3D Scanning of High Dynamic Scenes Using an RGB-D Sensor and an IMU on a Mobile Device

YANGDONG LIU, WEI GAO, AND ZHANYI HU

National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China

Corresponding author: Wei Gao (wgao@nlpr.ia.ac.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB0502002, and in part by the Natural Science Foundation of China under Grant 61872361, Grant 61772444, and Grant 61421004.

ABSTRACT

With the development of RGB-D sensors and mobile devices, 3D scanning has witnessed great progress in recent years. KinectFusion opens up an era of RGB-D 3D reconstruction, which integrates captured depth images into a voxel-based representation. A number of improvements have been applied to the KinectFusion to reduce large footprint and make the 3D reconstruction on mobile devices possible. However, these methods are designed to handle static scenes. In this paper, we propose a method which can perform the 3D scanning of high dynamic scenes using an RGB-D sensor and an inertial measurement unit (IMU) on a mobile device. We first introduce a novel method to segment the depth images into static and dynamic elements with the use of sparse optical flow and the rotational part of the IMU. Then, we determine the camera pose with pixels labeled as static. At last depth, images are integrated into a voxel-based representation. The truncated signed distance function values of static voxels are updated while dynamic voxels are set to free-space. The experiments show that compared with some state-of-the-art systems, our method has better results when scanning high dynamic scenes and has comparable results when scanning low dynamic and static scenes. Besides, our method processes eight frames/s on an Apple iPad Air 2.

INDEX TERMS

Dynamic scenes, background segmentation, RGB-D, mobile device.

I. INTRODUCTION

3D scanning of indoor scenes is key to robotics and augmented reality, which has witnessed a great progress with the development of RGB-D sensors such as the Microsoft Kinect [1] and the Intel RealSense [2]. These sensors provide RGB images together with depth images. It is convenient to integrate these input images into dense 3D models. With the popularity of mobile devices such as mobile phones and tablet computers, 3D scanning of indoor scenes on mobile devices turns to be promising. Recently, devices such as Google Tango [3] and Occipital Structure Sensor [4] make dense 3D scanning on mobile devices practicable. As a consequence, we aim at on-line 3D scanning on mobile devices.

Among the 3D scanning methods, KinectFusion is an outstanding one which is able to generate photo-realistic dense 3D models in real-time [5], [6]. It inspires lots of researches. These methods integrate depth maps into voxel-based representations [7], which has the advantages of computational efficiency and algorithmic simplicity. However, these methods are hard to handle dynamic scenes because voxels within a truncation region around the dynamic surface need to be updated. Alternatively, captured depth images can be integrated into point-based or surfel-based representations [8], [9]. Surfels are effective to handle dynamic scenes because only the dynamic surfels need to be updated. However, these methods have the disadvantages of limited model quality, significant computational cost and inapplicability when continuous surfaces are needed.

Methods above rely on a certain assumption that scanned scenes are static, which makes them merely be able to manage very low dynamic or static scenes. These methods implicitly handle dynamic elements with the use of a robust cost function and penalizes the high-residual points, which limits their applicability when the scenes contain high dynamic elements such as there is a walking person in the view. To cope with this problem, many methods are proposed to explicitly handle dynamic elements. One kind of method regards dynamic objects as outliers or noise. It firstly detects and filters out the dynamic object and then uses the static background to...
calculate the camera pose. Afterwards, the static background and the dynamic foreground are reconstructed independently. In order to effectively deform the reconstructed model, most methods integrate input images into surfel-based representations. The other kind of method models the non-rigid motion of general deforming scenes. The significant computational cost of above methods makes them unable to be used on mobile devices. Besides, these methods have not made use of inertial measurement unit (IMU) data that is easy to be captured on mobile devices.

In this work, we propose a method that is able to reconstruct dynamic scenes using a voxel-based representation on a mobile device. Experiments show that compared with some state-of-the-art systems, our method has better results when scanning high dynamic scenes and has comparable results when scanning low dynamic and static scenes. Our method achieves frame rates 8FPS on an Apple iPad Air 2. The main contributions of the proposed method are as follows:

- We propose a novel method to segment RGB-D images into dynamic and static elements with the use of IMU data and sparse optical flow. This method is efficient on mobile devices.
- We integrate depth images into a voxel-based representation while dynamic voxels are set to free-space. As far as we know, we are the first to reconstruct the static backgrounds of dynamic scenes using voxel-based representations.
- We present a system which is able to perform voxel-based 3D reconstruction of high dynamic scenes on a mobile device. Previous works such as StaticFusion and Co-Fusion only can perform surfel-based 3D reconstruction on a PC.

II. RELATED WORK

In this section we firstly discuss related works on the voxel-based 3D reconstruction. Then we discuss some methods that are able to reconstruct dynamic scenes.

A. VOXEL-BASED 3D RECONSTRUCTION

KinectFusion is an RGB-D 3D reconstruction system, which integrates depth images into a voxel-based representation [5], [6]. They store a Truncated signed distance function (TSDF) value and its weight in every voxel of a predefined volume. The TSDF value is the distance between the center of a voxel and the nearest surface of the observed object. The pipeline of KinectFusion consists of four main modules that are depth map conversion, camera tracking, volumetric integration and 3D rendering. Camera poses are calculated by a frame-to-model Iterative Closest Point (ICP) algorithm. After that, the incoming depth image is integrated into the voxel-based representation through a weighted running average procedure. Due to the computational efficiency of voxel-based representation, we choose it to integrate depth images in our method. In order to cope with its disadvantage of large footprint, many methods are proposed afterwards. Moving volume method stores and processes voxels in the field of view in the device memory, while other voxels are turned into meshes and transferred out from the device memory [10], [11]. However, this process is invertible. Other methods allocate and update voxels around the actual surfaces other than the whole scanned space, then they use Octrees [12]–[14] or hash tables [15]–[17] to index these allocated voxels. These methods reduce the occupied memory greatly and enable the voxel-based 3D reconstruction to be used on memory limited devices such as mobile devices. InfiniTAM [16], [17] and CHISEL [18] are two systems that can perform 3D scanning on-line on mobile devices with a hash table to index allocated voxels. We adopt this idea in our system. However, these methods assume that the scanned scene is static. They have not handled the problem of tracking failure when scanning dynamic scenes.

