PoseFusion2: Simultaneous Background Reconstruction and Human Shape Recovery in Real-time

Huayan Zhang[1], Tianwei Zhang[1,*], Tin Lun Lam[1,2,*], and Sethu Vijayakumar[3,4]

***Abstract*— Dynamic environments that include unstructured moving objects pose a hard problem for Simultaneous Localization and Mapping (SLAM) performance. The motion of rigid objects can be typically tracked by exploiting their texture and geometric features. However, humans moving in the scene are often one of the most important, interactive targets – they are very hard to track and reconstruct robustly due to non-rigid shapes. In this work, we present a fast, learning-based human object detector to isolate the dynamic human objects and realise a real-time dense background reconstruction framework. We go further by estimating and reconstructing the human pose and shape. The final output environment maps not only provide the dense static backgrounds but also contain the dynamic human meshes and their trajectories. Our Dynamic SLAM system runs at around 26 frames per second (*fps*) on GPUs, while additionally turning on accurate human pose estimation can be executed at up to 10 *fps*.**

I. Introduction

Increasingly, robots are expected to work in close collaboration with humans in dynamic environments. Such collaborative working spaces may contain multiple types of dynamic objects that invalidate classical SLAM frameworks. Many recent works [1], [2], [3], [4] have addressed the problem by attempting to identify and remove the dynamic objects, in order to apply the classical static SLAM frameworks. These methods directly remove dynamic objects to improve the robustness of the front-end visual odometry (VO) and cannot represent the interactive targets (*e.g.*, humans) of the robot. The reconstruction of dynamic human object is challenging for the SLAM system. Because, first, tracking of non-rigid surfaces poses difficulties for the visual odometry of moving cameras. Second, non-rigid surface representation is very expensive for 3D rendering.

In this paper, we focus on the human rich dynamic environments and propose a real-time **dynamic SLAM** solution with **human object recovery**. We firstly follow the human segment removal idea of PoseFusion (PF) [1] — adopting learning-based method [5] detect and remove human objects for precise static background reconstruction. We then recover them using SMPL [6] human models for robot interaction. Finally, the recovered meshes, human motion trajectories, and camera trajectory are presented within the reconstructed static background maps. Our contributions are:

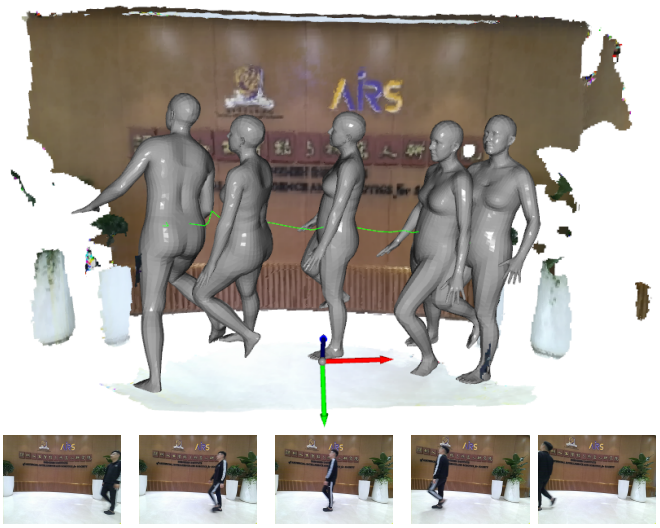


Fig. 1: Dynamic Scene Reconstruction. Our proposed method performed dynamic human object removal, tracking and recovery and static environment mapping in 26 *fps*. The green line is the estimated human moving trajectory.

- A fast learning-based method to track moving humans and to filter the background's residual human bodies (around 26 *fps* without online mesh rendering).
- An improved loop closure in long distance/large area dynamic human environments.
- A fast human pose and shape recovering pipeline to stack the dynamic human meshes with their moving trajectories.

II. Related Works

A. Dynamic SLAM solutions

Most of the existing dynamic SLAM solutions try to deal with the dynamic environment problem by finding and removing dynamic objects. Based on their object recognition approaches, we divide the current state-of-the-art into motion segmentation-based and object detection-based methods.

Object detection-based dynamic SLAM methods usually utilize advanced deep learning-based object detectors to remove dynamic objects, and then enable the classical static SLAM frameworks in the dynamic environments. Zhang *et al.* [1] combined Openpose [7] skeleton estimator into ElasticFusion (EF) [8] to remove dynamic human body point clouds in dense background reconstruction. Bescos *et al.* [2]

[1]The Shenzhen Institute of Artificial Intelligence and Robotics for Society (AIRS), 518129 Shenzhen, China.
[2]The Chinese University of Hong Kong, Shenzhen, 518172 Shenzhen, China.
[3]The School of Informatics, University of Edinburgh, Edinburgh and The Alan Turing Institute, UK.
[4]The author is a visiting researcher with the Shenzhen Institute of Artificial Intelligence and Robotics for Society (AIRS).
*Corresponding Author: zhangtianwei@cuhk.edu.cn; tllam@cuhk.edu.cn

