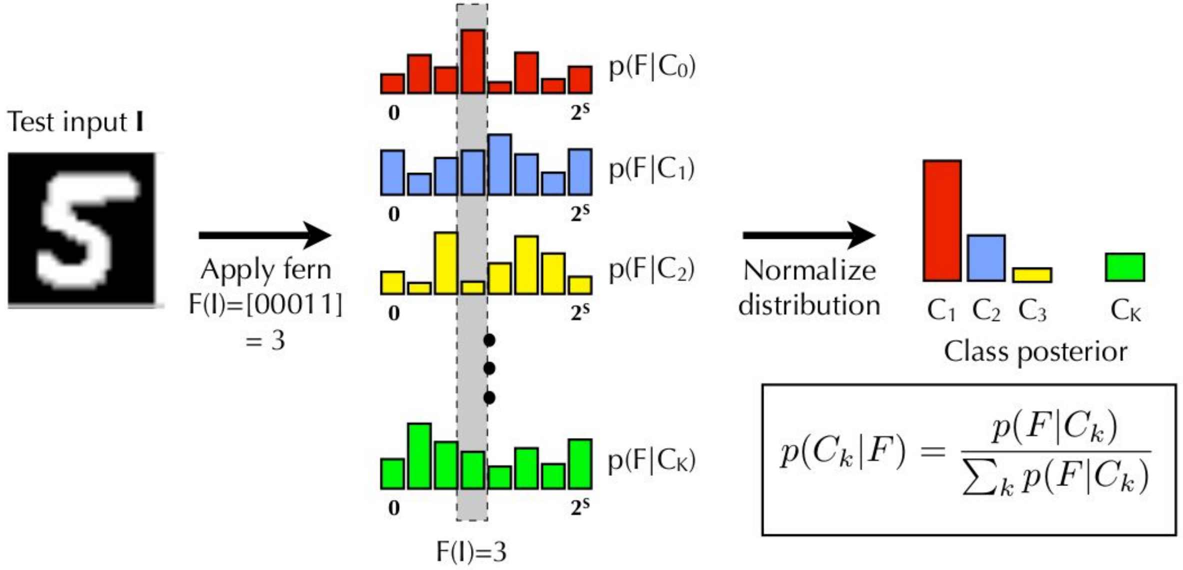




From Random Forests and Ferns

POSEFUSION FOR HUMANOID

4. Relocalization by Random Ferns

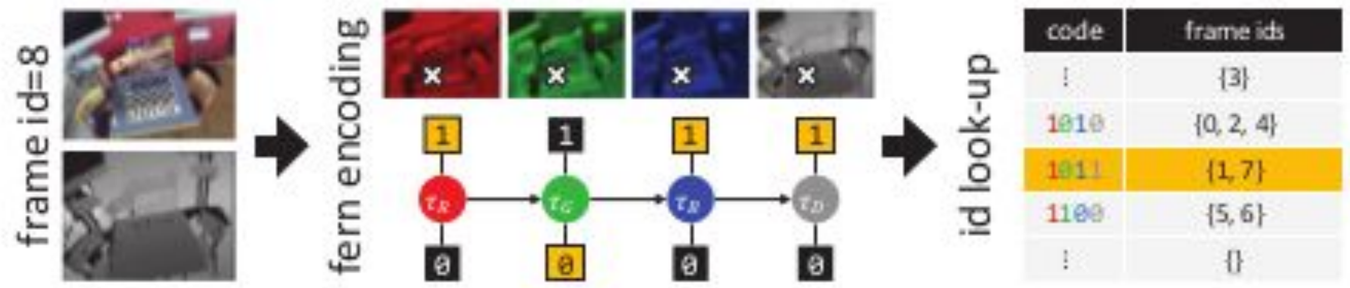


$$p(C_k|F) = \frac{p(F|C_k)}{\sum_k p(F|C_k)}$$

From Random Forests and Ferns

POSEFUSION FOR HUMANOID

4. Relocalization by Random Ferns



code	frame ids
⋮	{3}
1010	{0, 2, 4}
1011	{1, 7}
1100	{5, 6}
⋮	{}

From paper: Real-Time RGB-D Camera Relocalization

POSEFUSION FOR HUMANOID

5. ElasticFusion(EF)

a. Calculate pose based on RGB-D images with ICP algorithm.
b. If the error in ICP algorithm exceeds threshold, use relocalization. Otherwise, go ahead.
c. Utilize random ferns as global loop closure detection

ICP: for each point in mesh A, find the closest point in mesh b. Then calculate the R and T for each pair and repeatedly apply R and T to all point in A to do transformation so that we can find the best R and T