B. HANDLING OF DYNAMIC SCENES

Many methods use a robust cost function and penalize the high-residual points to handle dynamic scenes. Kerl et al. [19] demonstrate this kind of implicit handling is robust to reconstruct scenes that contain small moving objects. Some surfel-based 3D reconstruction methods integrate points that are repeatedly observed to filter out dynamic elements [8], [9]. ElasticFusion is an outstanding one of them [9]. However, these methods fail to track camera poses when the scenes are of high dynamic. Recently, many methods that explicitly handle dynamic scenes are proposed for RGB-D simultaneous localization and mapping (SLAM). Kim and Kim [20] obtain the background by analyzing the distribution of depth difference between consecutive images and compute the background model. VO-SF firstly clusters the point cloud geometrically and then performs a two-fold segmentation which classifies the clusters as static or dynamic [21]. The segmentation is based on the idea that the tracking residuals of the background are low and vice versa. These two methods above are not able to reconstruct the scanned scenes. StaticFusion segments the scene based on the residual of tracking cost function and tracks the camera through a frame-to-model way [22]. Then it fuses the static part into a surfel-based representation to reconstruct the background. Co-Fusion segments the scene into different objects while simultaneously tracks and reconstructs their 3D shape [23]. This method fuses RGB-D images into multiple surfel-based representations. Bescós et al. [24] propose a system named DynaSLAM which detects the moving objects by multiview geometry, deep learning, or both. They use a CNN to pixel-wise segment the priori dynamic objects such as people and cars. Tracking and mapping of the system are based on ORB-SLAM2 with tracked features being on the static objects. However, the significant computational cost of above methods makes them unable to be used on mobile devices. Meanwhile, these methods have not used the IMU data to offer a prior. Kim et al. [25] propose to use IMU to classify 3D feature points into dynamic or static. They only use the static feature points to determine camera poses and have not reconstructed the 3D model of the scanned scene.
In this paper, we propose a method that can reconstruct high dynamic scenes into voxel-based representations on a mobile device. We firstly introduce a novel method to segment the captured image into dynamic foreground and static background on mobile devices. We segment the image pixel by pixel with the use of sparse optical flow and the rotational part of the IMU. Then the point cloud and the RGB information corresponding to the static background are aligned to the reconstructed model through the minimization of the geometrical error and the photometric error. Afterwards, depth images are integrated into a voxel-based representation while dynamic voxels are set to free-space. Through this way, static background of the high dynamic scene is reconstructed on a mobile device.

III. METHOD
In this section we firstly discuss the overview of our system. Then we discuss each of the procedure in the following sections in detail.

A. SYSTEM OVERVIEW
Our system consists of six modules. The system overview is shown in Fig. 1 and is discussed briefly as follows:

1) IMU INTEGRATION
The gyroscope data between two consecutive depth images are integrated to provide a relative rotation. This relative rotation is used in background segmentation and camera tracking.

2) DEPTH MAP CONVERSION
When a depth image is captured, we calculate the 3D point and normal in the camera coordinate system for each pixel. Additionally, we calculate the standard deviation of the depth noise and the gradient for each pixel. After that, we give each pixel a label indicating whether it is on the depth discontinuities.

3) BACKGROUND SEGMENTATION
In this module we aim to segment the captured images into static and dynamic elements. At first we cluster the 3D points using K-Means. Then we segment the captured images into background or foreground with the use of sparse optical flow and IMU data.

4) CAMERA TRACKING
We register the input image and the ray-casted image from the proceeding camera pose by means of minimizing the geometrical error and the photometric error to get a 6 degree of freedom (DoF) rigid camera pose. If IMU data are available, we use them to calculate the initialization of the relative rotation.

5) VOLUMETRIC INTEGRATION
After estimating the camera pose, we allocate voxels that are around the actual surface and index them by a hash table. We modify the classic integration method based on different conditions to manage dynamic scenes.

6) 3D RENDERING
The voxel-based representation is ray-casted to extract views of the implicit surface for camera tracking and visualization. Each of the procedures is detailed in the following sections.

B. DEPTH MAP CONVERSION
In the beginning we apply a bilateral filter [26] to de-noise the incoming depth map $D_i$ that is captured at time $i$. Then we back-project each pixel $u = (u, v)^T$ and its depth value $d_i(u)$ into a 3D point $v_i(u)$ in the camera coordinate system as follows:

$$v_i(u) = d_i(u)K^{-1}[u, 1]^T$$

where $K$ is the intrinsic calibration matrix. Meanwhile, each normal vector is calculated as:

$$n_i(u) = (v_i(u + 1, v) - v_i(u, v))$$
$$\times (v_i(u, v + 1) - v_i(u, v))$$

$$n_i(u) = n_i(u)/\|n_i(u)\|$$

where $n_i(u)$ is normalized to unit length.

We also calculate the standard deviation of the depth noise for each pixel based on the noise model. For Microsoft Kinect v1, we depict the angle between surface normal and the z axis as $\theta_i(u)$. When $\theta_i(u)$ is approaching $\pi/2$, the calculation goes as follows [27]:

$$\sigma_i(u) = 0.0012 + 0.0019(d_i(u) - 0.4)^2 + \frac{0.0001}{\sqrt{d_i(u)}} \frac{\theta_i(u)^2}{(\frac{\pi}{2} - \theta_i(u))^2}$$
If \( \theta_i(u) \in [0, \pi/3] \), the calculation is reduced to:
\[
\sigma_i(u) = 0.0012 + 0.0019(d_i(u) - 0.4)^2
\]  
(5)

For Occipital Structure Sensor, we fit the depth precision curve on its website\(^1\) and calculate the standard deviation of the depth noise as:
\[
\sigma_i(u) = 0.003d_i^2(u)
\]  
(6)

Furthermore, in order to detect depth discontinuities, we need to calculate the gradient of each pixel as:
\[
g_i(u) = \text{Sobel}(u)
\]  
(7)

Here \( \text{Sobel}(\cdot) \) is the Sobel gradient operator, which is chosen owing to its robustness to noise.