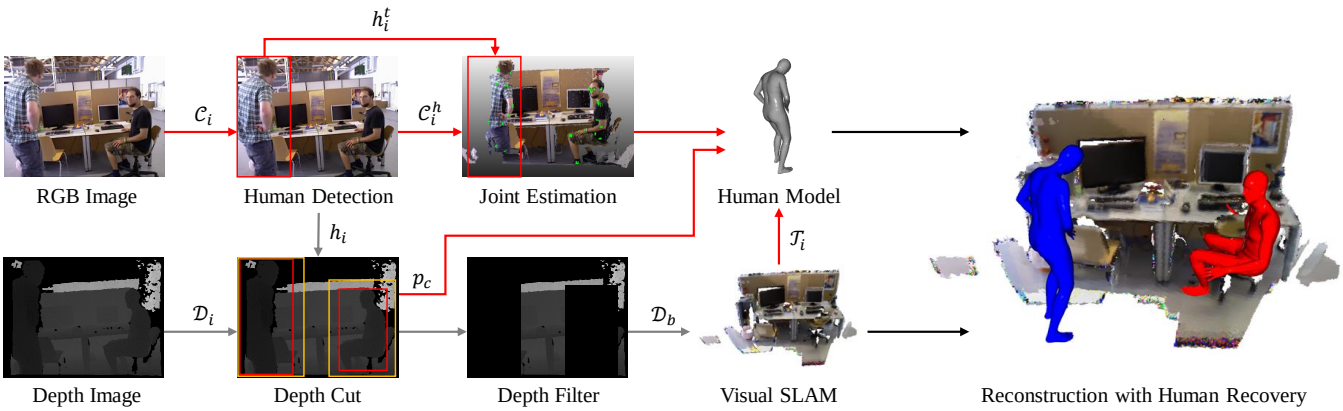


Fig. 2: Approach Flowchart. We adopt a fast Dual Bounding Box (the yellow and red boxes) object detector to separate humans objects from backgrounds. The segmented human shapes are recovered to human models and finally represented together with reconstructed backgrounds.

proposed DynaSLAM which applied Mask-RCNN [9] to detect human objects in RGB images and adopted ORB-SLAM2 [10] for camera tracking. The Mask-RCNN outputs accurate human silhouettes, but it takes abound 300 ms per frame.

Motion segmentation based approaches attempted to find dynamic pixels or point clouds rather than recognizing moving objects. Scona *et al.* proposed StaticFusion [3] that combined scene flow computation with EF's Visual VO to achieve real-time static background reconstruction in a small-sized room. Judd *et al.* provided a multi-object motion segmentation method in [4], which applied sparse feature points alignment to separate and track multiple rigid objects.

B. *Dynamic Human Reconstruction*

The general solution for dynamic object reconstruction in dynamic environments is to omit the rigid backgrounds and explicitly model the scene's non-rigid components. Traditional methods use the pre-scanned template and match the template to live RGB or RGB-D streams, a recent work based on this idea is *e.g.*, livecap [11]. They developed the deep learning framework of template-based human motion tracking and obtained real-time performance. Another is the volumetric approach. Newcombe *et al.* proposed the first template-free on-the-fly framework in [12] for non-rigid surfaces reconstruction. More recently, SurfelWarp [13] extended this framework and cut down the memory and computation cost by employing Surfel rather than a 3D volume representation.

To deal with the noise and reduce online computation, another group of methods fit learned human models, *e.g.*, SMPLify [6] and SMPL-X [14], to sparse **Pose Estimation** results for shape recovery. Kanazawa *et al.* proposed Human Mesh Recovery (HMR) [15] which developed a Generative Adversarial Network to recover 3D SMPL human shapes from the estimated 2D joint key-points. Based on HMR, Muhammed *et al.* developed an efficient Video Inference for Human shape Estimation (VIBE) [16].

C. *Human Shape Recovery within SLAM*

In terms of **Human Shape Recovery in SLAM** applications, there are not many exemplars yet that work robustly. The first attempt was [17]. The authors take dense RGB-D SLAM reconstruction results and SMPL-X human model as input to generate **Offline Static data**, which contain expressive 3D human bodies that naturally interact with the 3D environment. Then, Rosinol *et al.* proposed a human node concept in dynamic SLAM scene and a graph CNN approach to regress SMPL vertices and track the human torsos in [18]. More recent, Li *et al.* proposed SplitFusion (SpF) [19] which, in parallel, reconstructs rigid backgrounds and non-rigid human objects.

Overall, dense key-point tracking based non-rigid dynamic human reconstructing methods such as [19] [12] [13] are able to show more surface details *e.g.*, hairs, clothes folds and emotions. However, it's hard to improve noise robustness and online efficiency. Model-based human shape estimation approaches, such as [14] [16] [15] can efficiently output barebone 3D human meshes with the help of fast learning-based 2D human pose estimation tools. This scheme is promising for real-time systems and the basic human mesh is good enough for Human Robot Interactive (HRI) applications. However, these model based methods require rich body joint movement input to ensure accurate whole body shape recovery. This requirement is usually hard to satisfy in SLAM systems where human target moves in close proximity to the mobile robots.

III. SYSTEM FRAMEWORK AND METHODS

In this study, we present a fully automatic system which simultaneously reconstructs static background and recover 3D dynamic human objects in the environments. To improve the loop closure performance in dynamic environments, we integrates BundleFusion [20] as the baseline method. Different from SpF, as a trade-off between real-time performance and human mesh accuracy, we implement human tracking in feature-based SLAM and reconstruct dynamic 3D human silhouette bodies using the model-based method. The method pipeline is shown in Fig. 2. The various components of the

Algorithm 1 Human Detection and Outlier Filtering

Input: Color image $\mathcal{C}_i$ and Depth image $\mathcal{D}_i$
Output: Human detection $\mathcal{D}_h$ and Filtered background $\mathcal{D}_b$