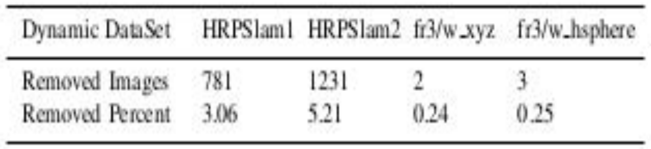
Experiment Results

Comparison in different level dynamic dataset:

1. **HrpSlam1**: robot walks around a chair and a table and try to reconstruct these obstacles within one loop.
2. **HrpSlam2**: robot walks along the walls of the room and trying to reconstructed the whole room with one loop closure.
3. **fr3/walk_xyz**: Two persons walk through an office scene. The Asus Xtion sensor has manually been moved along three directions (xyz) while keeping the same orientation.
4. **fr3/walk_hsphere**:Two persons walk through an office scene. The Asus Xtion sensor has been moved on a small half sphere of approximately one meter diameter.

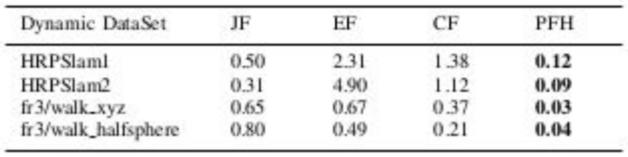
Experiment Results

TABLE I: Removed Blurred Images Numbers



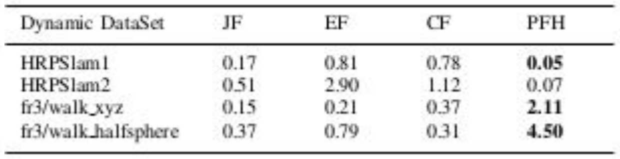
Dynamic DataSet	HRPSlam1	HRPSlam2	fr3/w_xyz	fr3/w_hsphere
Removed Images	781	1231	2	3
Removed Percent	3.06	5.21	0.24	0.25

TABLE II: Translate ATE RMSE (m)



Dynamic DataSet	JF	EF	CF	PFH
HRPSlam1	0.50	2.31	1.38	**0.12**
HRPSlam2	0.31	4.90	1.12	**0.09**
fr3/walk_xyz	0.65	0.67	0.37	**0.03**
fr3/walk_halfsphere	0.80	0.49	0.21	**0.04**

TABLE III: Translate RPE RMSE (m/s)



Dynamic DataSet	JF	EF	CF	PFH
HRPSlam1	0.17	0.81	0.78	**0.05**
HRPSlam2	0.51	2.90	1.12	0.07
fr3/walk_xyz	0.15	0.21	0.37	**2.11**
fr3/walk_halfsphere	0.37	0.79	0.31	**4.50**

ATE :directly measures the difference between points of the ground truth and the estimated trajectory.

RPE: computes the error in the relative motion between pairs of timestamps

Experiment Results

Relative static dataset test result:

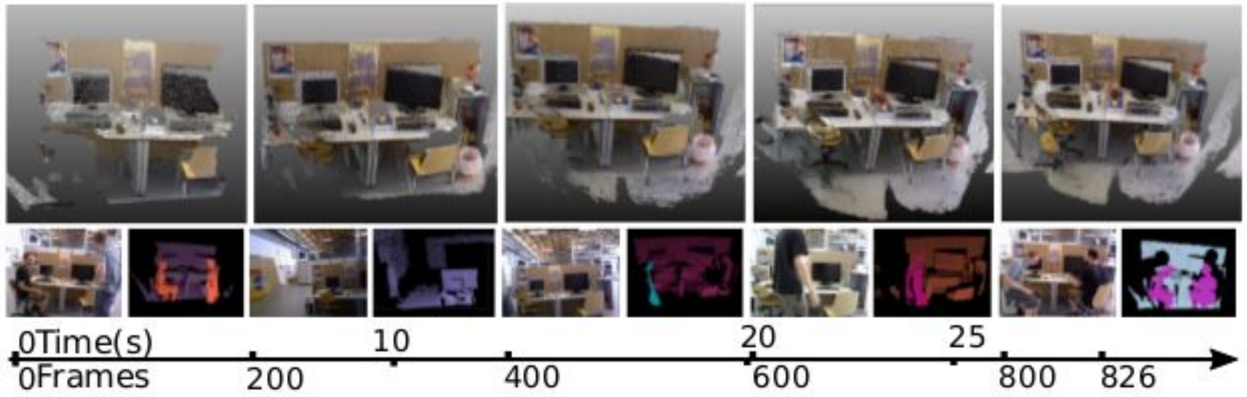


Fig. 5: Experiment result of fr3/walking from TUM benchmark [4]. The first row shows the reconstructed maps, the second row shows the respectively RGB input and foreground-background segmentations. As the frame grows, the first row reconstruction is gradually completed, the RGB viewpoint moves as the camera moves, the point clouds segmentation is also changing. There is no foreground cluster in the 4th image, the second row, since the guy walks out of the scene.