Due to occlusion and large angle between the surface normal and the optic axis, depth noise on the depth discontinuities is always large. We need to detect depth discontinuities and reduce their influences on our method. The label indicating whether \( u \) is on depth discontinuities is calculated as:
\[
l_i(u) = \begin{cases} 
1, & g_i(u) \geq \alpha \times \sigma_i(u) \\
0, & g_i(u) < \alpha \times \sigma_i(u)
\end{cases}
\]  
(8)

where \( \alpha \) is a scaling parameter. The depth discontinuity label is used in the background segmentation procedure.

### C. BACKGROUND SEGMENTATION

Dynamic elements in the scenes lead to inconsistent constraints in the cost function of tracking process. In order to robustly calculate the camera poses, dynamic elements need to be segmented and filtered out firstly. We geometrically cluster the incoming point cloud into several parts and use a novel segmentation method to get the pixel-wise foreground/background segmentation. Our background segmentation pipeline is shown in Fig. 2.

1) GEOMETRIC CLUSTERING

As in VO-SF and StaticFusion, we apply the classic K-Means to the 3D coordinates of the incoming point cloud. We assume that the 3D points in a cluster approximately have the same motion and in the following segmentation we give a label to each cluster indicating whether it is dynamic or static. We choose K-Means other than super-pixels because of the computational efficiency and the convenience of feature detection. As for the number of clusters, too few leads to very large regions in which points may have different motion and too many makes it hard to detect enough features in each cluster. We empirically use 20 clusters.

2) OPTICAL FLOW AND IMU DATA BASED SEGMENTATION

The basic idea of our segmentation method is that if the rotational components of static 3D points in consecutive images are compensated, the static 3D points have the same vector difference caused by camera motion. Here we briefly prove it. We use \( T_{g,i} = [R_{g,i}][t_{g,i}] \) and \( T_{g,i-1} = [R_{g,i-1}][t_{g,i-1}] \) to depict the 6-DoF transformations from the camera coordinate system to the global coordinate system at time \( i \) and time \( i-1 \) respectively. For a static 3D point \( v_g \), which is depicted in the global coordinate system and can be seen in two consecutive images, it can be represented as,
\[
v_w = R_{g,i}v_i + t_{g,i}
\]  
(9)

\[
v_w = R_{g,i-1}v_{i-1} + t_{g,i-1}
\]  
(10)

These two equations are combined to give a constraint,
\[
\Delta R v_i - v_{i-1} = R_{g,i-1}^T(t_{g,i-1} - t_{g,i})
\]  
(11)

Here \( \Delta R = R_{g,i-1}^T R_{g,i} \) and we can see that:
\[
\| \Delta R v_i - v_{i-1} \| = \| t_{g,i-1} - t_{g,i} \|
\]  
(12)

Equations above show that when the scanned scene is static, the differences between the rotationally compensated 3D points at \( i \) and their corresponding 3D points at \( i-1 \) are determined by the camera translation.

Based on the equation above, we design our segmentation method as follows. After 3D points are geometrically clustered into several groups, we detect a fixed number of Shi and Tomasi [28] features in each cluster to keep the features uniformly distributed in the current RGB image. Empirically the number of features is 30 in each cluster and their corresponding 3D points at \( i \) are determined by the camera translation.

Then we use the Lucas et al. [29] optical flow to determine corresponding features in the previous image. Due to that the 3D point corresponding to each feature is calculated in the depth image conversion procedure,

---

we obtain a number of matched 3D point pairs in consecutive images. Because the depth noise on the depth border is usually large, we discard 3D points whose discontinuity labels \( d_i(u) \) are 1. The matching procedure above is efficient and would not cost much time on a mobile device.

As the drift of a gyroscope is negligible in a very short time, we integrate the gyroscope data between consecutive images to obtain the relative rotation. The gyroscope data are captured by the IMU on mobile devices. The relative rotation of the current to the previous is depicted as \( \Delta R_{imu} \). For a 3D point \( v_i \) in the current image, its rotationally compensated coordinate \( \tilde{v}_i \) is,

\[
\tilde{v}_i = \Delta R_{imu} v_i
\]

(13)

The corresponding 3D point in the previous image is denoted as \( v_{i-1} \). Consequently, \( v_{i-1} \) and \( \tilde{v}_i \) have the same Euler angle. The motion vector \( m_{i, i-1} \) of a 3D point from time \( i-1 \) to \( i \) is:

\[
m_{i, i-1} = \tilde{v}_i - v_{i-1}
\]

(14)

We name the Z-component of the motion vector as compensated depth difference. The compensated depth difference \( d_{i, i-1} \) of a 3D point from time \( i-1 \) to \( i \) is:

\[
d_{i, i-1} = v_{i}^{(z)} - v_{i-1}^{(z)}
\]

(15)

the superscript \( (z) \) means to select the Z-component. Ideally, compensated depth differences of all static 3D points should be the same. Due to the existence of depth noise, compensated depth differences are different and the difference relies on depth noise. The mean of compensated depth difference is caused by the camera motion. We simply use a Gaussian distribution to model the compensated depth difference in consideration of efficiency.