$h \leftarrow human\_detection(\mathcal{C}_i)$
for each $h_i^t \in h_i$ **do**
  $\mathcal{D}_h^t, \mathcal{D}_b \leftarrow scene\_separation(\mathcal{D}_i, h_i^t)$
  $h_{min}^t, h_{max}^t \leftarrow compute\_histogram(\mathcal{D}_h^t)$
end for
$b_{min}, b_{max} \leftarrow compute\_histogram(\mathcal{D}_i)$
for each $h_i^t \in h$ **do**
  $x_{ul}^t, y_{ul}^t, x_{lr}^t, y_{lr}^t \leftarrow magnify\_range(\lambda * h_i^t)$
  for $u \in ({}^i x_{ul}^t, {}^i x_{lr}^t), v \in ({}^i y_{ul}^t, {}^i y_{lr}^t)$ **do**
    if $\mathcal{D}_b(u,v) \notin (b_{min} - 0.1, b_{max} + 0.1)$ **then**
      $\mathcal{D}_b(u,v) \leftarrow 0$
    end if
    if $\mathcal{D}_h^t(u,v) \notin (h_{min}^t - 0.2, h_{max}^t + 0.2)$ **then**
      $\mathcal{D}_b(u,v) \leftarrow \mathcal{D}_h^t(u,v)$
      $\mathcal{D}_h^t(u,v) \leftarrow 0$
    end if
  end for
end for

pipeline are detailed in the following section: 1) we adopt a fast Bounding Box (BBox) object detector to separate humans from backgrounds (Section III-A); 2) we remove and track the moving human objects for static backgrounds reconstruction using a dual BBox method (Section III-B); 3) finally the separated human segments are fed into 2D pose and 3D shape estimation (Section III-C).

A. Fast Human Detection and Outlier Filtering

An RGB-D stream is taken as input, where the image pair is denoted as $f_i = (\mathcal{C}_i, \mathcal{D}_i)$, where $\mathcal{C}_i : \Omega \rightarrow \mathbb{R}^3$ and $\mathcal{D}_i : \Omega \rightarrow \mathbb{R}$ stand for the i-th color image and depth image, respectively. $\Omega \subset \mathbb{R}^2$ is the image domain. Fig. 2 shows the system framework, each human body is detected by a learning-based algorithm, and human detection is refined based on geometric relationship. The RGB frame $\mathcal{C}_i$ is first input into YOLOv4 [5] for human detection. YOLOv4 is a one-stage detection method that can detect 80 types of objects in the RGB image. To speed up object detection as much as possible, we applied an accelerated implementation of YOLOv4 named tkDNN. It speeds up the inference of YOLOv4 while maintaining accuracy.

Once the object is detected, the categories, bounding boxes, and confidence of all objects in the scene can be derived. The bounding box is composed of the coordinate values of the upper left point (x_{ul}, y_{ul}) and the lower right point (x_{lr}, y_{lr}) in the image plane. We store N bounding boxes containing human bodies with confidence $c > 0.5$ as a set $h_i = \{h_i^1, h_i^2, ..., h_i^N\}$, where N is the number of humans detected in $\mathcal{C}_i$. We use $h_i^t = ({}^i x_{ul}^t, {}^i y_{ul}^t, {}^i x_{lr}^t, {}^i y_{lr}^t)$ to represent the t-th bounding box in h_i. The detected human body is treated as the non-rigid object in our framework, which is only used for human motion tracking and recovery. The static part will be used for performing SLAM.

However, it is not feasible to remove the range of the human body in the depth image directly based on the detection results in the RGB image. The reasons are as follows:

- The object detector may output an incomplete bounding box of human detection.
- The depth image and color image are not strictly aligned, which may lead to the human bodies' residual.
- The high-speed movement of the human and the depth camera results in the blur of the captured image, and the depth ranging based on ToF will be disturbed.

To deal with these problems, we remove the remnant outliers based on geometric features in the depth image. Considering that the outliers are near the bounding boxes h_i, their depth values must be close to the human body. We use histogram statistics to detect abnormal depth values. The data distribution of the human part will differ significantly from the background. We first divide $\mathcal{D}_i$ into background $\mathcal{D}_b$ and human parts based on h_i. Each human part is denoted by $\mathcal{D}_h^t$. Then, we can get the histograms of $\mathcal{D}_b$ and each $\mathcal{D}_h^t$. For the interval with the most occurrences, we use it as the principal component of $\mathcal{D}_b$ and each $\mathcal{D}_h^t$. It means that the depth of $\mathcal{D}_b$ is mainly in interval (b_{min}, b_{max}), and each $\mathcal{D}_h^t$ is mainly in interval (h_{min}^t, h_{max}^t). To preserve more background details, we filter the abnormal depth near each h_i^t and recover parts of the background. More specifically, we extend the filtering range to $\lambda * h_i^t$ to filter boundary depth residuals. λ is an experimentally selected constant. After the above steps, the filtered background $\mathcal{D}_b$ is used for performing SLAM. The human detection $\mathcal{D}_h$ is input to the next step. The pseudocode for human detection and outlier filtering is presented in Algorithm 1.

B. Human motion tracking

Human motion tracking aims to find the movement trajectory of each human character during SLAM. From the previous stage, we have the human detection $\mathcal{D}_h$ and filtered background $\mathcal{D}_b$. We firstly estimate the camera pose in the world coordinate system $\mathcal{F}_w$. The initial position of the camera is taken as the origin of $\mathcal{F}_w$. Then, we calculate the position of each human in the camera coordinate system $\mathcal{F}_c$. Based on temporal and spatial patterns, we finally get the movement trajectory of each human.

Following the RGB-D fusion framework [20], the camera pose $\mathcal{T}_i \in SE(3)$ is estimated by using an efficient global pose algorithm. $\mathcal{T}_i$ denotes a 4×4 rigid transformation relative to $\mathcal{F}_w$. It is formulated as an optimization problem of sparse features and dense photometric and geometric constraints. The $\mathcal{T}_i$ can be solved from the energy function:

$$\mathcal{T}_i = \arg\min_{\mathcal{T}_i} \{w_s E_s(\mathcal{T}_i) + w_d E_d(\mathcal{T}_i)\} \tag{1}$$

in which $E_s(\mathcal{T}_i)$ is sparse term for coarse alignment, $E_d(\mathcal{T}_i)$ is the dense term for refined alignment, and w_s and w_d are weights corresponding to the sparse and dense term, respectively. This energy function is solved by a GPU-based algorithm [20]. Our method only estimates the camera pose

in the filtered background $\mathcal{D}_b$ so that we can get the robust motion estimation.