Experiment Results

Relative dynamic dataset test result:

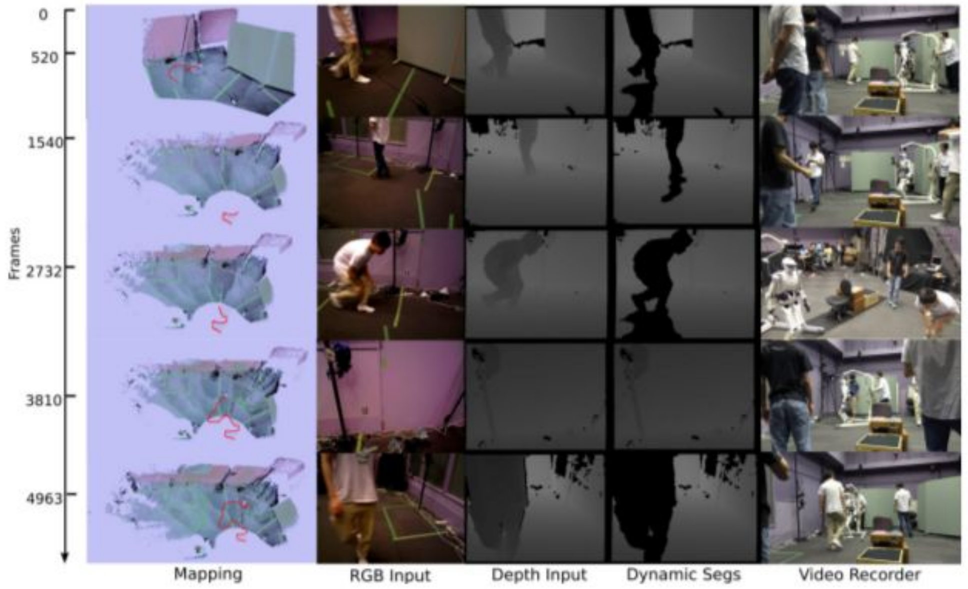
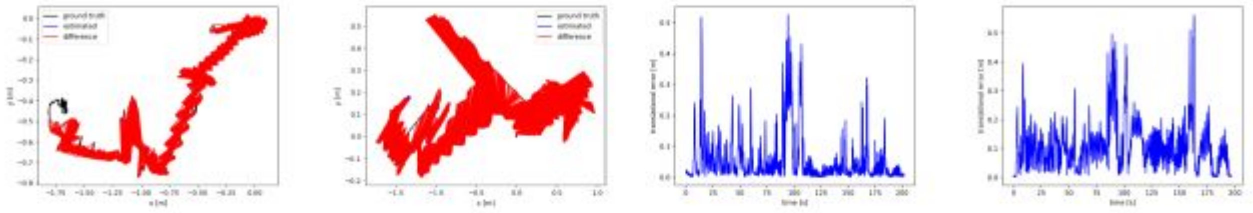


Fig. 7: The proposed method working on HrpSlam2. Top to bottom, camera frame from 0 to 6000 (30 FPS). From left to right: real time mapping result, robot RGB input image, depth input image, dynamic object segmentation (colored in pure black) outputs, and the instant video screen shot. The Leftest reconstructed maps which include no moving objects, and keeps a right camera localization and tracking

Experiment Results



(a) PFH Reconstruction ATE (b) JF Reconstruction ATE (c) PFH Reconstruction RPE (d) JF Reconstruction RPE

Fig. 8: Evaluation of the Proposed PFH compare to JF, both are the result of first 6000 frames on HrpSlam2, as shown in Fig7. (a) and (c) are absolute trajectory error (ATE) and the relative pose error (RPE) of PFH, while (b), (d) are ATE and RPE of JF. PFH achieves very small trajectory error, average 12 cm (short red line segments in (a)), while JF gets big ATE, long red line segments in (b). (c) and (d) indicate PFH achieves about 3 times smaller average RPE than JF.

Here it only shows the ATE and RPE before robot falls down for “fairness” , which does not indicate this SLAM method has ability to resist falling down although it shows some data in previous table.

Personal Thoughts

1. Too many parameters need to be tuned: min-cut
2. Only detect humans as dynamic object: openpose constrain
3. Dataset is unavailable: The paper says that we can access HrpSlam dataset, one of contribution of this paper. However cannot find the dataset
4. Robot fall down: finally the paper concludes that its method does a good job when robot falls down. However, it only uses data without falling down to test its algorithm

Thanks!