However, if many dynamic features have consistent motions, the distribution may be bimodal or multimodal. The highest peak corresponds to the compensated depth difference caused by camera motion. We need to determine the highest peak firstly. Here we use a compensated depth difference histogram. For Kinect v1 and Structure Sensor, the minimum measured depth is 0.4m. If we ignore the depth noise caused by the angle, when the depth is 0.4m, the standard deviation of depth noise is approximately 0.0012m and 0.0005m respectively. The range of each interval in the histogram is a little larger than these values and set to 0.002m and 0.001m respectively to cover the depth differences caused by depth noise. When dynamic objects have a speed of 3m/s (the normal walking speed of pedestrian is 1.1m/s-1.5m/s), it means they move 0.1m during the time between consecutive images. Dynamic objects are hard to move faster than this value. Therefore, for Microsoft Kinect v1, we uniformly divide \([-0.1m, 0.1m]\) into 100 bins and count how many compensated depth difference fall into each bin. After constructing the histogram, we find the bin with the largest value in the histogram. From the increasing direction and the decreasing direction centered at the bin with the largest value, we find the first bins of both directions which have less than 10% of the largest value. These two bins construct a range that is depicted as \([a, b]\). We fit a Gaussian distribution to the data in \([a, b]\) and calculate the mean and the standard deviation of the distribution. Then we determine the 95% confidence interval of the Gaussian distribution. The interval whose endpoints are,

\[
\mu \pm 1.96\sigma_h
\]

(16)

where \( \mu \) and \( \sigma_h \) are the mean and the standard deviation of the data in \([a, b]\). For 3D points whose compensated depth differences are inside this interval, we label them as static. We label 3D points whose compensated depth differences are outside this interval as dynamic.

Furthermore, we label 3D points whose compensated depth differences are larger than a noise threshold as dynamic. We set this noise threshold \( T_n \) as,

\[
T_n = \beta \times \sigma_i(u)
\]

(17)

where \( \sigma_i(u) \) is the standard deviation of the depth noise and \( \beta \) is a constant. We set \( \beta = 8 \) because the compensated depth differences caused by the depth noise is hardly more than \( 8 \times \sigma_i(u) \) according to the property of the Gaussian distribution.

For clusters, we label them as dynamic if the proportion of dynamic 3D points is larger than a certain percentage \( p \). If this value is very large, clusters with small dynamic regions may fail to be labeled as dynamic. If this value is too small, clusters with many incorrect matches may be labeled as dynamic. We empirically set \( p = 20\% \). Through this way we obtain the pixel-wise dynamic/static segmentation. Only static pixels are used in the following camera tracking procedure. An example of background segmentation result is shown in Fig. 3. The corresponding compensated depth difference histogram is shown in Fig. 4. In this example, the scanned scene is composed of little dynamic elements. The man’s left shoulder and his head are labeled as static because the cluster corresponding to these two parts is static in this example.

D. CAMERA TRACKING

When the point-pairs in consecutive depth images have been obtained by the projective data association, the transformation \( T_{R,i} \) from camera coordinate system at time \( i \) to the global coordinate system is calculated by minimizing a joint error.

The geometrical error \( E_{geo}(T_{R,i}) \) as follows,

\[
E_{geo}(T_{R,i}) = \sum_{u \in B_{I_{i}}} (\hat{v}_{1}^{R}(\tilde{u}) \cdot (T_{R,i} \hat{v}_{1}(u) - \hat{v}_{1}^{R_{i-1}}(\tilde{u})))^2
\]

(18)

where \( \tilde{u} \) is the projectively associated pixel corresponding to \( u \). Here \( \hat{v}_{1}(u) = [v_{1}(u), 1]^T \) is the homogenous form of point \( v_{1}(u) \) in the current depth image and \( \hat{v}_{1}^{R_{i-1}}(\tilde{u}) \) is the predicted point at the previous time. The predicted normal of \( \hat{v}_{1}^{R_{i-1}}(\tilde{u}) \) is \( \hat{n}_{i-1}(\tilde{u}) \).

We also minimize the photometric error \( E_{pho}(T_{R,i}) \) as follows,

\[
E_{pho}(T_{R,i}) = \sum_{u \in I_{i}} (I_{i}(u) - I_{i-1}(\tilde{u}))^2
\]

(19)
FIGURE 3. Illustration of background segmentation. (a) Previous RGB image. (b) Current RGB image. (c) Current depth image in gray. (d) Clustering of 3D coordinates corresponding to current depth image. (e) Detected features (black circles denote static features while white circles denote dynamic features). (f) Depth image of static elements.

FIGURE 4. Histogram of compensated depth difference.

where \( \hat{u} \) is the warped pixel coordinate corresponding to \( u \) in the previous image.

The joint error is,

\[
E(T_{g,i}) = E_{geo}(T_{g,i}) + wE_{pho}(T_{g,i})
\]  

(20)

where \( w = 0.1 \) in line with other works. The joint error is minimized using the Gauss-Newton approach to obtain the 6-DoF rigid camera pose.

If the IMU data are available, we integrate the gyroscope data and use the integration as initialization for the rotational component of the relative camera pose. This can accelerate convergence of cost function minimization and reduce rotation drift.

E. VOLUMETRIC INTEGRATION

As the voxel-based representation has an advantage of computational efficiency, it can be used for 3D reconstruction on mobile devices. This method uses TSDF values to indicate a relative distance to the scanned surface. Truncation distance accounts for the uncertainty of depth measurement. Through this way the surface is encoded into a implicit function. Iso-surface with the value of zero is the reconstructed surface, which is exacted by the classic Marching Cubes algorithm [30]. In this section we first discuss the spatially hashed TSDF which is used on mobile devices briefly, then we introduce our modification to the integration procedure to enable it to manage dynamic scenes.