Then, the position of each human relative to $\mathcal{F}_c$ is calculated. We use the center point of each h_t as the reference. Instead of directly extracting the depth values of the center point in the depth frame, we calculate the average depth values d_m in each $\mathcal{D}_h^t$. The reasoning is that the center point is not necessarily located on the human body when it is not upright, such as in a sitting pose. The 3D center point p_c relative to $\mathcal{F}_c$ is defined in homogeneous coordinates and computed by the pinhole camera model:

$$p_c = (\frac{x_{ul} + x_{lr} - 2c_x}{2f_x} d_m, \frac{y_{ul} + y_{lr} - 2c_y}{2f_y} d_m, d_m, 1)^T \tag{2}$$

where f_x, f_y, c_x, c_y are the intrinsic parameters of the camera. Each p_c relative to $\mathcal{F}_w$ in the i-th frame can be calculated as:

$$^i p_c^w = \mathcal{T}_i p_c \tag{3}$$

After these steps, we can get the position of each human body during SLAM. However, it cannot generate a continuous trajectory. We only get a series of discrete key points; these points do not correspond to each human body. The reason is that the h_i calculated at each moment does not contain a unique identity. The human body represented by h_i^t and h_{i-1}^t in the last time may be different. To address this, we design a heuristic to generate a continuous trajectory for each human character. We assume that if the center point of h_i^t and h_{i-1}^t are closest, they represent the same human.

C. Human Shape Recovery in Static Reconstruction

We use a model-based method to recover the human body. Following the human recovery framework VIBE [16], we use the SMPL model to express the human shape and pose. The 2D human joint is estimated in the range of each h_i^t based on Openpose [7]. Each human joint is projected to 3D planes to find the orientation of the human body. Then, the SMPL model's parameters are estimated based on pose prior and shape prior. After the above steps, each human's model can be recovered. However, the recovered model does not yet contain spatial information. To recover the human model in static reconstruction, we must find their spatial relationship, which we represent by a transformation matrix $\mathcal{T}_h^w$.

The human's base coordinate system $\mathcal{F}_h$ is located in their waist. It has the same x-axis orientation as $\mathcal{F}_w$, but the y-axis and z-axis are reversed. So we firstly rotate $\mathcal{F}_h$ by 180 degrees about the current x-axis to make $\mathcal{F}_h$ and $\mathcal{F}_w$ the same direction. The human model output by VIBE is inferenced in $\mathcal{F}_c$. To get the posture of the human model under $\mathcal{F}_w$, we then rotate it based on camera pose's rotation part $\mathcal{R}_i \in SO(3)$. Finally, we displace $\mathcal{F}_h$ according to human's 3D center point $^i p_c^w$ under $\mathcal{F}_w$. The final transformation is computed as:

$$\mathcal{T}_h^w = \begin{bmatrix} \mathcal{R}_i \mathcal{R}_{x,\theta} & ^i p_c^w \\ \mathbf{0} & 1 \end{bmatrix} \tag{4}$$

where $\mathcal{R}_{x,\theta}$ represents a rotation of angle θ about the current x-axis and $\theta = 180$. After we have inferred $\mathcal{T}_h^w$ and the human model, it can be recovered in the static reconstruction.

IV. Experiments Results and Evaluations

We evaluate the proposed approach under three metrics: **visual odometry, scene mapping** and **time efficiency**. To ensure the scalability of the method to different application scenarios, we test it on both benchmark dynamic SLAM data sets and real world environments. The public data sets used include:

1) TUM RGB-D SLAM data set [21]: A benchmark containing RGB-D and ground-truth data for the evaluation of VO and visual SLAM systems – the *fr3* sequences are iconic and one of the earliest dynamic SLAM scene standards, complete with online evaluation tools [22].

2) HRPSlam humanoids robot dynamic SLAM dataset [23]: The first humanoid robot dynamic SLAM data set, HRPSlam2 (23543 frames, 879.2 seconds) scene is very challenging, as it involves several cases of robot's falling and re-initializing. These sudden motions caused by the robot falling and the discontinuity of visual information predicated higher **camera relocation** and **global loop closure** requirements for SLAM approaches.

In addition, the real-world AIRS indoor dynamic scenes, shown in Fig. 1 and 6, are captured with an Azure Kinect sensor. All the reported results were obtained on a desktop with Intel Core^{TM} i9-9980XE CPU @ 3.00 GHz $\times$ 36, 128 GB System memory and Four GeForce RTX 2080 Ti GPUs.

TABLE I: Absolute Trajectory Error RMSE (cm)

Sequence	DynaSLAM	SF	PF	BF	Ours
fr1/xyz	1.0	1.4	1.9	2.0	2.0
fr1/desk2	2.2	5.2	4.0	7.7	7.7
fr3/walking_xyz	1.7	9.2	4.8	Fail	5.1
HRPSlam2.1	**4.2**	51.4	31	7.5	7.1
HRPSlam2	**5.4**	174	Fail	34.5	10.8

A. Visual Odometry Evaluations

The SLAM solution's VO is usually evaluated by the camera tracking Absolute Trajectory Errors (ATE). We compute ATE's root-mean-square deviation (RSME) via the tools in [22]. We compared the proposed method with state-of-the-art (SOTA) dynamic environment reconstruction methods: DynaSLAM, StaticFusion (SF), PoseFusion (PF) and the baseline approach BundleFusion (BF).