1) SPATIALLY HASHED TSDF

Voxels are organized as voxel blocks. A voxel block is composed of \( N \times N \times N \) voxels. Typically \( N \) is set to 8. In order to reduce large footprint, as in InfiniTAM, we create a line segment within \( d_i(u) \pm n \) along the pixel ray of \( u \). Here \( n \) is the TSDF truncation distance, which is always a multiple of voxel size. Voxel blocks on this line segment are allocated and are indexed by a hash table. Calculation of hash values is detailed in [16].
2) DEPTH INTEGRATION

Incoming depth images are used to allocate voxel blocks or update the TSDF value of allocated voxel blocks. The SDF value is calculated as,

\[ sdf_i = d_i(u) - V^{(z)} \]  

Here V depicts a voxel and the superscript \((z)\) means to selects Z-component of the voxel’s coordinate. In this equation, \( u \) is the projective pixel in the depth image using the camera pose calculated in the tracking procedure. If \( sdf_i > -n \), the TSDF value is updated using a weighted running average:

\[ tsdf_i = \frac{tsdf_{i-1}w_{i-1} + \min(1, \frac{d_i}{n})w_i}{w_{i-1} + 1} \]  
\[ w_i = w_{i-1} + 1 \]

where \( w \) is the weight of current value and is always set to be the number of observations.

The processing above is for integration of depth images captured in static scenes. We modify the traditional integration method so as to remove dynamic scenes. The dynamic removal procedures are in the following:

1) We do not allocate the voxel blocks on the line segments that are corresponding to dynamic pixels.

2) For allocated voxels which have no projective depth values in the current depth image, the voxels are regarded as dynamic. We set their TSDF values to be 1, which means they are dynamic. However, if static voxels are in the measuring range at first but then are out of the measuring range as the camera moves, this processing will set their TSDF values to be 1. To cope with this problem, the scanned scenes should be kept within the measuring range to prevent them from being filtered out.

3) For allocated voxels, if the difference between their projective depth in the newly measured depth image and their Z-components of coordinates is larger than a threshold, it means they lie in front of the measured depth. These cases are free-space violations. It means these voxels are dynamic. This process is depicted as follows. If \( d_i(u) - V^{(z)} \geq T_d \), we set the TSDF value of V to be 1. Here \( T_d \) is a threshold which is calculated as,

\[ T_d = \gamma \times \sigma_i(u) \]  

where \( \sigma_i(u) \) is the standard deviation of the depth noise and \( \gamma \) is a constant.

4) If the projective pixel of a voxel is labeled as dynamic, the voxel is either dynamic, or is static and occluded by the dynamic object. If the voxel is dynamic, its TSDF value needs to be set to 1. If the voxel is static, its TSDF value needs to remain unchanged. In our method, we determine whether a voxel is dynamic or static by the SDF value. If \( d_i(u) - V^{(z)} < -T_d \), it means the static voxel is occluded by the dynamic object and we keep the TSDF value unchanged. If \( abs(sdf_i - n \times tsdf_{i-1}) < T_d \), we replace the TSDF value by 1.

F. 3D RENDERING

3D rendering is to extract the point cloud on the surface for camera tracking as well as for visualization. The KinectFusion system brought in ray-casting method which is commonly used in computer graphics. Ray-casting aims to find the intersection of pixel rays with the surface. InfiniTAM makes many improvements to speed up processing on mobile devices. The improvements are the use of minimum and maximum searching length, various ray-casting step lengths, efficient calculation of surface normals and approximate ray-casting. These improvements are also adopted in our system and details can be found in [16].

IV. EXPERIMENT

We test our method both on TUM RGB-D dataset [31] (including static scenes, low dynamic scenes and high dynamic scenes) and on our dataset (dynamic scenes). When dynamic persons are sitting, the scenes are regarded as low dynamic scenes. When dynamic persons are walking, the scenes are regarded as high dynamic scenes. TUM dataset has provided a ground truth of the camera trajectory. Our dataset consists of indoor RGB-D images and IMU data, which is captured by an Occipital Structure Sensor and an Apple iPAD Air 2. In this section we firstly quantitatively evaluate the estimated camera poses on TUM dataset. Then we qualitatively evaluate the 3D reconstruction results both on TUM RGB-D dataset and on our dataset. At last we evaluate the runtime both on PC and on an Apple iPAD Air 2. On a desktop PC with an Intel Core E5-1620 v2 CPU clocked at 3.7GHz and a Nvidia GeForce GTX 1060 GPU, our system achieves up to 14FPS when processing VGA images. On an Apple iPAD Air 2 which has the A8X CPU clocked at 1.5GHz and the PowerVR GPU, our method is able to perform the 3D scanning at 8FPS when processing QVGA images.

A. QUANTITATIVE EVALUATION

In order to evaluate the camera tracking performance of our method, we compare the tracking accuracy obtained by our system with some of the recent methods such as VO-SF [21], ElasticFusion [9], Co-Fusion [23], StaticFusion [22] and InfiniTAMv3 [17] on the TUM RGB-D dataset. VO-SF is a visual odometry method and the others are dense 3D reconstruction methods. ElasticFusion and InfiniTAMv3 are designed to handle static scenes whereas the others are designed to handle dynamic scenes. InfiniTAMv3 can perform 3D reconstruction on-line on mobile devices and the other 3D reconstruction methods reconstruct the scanned scenes on PCs. For InfiniTAMv3, we run the open source code 2 and use the hybrid intensity + depth tracker. For other methods, we use the quantitative evaluation results in [22].

To quantitatively evaluate the camera tracking, we use the metrics proposed by Handa et al. [31] including translational/rotational relative pose root-mean-square error (RPE RMSE) metric and translational absolute trajectory root-mean-square
error (ATE RMSE) metric. RPE measures relative pose error for pose pairs with a distance of 1 second and ATE measures the Euclidean distances between the estimated camera poses and the ground truth poses. We add a noise to the relative rotations of the ground truth to simulate the relative rotations obtained by the integration of gyroscope data. These relative rotations are used in the background segmentation procedure. The noise subjects to Gaussian distribution,

\[
\mathcal{N}(\theta | \sigma_r) = \mathcal{G}(\theta; 0, \sigma^2_r)
\]  

(26)

where \( \sigma_r \) is the standard deviation of the Gaussian distribution. More specifically, the noise is added to the angle of the angle-axis representation and \( \sigma_r = 0.005^\circ \). Here 0.005\(^\circ\) indicates the standard deviation of the rotation drifts is 9\(^\circ\) per minute when the image is recorded at 30Hz.