Table I shows the VO performances on TUM data set sequences (the sequences start from fr) and HRPSlam [23] RGB-D data sets. The first two $fr1$ sequences are static environments. All the methods achieve very small errors, which indicate that all methods work equally well in static environments. Note that ours and BF have the same error values in static scenes since we adopted BF as the basic method. The lower three rows are dynamic sequences. In TUM dynamic sequence *fr3/walking_xyz*, our proposed method achieves 5.1 cm ATE, which is better than

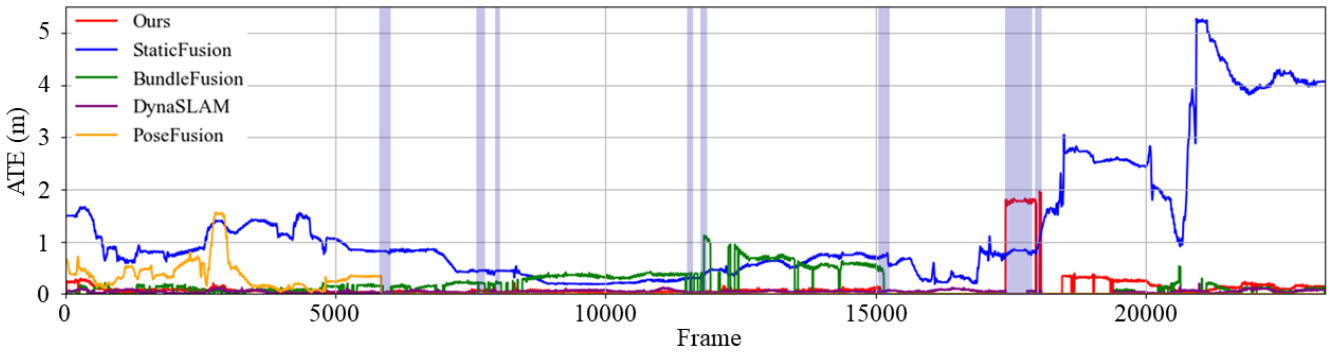


Fig. 3: Absolute Trajectory Errors (m) comparison in HRPSlam2 sequence. The shaded areas indicate the five times camera was violently disturbed from robot falling. Our method achieved the smallest errors amonst the online SLAM methods and succeeded in camera relocation under difficult fall scenarios.

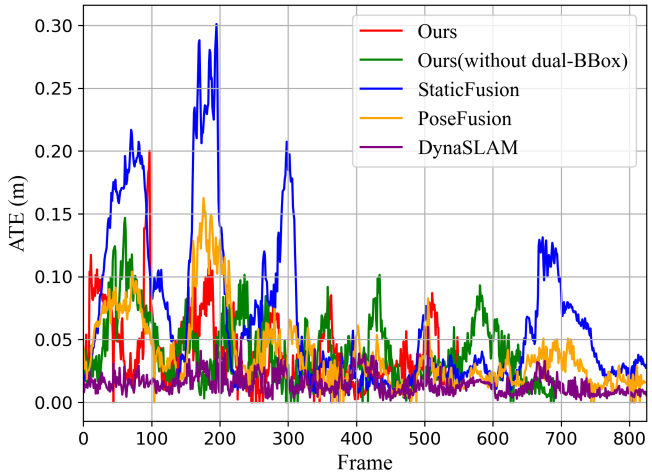


Fig. 4: The ATE of SOTA SLAM methods in the TUM $fr3$ dynamic sequence. Our method ATE was close to PF and better than SF.

SF's 9.2 cm (see the red curve in Fig. 4). Without the Dual-BBox algorithm, the ATE degrades to 5.9 cm. These results indicate that the proposed Dual-BBox strategy (we set $\lambda = 1.2$ in TUM and HRPSlam experiments, it costs 3.5 ms per frame) efficiently improves the VO performance to the SOTA level. PF spent more than 500 ms on human 3D point cloud segmentation to obtain 4.8 cm ATE. For the other SOTA methods, DynaSLAM achieved the best camera tracking result of 1.7 cm since it spends much time on accurate human mask segmentation. BF result is not shown since it failed in this sequence. The reason is that BF divides front-end VO and back-end global loop closure into two pipelines run on two separate GPUs. This framework design greatly improves global mapping performance in large room size scenes.

The plots in Fig. 3 shows the ATE values along with the frame IDs in HRPSlam2 whole sequence. The curves of different colors stand for: Our method (red), DynaSLAM (purple), StaticFusion (blue) and BundleFusion (green). Note that the ATE trajectories started from a non-zero offset since the evaluation tool [22] tried to align the whole global trajectories to the GTs. HRPSlam2.1 is the sequence before the robot first fall of HRPSlam2. All five methods are able to track the camera pose in this sequence, but in the following sequence, the method without robust camera repositioning fails. In this difficult scene, as our method applied advanced dynamic object detection and removal technique, VO errors are competitive to the other SOTA online methods. It achieved 10.8 cm ATE which is better than BF (34.5 cm) and SF (174 cm), as evidenced in the reconstructed maps (Fig. 8). Offline approach DynaSLAM obtained a smaller RMSE error of 5.4 cm with the help of Mask-RCNN's careful human silhouette segmentation. The PF camera tracking comprehensively fails after the humanoids falling at frame 5480 and it cannot re-locate the camera pose after the fall. There are two reasons to this. Firstly, PF applied Openpose as human object key-point detector, but it failed to find the human objects without heads (In HRPSlam, the camera is mounted on the robot facing the ground). Secondly, PF and SF are based on the ElasticFuison [8] framework. They emphasize local small area loop closure, but they lack robust global loop detection back-ends.

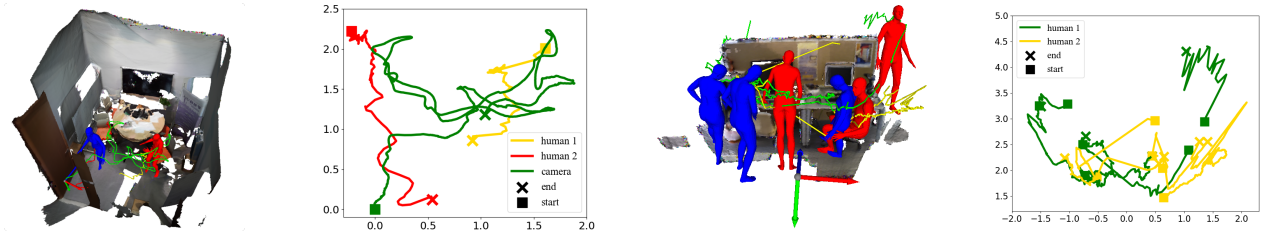
We evaluate the scene reconstruction performance of the dense SLAM methods from two aspects. The first is the performance of human object representation. The second is the ability to close the global loop in dynamic scenes.

B. Human Object Representation Evaluations

SplitFusion (SpF) and our method has the ability to simultaneously recover human shape and static backgrounds. SpF implements a VO and non-rigid tracking parallel pipeline. Its VO thread is exactly the same as PF, thus, its static background reconstruction performance is similar to ours. For the foreground human objects, SpF reconstructs the human mesh via non-rigid tracking, while we replace the human object by the estimated SMPL mesh. The proposed approach is superior to SpF in three perspectives. Firstly, Fig. 5 shows the $fr3\_walking\_xyz$ sequence scene reconstructions, from which it can be found that SpF non-rigid tracker can not track fast-moving objects, *e.g.*, calves, feet, and hands, while our model-based method represents the complete human shapes, see the red circle area. Moreover, SpF cannot distinguish connected objects, for instance, the chair was fused into the green human object. Such mesh representation is hard to recognize as an interactive target. As a comparison, our method output complete and clear human meshes. In addition, our method can effectively track moving

Ours SplitFusion

Fig. 5: Reconstruction results on *fr3_walking_xyz* sequence of TUM dataset. The input images are as same as Fig. 2. On the left, our method outputs complete and clear human meshes. On the right, SpF failed to track the fast subject (see missing feet circled in red). In addition, it cannot distinguish connected objects, *e.g.*, in the blue circle, the chair was fused into the green human mesh.



(a) AIRS scene (b) AIRS scene trajectories (c) TUM $fr3$ scene (d) TUM $fr3$ trajectories

Fig. 6: Scenes reconstruction with human meshes and trajectories. (a), AIRS room scene. (b), the estimated trajectories in (a). (c), TUM $fr3$ scene, three recovered meshes for each object were represented to show their motions. (d), human object trajectories of (c).

human targets by BBox center trajectory. As shown in Fig. 6, the two examples of AIRS and TUM scenes demonstrate that our method can insert a recovered human subject at a location along the estimated trajectory. In (a), we inserted the objects to their last seen positions; the 2D estimated object trajectories together camera's were plotted in (b). In the TUM scene (c), we represented three recovered meshes for each object to show their pose changes. Since TUM camera moved in a vertical plane, it was not plotted in (d).

C. Scene Reconstruction Evaluations

Our Dynamic environment mapping performance of a real dynamic scene is shown in Fig. 1. This sequence was captured in AIRS using an Azure Kinect RGB-D sensor. The proposed method performed dynamic human object removal, static environment mapping, and human object tracking and recovery at 21 *fps*, with the green line indicating the recovered human motion trajectory. Next, Fig. 7 captures the results of another AIRS dynamic scene reconstruction with multiple moving subjects. For SF, BF, and our proposed method, the mapping proceedings are shown. As DynaSLAM is an online VO and offline mapping approach, in the second row, only the feature-based camera tracking processes are shown. SF kept VO accuracy in the first 300 frames (image 1 to 2, third row), but it did not achieve sufficient loop closure to maintain global mapping. The moving objects in the third image result in a big VO drift error for BF, it then reduced these drift by global loop closure after the objects moving out of view. Our method removed the moving human objects simultaneously with local mapping. Therefore, we kept VO robustness and further improved the mapping results using BF's key-frame strategy.

TABLE II: Computation Speed *fps* of TUM Database

DynaSLAM	SpF	SF	PF	BF	Ours
1.58	0.2-	16.74	0.3-	24.04(37.89)	25.84

The mapping performance of the HRPSlam2 scene in Fig. 8 clearly demonstrates the global loop closure capability of these methods in a dynamic environment. See the first image, The trajectory of SF cannot be aligned with GT. Because it cannot re-locate itself after the camera tracking fails. In the second image, BF lost robustness in dynamic environments. However, with closed-loop detection, It realigned the camera pose to GT trajectory several times when the obstacle moved away, see the long slash lines. DynaSLAM remained robust VO performance, except for the failure in the full occlusion case that happened at HRPSlam2 dataset Frame 18570. To reduce the influence of the dynamic humans on the closed-loop detection, 1) we developed YOLOv4 based dual-BBox human object detector, which enhanced the human detection robustness in such a not full body visible scene. 2) We selected BF as the basic approach, the advanced key-frame processing and storing strategies contribute to reliable global loop closure. With these two improvements, our method accomplished excellent camera re-location and continue the mapping system even under difficult fall scenarios plus full occlusion case.