Comparisons with other systems on the TUM dataset are shown in Table 1, Table 2 and Table 3. In these tables, the first columns indicate the property of the scanned scenes. Static stands for static scenes. Low stands for low dynamic scenes. High stands for high dynamic scenes. The sequence walking_halfsphere skips the initial 5 seconds of walking_halfsphere because the initial 5 seconds are of very high dynamic. BS means that we only use background segmentation and BS + DR means that we use both background segmentation and dynamic removal. For BS mode, dynamic elements are integrated into the 3D reconstruction, which in turn corrupt the frame-to-model pose estimation. Generally BS + DR performs better than BS. Our background segmentation method relies on the correctness of feature detection and matching whereas other dynamic handling methods rely on the correctness of image alignment. When dynamic elements take the largest proportion of the image such as in walking_xyz and walking_halfsphere, incorrect alignment would cause false segmentations in other methods. In this situation, our method is able to segment the images more accurately and has the best tracking performance. Though our experimental results are a little worse than ElasticFusion and StaticFusion in static scenes and low dynamic scenes, dynamic scenes. High stands for high dynamic scenes. The sequence walking_halfsphere skips the initial 5 seconds of walking_halfsphere because the initial 5 seconds are of very high dynamic. BS means that we only use background segmention and BS + DR means that we use both background segmentation and dynamic removal. For BS mode, dynamic elements are integrated into the 3D reconstruction, which in turn corrupt the frame-to-model pose estimation. Generally BS + DR performs better than BS. Our background segmentation method relies on the correctness of feature detection and matching whereas other dynamic handling methods rely on the correctness of image alignment. When dynamic elements take the largest proportion of the image such as in walking_xyz and walking_halfsphere, incorrect alignment would cause false segmentations in other methods. In this situation, our method is able to segment the images more accurately and has the best tracking performance. Though our experimental results are a little worse than ElasticFusion and StaticFusion in static scenes and low dynamic scenes.

### Table 1. Translational relative pose error (unit: cm/s) on the TUM dataset. Bold shows the best result.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Trans. RPE RMSE (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>InfiniTAMv3</td>
</tr>
<tr>
<td>Static</td>
<td></td>
</tr>
<tr>
<td>fr1/xyz</td>
<td>3.5</td>
</tr>
<tr>
<td>fr1/desk</td>
<td>5.6</td>
</tr>
<tr>
<td>fr1/desk2</td>
<td>7.9</td>
</tr>
<tr>
<td>fr1/plant</td>
<td>5.5</td>
</tr>
<tr>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>fr3/sitting_static</td>
<td>1.0</td>
</tr>
<tr>
<td>fr3/sitting_xyz</td>
<td>2.9</td>
</tr>
<tr>
<td>fr3/sitting_halfsphere</td>
<td>3.8</td>
</tr>
<tr>
<td>High</td>
<td></td>
</tr>
<tr>
<td>fr3/walking_static</td>
<td>2.8</td>
</tr>
<tr>
<td>fr3/walking_xyz</td>
<td>23.6</td>
</tr>
<tr>
<td>fr3/walking_halfsphere</td>
<td>12.9</td>
</tr>
<tr>
<td>fr3/walking_halfsphere*</td>
<td>X</td>
</tr>
</tbody>
</table>

### Table 2. Rotational relative pose error (unit: deg/s) on the TUM dataset. Bold shows the best result.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Rot. RPE RMSE (deg/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>InfiniTAMv3</td>
</tr>
<tr>
<td>Static</td>
<td></td>
</tr>
<tr>
<td>fr1/xyz</td>
<td>1.44</td>
</tr>
<tr>
<td>fr1/desk</td>
<td>2.37</td>
</tr>
<tr>
<td>fr1/desk2</td>
<td>4.46</td>
</tr>
<tr>
<td>fr1/plant</td>
<td>1.61</td>
</tr>
<tr>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>fr3/sitting_static</td>
<td>0.42</td>
</tr>
<tr>
<td>fr3/sitting_xyz</td>
<td>0.98</td>
</tr>
<tr>
<td>fr3/sitting_halfsphere</td>
<td>3.12</td>
</tr>
<tr>
<td>High</td>
<td></td>
</tr>
<tr>
<td>fr3/walking_static</td>
<td>0.49</td>
</tr>
<tr>
<td>fr3/walking_xyz</td>
<td>5.32</td>
</tr>
<tr>
<td>fr3/walking_halfsphere*</td>
<td>2.81</td>
</tr>
<tr>
<td>fr3/walking_halfsphere</td>
<td>X</td>
</tr>
</tbody>
</table>
Table 3. Translational absolute trajectory error (unit: cm) on the TUM dataset. Bold shows the best result.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>InfiniTAMv3</th>
<th>ElasticFusion</th>
<th>VO-SF</th>
<th>Co-Fusion</th>
<th>StaticFusion</th>
<th>Ours (BS)</th>
<th>Ours (BS+DR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fr1/xyz</td>
<td>2.0</td>
<td>1.2</td>
<td>5.1</td>
<td>1.4</td>
<td>1.4</td>
<td>2.0</td>
<td>1.9</td>
</tr>
<tr>
<td>fr1/desk</td>
<td>8.8</td>
<td>2.1</td>
<td>5.6</td>
<td>17.7</td>
<td>2.3</td>
<td>7.8</td>
<td>7.4</td>
</tr>
<tr>
<td>fr1/desk2</td>
<td>12.7</td>
<td>5.7</td>
<td>17.4</td>
<td>16.8</td>
<td>5.2</td>
<td>8.1</td>
<td>8.9</td>
</tr>
<tr>
<td>fr1/plant</td>
<td>5.6</td>
<td>5.3</td>
<td>7.8</td>
<td>12.6</td>
<td>11.3</td>
<td>5.5</td>
<td>5.3</td>
</tr>
<tr>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fr3/sitting_static</td>
<td>1.0</td>
<td>0.8</td>
<td>2.9</td>
<td>1.1</td>
<td>1.3</td>
<td>1.1</td>
<td>1.1</td>
</tr>
<tr>
<td>fr3/sitting_xyz</td>
<td>4.6</td>
<td>2.2</td>
<td>11.1</td>
<td>2.7</td>
<td>4.0</td>
<td>5.0</td>
<td>4.5</td>
</tr>
<tr>
<td>fr3/sitting_halfsphere*</td>
<td>3.8</td>
<td>42.8</td>
<td>18.0</td>
<td>3.6</td>
<td>4.0</td>
<td>3.8</td>
<td>3.7</td>
</tr>
<tr>
<td>High</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fr3/walking_static</td>
<td>3.2</td>
<td>29.3</td>
<td>32.7</td>
<td>55.1</td>
<td>1.4</td>
<td>3.1</td>
<td>2.9</td>
</tr>
<tr>
<td>fr3/walking_xyz</td>
<td>29.1</td>
<td>90.6</td>
<td>87.4</td>
<td>69.6</td>
<td>12.7</td>
<td>14.9</td>
<td>12.6</td>
</tr>
<tr>
<td>fr3/walking_halfsphere*</td>
<td>22.7</td>
<td>48.6</td>
<td>48.2</td>
<td>75.6</td>
<td>6.3</td>
<td>16.1</td>
<td>15.4</td>
</tr>
<tr>
<td>fr3/walking_halfsphere</td>
<td>X</td>
<td>63.8</td>
<td>73.9</td>
<td>80.3</td>
<td>39.1</td>
<td>36.2</td>
<td>33.6</td>
</tr>
</tbody>
</table>

Our method can reconstruct the scenes using voxel-based representations. Besides, our handling of dynamic scenes is more efficient and can be used on mobile devices. Compared with InfiniTAMv3 that can be used on mobile devices, our method has better performance.

Camera trajectories of walking_xyz and walking_halfsphere* are shown in Fig. 5. We can see that our method generates more accurate trajectories than InfiniTAMv3. In order to evaluate the robustness to the noise in relative rotations, we compare the translational ATE RMSE both on walking_xyz and on walking_halfsphere*. We add noises of $\sigma_{r} \in [0^\circ, 0.05^\circ]$ with a step of $0.005^\circ$ to the relative rotations calculated by the ground truth rotations. Results are shown in Fig. 6. When adding less than $0.05^\circ$ Gaussian noise to the relative rotations between consecutive images, the translational ATE RMSE rises very slowly when the standard deviation of noise rises. Results show that our method is not sensitive to Gaussian noise.

B. QUALITATIVE EVALUATION

As the TUM dataset has not provided the ground truth 3D model of the scanned scenes, we evaluate the 3D reconstruction results of our method qualitatively. We compare the 3D reconstruction results with InfiniTAMv3. Qualitative results are shown in Fig. 7. From the results we can see that using background segmentation improves camera tracking accuracy and applying dynamic removal generates a clear and accurate 3D reconstruction. Besides, our method generates a continuous model while StaticFusion generates a surfel model. We also test our method on sequences captured by a hand-held device. 3D reconstruction results are shown in Fig. 8 and Fig. 9. Sequence 1 is captured by low dynamic sensors while sequence 2 is captured by high dynamic sensors. In Fig. 8, when the man moves from the first position to the second position, our method sets voxels of the dynamic...
Y. Liu et al.: 3D Scanning of High Dynamic Scenes Using an RGB-D Sensor and an IMU on a Mobile Device

FIGURE 7. 3D reconstruction results after integrating the first 240 images of walking halfsphere sequence. (a) Reconstructed model of StaticFusion. (b) Reconstructed model of InfiniTAMv3. (c) Reconstructed model of our method (BS + DR).

FIGURE 8. 3D reconstruction results (front view) of our sequence 1. (a)-(c) Three examples of RGB images. (d)-(g) Reconstructed models of InfiniTAMv3, ElasticFusion, Co-Fusion and StaticFusion. (h) Reconstructed model of our method (BS + DR).

elements to free space while InfiniTAMv3 and ElasticFusion integrates them multiple times. Co-Fusion detects and reconstructs dynamic elements. StaticFusion removes part of dynamic elements in the reconstructed model. However, compared with our method, these two methods generate less complete models. In Fig. 9, when the man is walking away and the sensors are moving from left to right, our method determines camera poses of smaller drift and generates more accurate reconstructions than InfiniTAMv3. ElasticFusion, Co-Fusion and StaticFusion cannot generate complete models as they use surfels to represent the models. Besides, if the camera is moving fast, these methods fail to fuse the depth images.

In order to illustrate how the dynamic removal procedure manages dynamic objects during 3D reconstruction, in Fig. 10 we show the 3D rendering results during integration of the walking halfsphere sequence. When the man is walking from left to right, dynamic features are detected through the analysis of corresponding compensated depth difference histograms and our background segmentation method can detect dynamic objects effectively. The BS mode can track the camera poses more accurately compared with InfiniTAMv3. Furthermore, the BS + DR mode is able to set dynamic voxels to free-space and succeeds in not to integrate dynamic pixels into the voxel-based representation. Because TSDF values of voxels that have no projective depth values are set to be 1, the lack of depth information on depth discontinuities or on the black objects causes black holes in (j)-(l). These holes can be filled up when the depth information is available again.