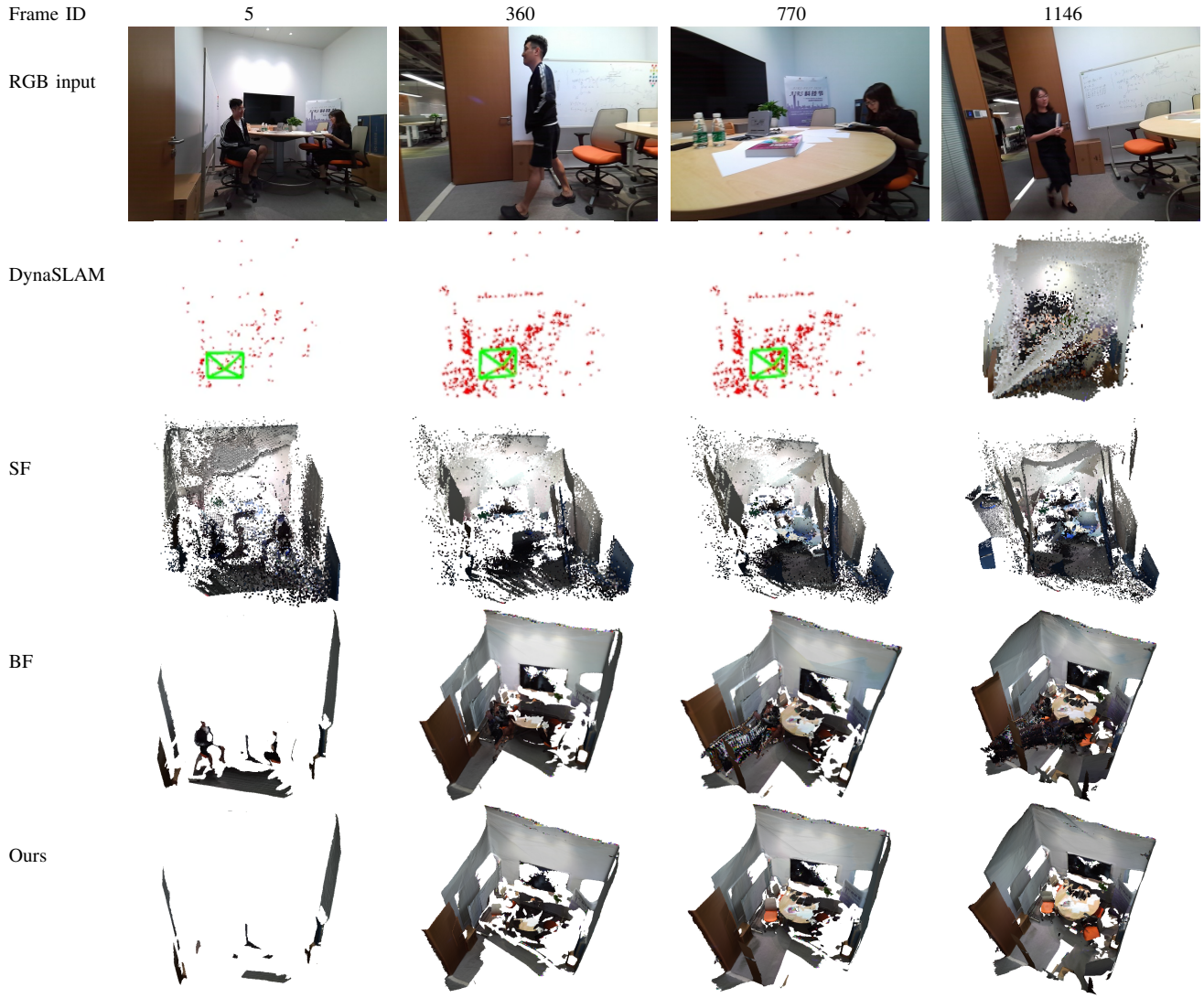


Fig. 7: AIRS real dynamic scene reconstruction proceedings. As DynaSLAM is an online VO and offline mapping approach, the camera tracking with sparse ORB feature points are shown with frame IDs. For SF, BF, and our proposed method, the resultant mappings are shown. Please refer to text for discussion of the results

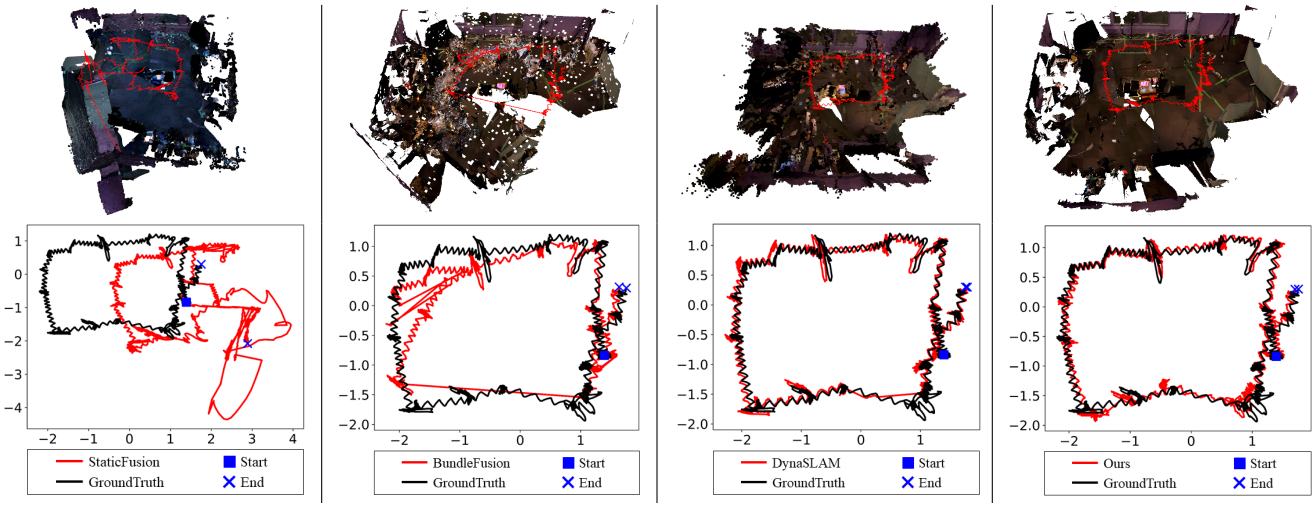


Fig. 8: Mapping performances in HRPSlam2 sequence that contains difficult humanoid robot falling cases. From left to right: SF performed wrong loop detection; BF VO failed in dynamic scene, but it re-located the camera pose after humans moving away. DynaSLAM also failed in the robot fall near the end; Our method accomplished excellent re-location and mapping results. For the details please refer to Section IV-A.

D. Time Cost Evaluations

The online processing *fps* comparison is shown in Tab. II. The first four methods use a single GPU and BF uses two GPUs. The *fps* of SpF and PF depend on the situations within the dynamic scenes. For SpF, the frames containing complex non-rigid motions cost more computation time; for PF, the number of human objects and visible body joints result in lower *fps* performance. In the TUM dynamic sequence, SpF is lower than 0.2 *fps*, and PF is lower than 0.3 *fps* on average. DynaSLAM spends more than $500\ ms$ on Mask-RCNN 2D human object segmentation, and then applies ORB based VO front end for camera tracking. BF performed at 37.89 *fps* in static scenes, *e.g.*, $f1/xyz$, but it drops to 24 *fps* in a dynamic scene since the back-end loop detector fails when it tries to close the loop in dynamic environments. Our method achieved 25.84 *fps* in the TUM dataset which contains two fast-moving human objects (see Fig. 6 (c) and (d)) and 27.79 *fps* in HRPSlam dataset which contain five moving humans (see Fig. 8). Our real-time performance in such hard dynamic scenes benefits from speed-up resulting from the usage of four-GPU – here, we adopted BF's dual-GPU system design as a basic supplemented with a YOLOv4 based dual-BBox human object detector and SMPL model-based human mesh recovery pipeline.

V. Conclusions

In this paper, we present a real-time dense RGB-D SLAM approach for reconstructing accurate static backgrounds together with representing human pose and shape in dynamic humans environment. To the best of our knowledge, this is the first SLAM approach that provides both static backgrounds and dynamic human objects in real-time. The whole system runs at 26 *fps* without online GUI rendering (21 *fps* with GUI). The represented human mesh is also directly amenable to be deployed for human-robot collaborative loco-manipulation with mobile robots to efficiently reason over target actions in unknown dynamic environments. Future research directions include involving temporal coherence to refine the estimated 3D human shape and optimizing the loop detector by considering environment semantic information.

ACKNOWLEDGEMENT

This work is supported by the Shenzhen Institute of Artificial Intelligence and Robotics for Society (2019-ICP002), The Alan Turing Institute and EU H2020 project Enhancing Healthcare with Assistive Robotic Mobile Manipulation (HARMONY, 9911237).

References

[1] T. Zhang and Y. Nakamura, "Posefusion: Dense rgb-d slam in dynamic human environments," in *Proceedings of the 2018 International Symposium on Experimental Robotics*, 2020, pp. 772–780.

[2] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, "Dynaslam: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4076–4083, 2018.

[3] R. Scona, M. Jaimez, Y. R. Petillot, M. Fallon, and D. Cremers, "Staticfusion: Background reconstruction for dense rgb-d slam in dynamic environments," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2018, pp. 1–9.

[4] K. M. Judd, J. D. Gammell, and P. Newman, "Multimotion visual odometry (mvo): Simultaneous estimation of camera and third-party motions," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3949–3956.

[5] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[6] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "SMPL: A skinned multi-person linear model," *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, vol. 34, no. 6, pp. 248:1–248:16, Oct. 2015.

[7] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "Openpose: realtime multi-person 2d pose estimation using part affinity fields," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 1, pp. 172–186, 2019.

[8] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "Elasticfusion: Real-time dense slam and light source estimation," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1697–1716, 2016.

[9] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE International conference on computer vision*, 2017, pp. 2961–2969.

[10] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[11] M. Habermann, W. Xu, M. Zollhoefer, G. Pons-Moll, and C. Theobalt, "Livecap: Real-time human performance capture from monocular video," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 2, pp. 1–17, 2019.

[12] R. A. Newcombe, D. Fox, and S. M. Seitz, "Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 343–352.

[13] W. Gao and R. Tedrake, "Surfelwarp: Efficient non-volumetric single view dynamic reconstruction," in *Robotics: Science and System (RSS)*, 2018.

[14] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. Osman, D. Tzionas, and M. J. Black, "Expressive body capture: 3d hands, face, and body from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 975–10 985.

[15] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik, "End-to-end recovery of human shape and pose," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7122–7131.

[16] M. Kocabas, N. Athanasiou, and M. J. Black, "Vibe: Video inference for human body pose and shape estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5253–5263.

[17] Y. Zhang, M. Hassan, H. Neumann, M. J. Black, and S. Tang, "Generating 3d people in scenes without people," in *Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[18] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone, "3D dynamic scene graphs: Actionable spatial perception with places, objects, and humans," in *Robotics: Science and Systems (RSS)*, 2020.

[19] Y. Li, T. Zhang, Y. Nakamura, and T. Harada, "Splitfusion: Simultaneous tracking and mapping for non-rigid scenes," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5128–5134.

[20] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, p. 1, 2017.

[21] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 573–580.

[22] *Useful tools for the RGB-D benchmark*. [Online]. Available: https://vision.in.tum.de/data/datasets/rgbd-dataset/tools

[23] T. Zhang and Y. Nakamura, "HRPSlam: A benchmark for RGB-D dynamic slam and humanoid vision," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, 2019, pp. 110–116.