Our background segmentation would fail if the image is mainly composed of dynamic elements that have consistent motion and most features are on the dynamic elements. Static features would be marked as dynamic because that statistically the number of them is fewer. An example of this problem is shown in Fig. 11. The body of the man has consistent motion and most detected features are located on the body of the man. Even the histogram has Gaussian distribution, the highest value corresponds to depth difference caused by
the motion of the man. Our method inaccurately marks static elements as dynamic. Furthermore, high dynamic would cause motion blur. In this case our method would detect very little even no features on dynamic elements and may label high dynamic clusters as static.

C. RUNTIME ANALYSIS

To complete the efficiency evaluation of our method, in Table 4 we show the average computational time for different procedures processed both on PC and on iPad Air 2. Images in the TUM dataset have a resolution of 640 x 480 and sequences captured by us is of 320 x 240. Geometric clustering and background segmentation are processed on CPU while camera tracking and depth integration are processed on GPU. In our method, feature detection and tracking in each cluster cost most of the time during background segmentation. There exists a speed up of our method in the future.

We also compare the runtime of background segmentation procedure among different methods in Table 5. As VO-SF utilizes multi-core programming to achieve a runtime of about 80ms, we only compare with Co-Fusion and StaticFusion in this table. StaticFusion costs more time than ours when processing VGA images. Runtime of Co-Fusion increases significantly when more objects are added to the list of active models. Generally our method is more efficient than other

---

**TABLE 4. Average computational time (unit: ms) of our method.**

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Geometric Clustering</th>
<th>Background Segmentation</th>
<th>Tracking and Integration</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>fr3/walking_xyz (640 x 480, PC)</td>
<td>22.6</td>
<td>38.7</td>
<td>9.5</td>
<td>70.8</td>
</tr>
<tr>
<td>fr3/walking_halfsphere (640 x 480, PC)</td>
<td>24.1</td>
<td>39.9</td>
<td>9.1</td>
<td>73.1</td>
</tr>
<tr>
<td>our sequence1 (320 x 240, iPad Air 2)</td>
<td>21.4</td>
<td>48.3</td>
<td>63.3</td>
<td>133.0</td>
</tr>
<tr>
<td>our sequence2 (320 x 240, iPad Air 2)</td>
<td>23.1</td>
<td>49.6</td>
<td>65.7</td>
<td>138.4</td>
</tr>
</tbody>
</table>

**TABLE 5. Average computational time of background segmentation procedure for different methods.**

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Co-Fusion</th>
<th>StaticFusion</th>
<th>Our Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>fr3/walking_xyz (640 x 480, PC)</td>
<td>64.6</td>
<td>104.6</td>
<td>61.3</td>
</tr>
<tr>
<td>fr3/walking_halfsphere (640 x 480, PC)</td>
<td>82.9</td>
<td>106.3</td>
<td>64.0</td>
</tr>
</tbody>
</table>
FIGURE 10. 3D rendering during integration of walking_hallsphere sequence. (a)-(c) Detected features (black circles denote static features while white circles denote dynamic features). (d)-(f) 3D rendering results during integrating of InfiniTAMv3. (g)-(i) 3D rendering results during integrating of our method (BS). (j)-(l) 3D rendering results during integrating of our method (BS + DR). (m)-(o) Depth images of static elements. (p)-(r) Corresponding histograms of compensated depth difference.
methods when the scanned scenes contain high dynamic elements.

V. CONCLUSION

As most of 3D scanning methods are designed to handle static scenes, in this paper we propose a 3D scanning method which can segment input depth images into static or dynamic, and integrate depth images into a voxel-based representation. Our method is efficient and is able to be used on mobile devices. One of our contribution is a novel background segmentation method with the use of sparse optical flow and IMU data. The other contribution lies in that we modify the traditional integration method and dynamic voxels are set to free-space. As far as we know, we are the first to reconstruct the static backgrounds of dynamic scenes using voxel-based representations. Experiments show that compared with some state-of-the-art systems, our method has better results when scanning high dynamic scenes and has comparable results when scanning low dynamic and static scenes. Our method is efficient and achieves frame rates 8FPS on an Apple iPad Air 2.

Nevertheless, our method has some problems. The first problem is that our segmentation method relies on sparse features and sparse optical flow. High dynamic may cause motion blur in the RGB image. In this case, no enough features may be detected in blurred clusters. Even if enough features are detected in a cluster, features may spread over the regions that are not blurred. Our method may label high dynamic clusters as static. The second problem is that if most features are on dynamic elements that have consistent motion, our method may segment images incorrectly. In our future work, we would explore the feasibility of using the accelerometer data to segment the dynamic elements.

REFERENCES


WEI GAO received the B.S. degree in computational mathematics and the M.S. degree in pattern recognition and intelligent systems from Shanxi University, in 2002 and 2005, respectively, and the Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences, in 2008. He has been with the Robot Vision Group, National Laboratory of Pattern Recognition, Institute of Automation, since 2008, where he is currently an Associate Professor. His current research interests include computer vision, 3D reconstruction, and image processing.

ZHANYI HU received the B.S. degree in automation from the North China University of Technology, Beijing, China, in 1985, and the Ph.D. degree in computer vision from the University of Liège, Belgium, in 1993. Since 1993, he has been with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, where he is currently a Professor. His research interests include robot vision, which include camera calibration and 3D reconstruction, and vision guided robot navigation. He was the Local Chair of ICCV2005, an Area Chair of ACCV2009, and the PC Chair of ACCV2012.

YANGDONG LIU received the B.S. degree from the North University of China, in 2012, and the M.S. degree from the Beijing University of Posts and Telecommunications, in 2015. He is currently pursuing the Ph.D. degree with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences. His research interests include computer vision, 3D reconstruction, and image processing.

WEI GAO received the B.S. degree in computational mathematics and the M.S. degree in pattern recognition and intelligent systems from Shanxi University, in 2002 and 2005, respectively, and the Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences, in 2008. He has been with the Robot Vision Group, National Laboratory of Pattern Recognition, Institute of Automation, since 2008, where he is currently an Associate Professor. His current research interests include computer vision, 3D reconstruction, and image processing.