

DUDMap: 3D RGB-D mapping for dense, unstructured, and dynamic environment

Özgür Hastürk¹ and Aydan M Erkmen²

Abstract

Simultaneous localization and mapping (SLAM) problem has been extensively studied by researchers in the field of robotics, however, conventional approaches in mapping assume a static environment. The static assumption is valid only in a small region, and it limits the application of visual SLAM in dynamic environments. The recently proposed state-of-the-art SLAM solutions for dynamic environments use different semantic segmentation methods such as mask R-CNN and SegNet; however, these frameworks are based on a sparse mapping framework (ORB-SLAM). In addition, segmentation process increases the computational power, which makes these SLAM algorithms unsuitable for real-time mapping. Therefore, there is no effective dense RGB-D SLAM method for real-world unstructured and dynamic environments. In this study, we propose a novel real-time dense SLAM method for dynamic environments, where 3D reconstruction error is manipulated for identification of static and dynamic classes having generalized Gaussian distribution. Our proposed approach requires neither explicit object tracking nor object classifier, which makes it robust to any type of moving object and suitable for real-time mapping. Our method eliminates the repeated views and uses consistent data that enhance the performance of volumetric fusion. For completeness, we compare our proposed method using different types of high dynamic dataset, which are publicly available, to demonstrate the versatility and robustness of our approach. Experiments show that its tracking performance is better than other dense and dynamic SLAM approaches.

Keywords

Dynamic mapping, visual SLAM, localization, 3D reconstruction

Date received: 9 February 2021; accepted: 21 April 2021

Topic Area: Vision Systems

Topic Editor: Antonio Fernandez-Caballero

Associate Editor: Shengyong Chen

Introduction

Simultaneous localization and mapping (SLAM) is to produce a consistent map of environment and to estimate the pose in the map using noisy range sensor measurements. SLAM problem has been extensively studied by researchers in the field of robotics. After the appearance of Kinect, there are many solutions, which fuse the color image and depth map. Visual SLAM produces a sparse solution by relying on points matching, whereas direct methods can produce a dense reconstruction by

¹Advanced Control Technologies Department, ROKETSAN Missile Industries Inc., Ankara, Turkey

²Electrical and Electronics Engineering Department, Middle East Technical University, Ankara, Turkey

Corresponding author:

Özgür Hastürk, Advanced Control Technologies Department, ROKETSAN Missile Industries Inc., Kemalpaşa Mahallesi Şehit Yüzbaşı Adem Kutlu Sokak No: 21, 06780 Elmadağ, Ankara, Turkey.

Email: ohasturk@roketsan.com.tr



Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (<https://creativecommons.org/licenses/by/4.0/>) which permits any use, reproduction and distribution of the work without

further permission provided the original work is attributed as specified on the SAGE and Open Access pages (<https://us.sagepub.com/en-us/nam/open-access-at-sage>).

minimization of the photometric error. However, none of the above methods addresses the problem of dynamic objects in the environment.

Conventional approaches in mapping assume that the environment is static. Although the static assumptions are valid in a small region, change is inevitable when dynamic elements exist or large-scale mapping is necessary. By classifying dynamic content as outliers, a small fraction can be managed. However, SLAM problem in highly dynamic scenes is still not solved completely because there is no suggested framework found in the literature.

Another biggest difficulty in robot navigation is unstructured environment. In unstructured environments, it is not easy to find discrete geometries because of noisy edge or plane. Significant research has been carried out for unstructured environments, especially in the field of autonomous navigation, and a number of effective approaches have been developed using laser range finder. However, there is no effective RGB-D SLAM method for real-world unstructured and dynamic environments.

In this study, we propose DUDMap (see https://www.dropbox.com/s/lsexrz82ewdzo0w/DUDMAP_sample.mp4?dl=0): dense, unstructured, and dynamic mapping. Our approach requires neither explicit object tracking and object classifier nor purely geometric method in contrast to recent approaches discussed by Yu et al.¹ and Taneja et al.,² which makes it robust to any type of moving object. Furthermore, we assume a dynamic environment consisting of static and dynamic classes having generalized Gaussian distribution to detect dynamics. We reconstruct scene geometry using signed distance function (SDF) instead of surfels. This makes our method to easily create a dense final mesh and such representation is useful in robotic applications because it defines the distance to surface.

The main contribution of this article is a novel and an effective SDF-based SLAM algorithm that is resistant to dynamics and also the following:

- We identify the dynamics using image registration residual combining with Gaussian mixture model. The number of dynamic objects does not limit our approach because we do not employ any type of moving object detection and tracking.
- Our method generates a final intense 3D mesh without using semantic information or object classifier. We eliminate repeated views and use only consistent data for decreasing the required computational power.
- We compare our method with other state-of-the-art systems using TUM dataset,³ together with other high dynamic datasets including Bonn,⁴ VolumeDeform,⁵ and CVSSP RGB-D dataset⁴¹ (used with permission), which are publicly available, showing the superior performance of our approach.
- To evaluate the outdoor performance of our method, we use commercially available ZED camera for map

generation and dynamic filtering. Experiments illustrate that our method produces consistent result both in indoor and outdoor applications. These are demonstrations of real-world unstructured dynamic environments of our approach.

The rest of this article is organized as follows. The second section reviews state-of-the-art visual SLAM methods that attack the problem of dynamic environments. The third section is devoted to the overall structure of our system by giving details about proposed approaches for local key-frame extraction and dynamic removal. The fourth section shows the experiments conducted and gives the evaluation result by comparing our method against other state-of-the-art methods, whereas the fifth section provides concluding remarks.

State-of-the-art methods

ORB-SLAM2⁷ (latest version ORB-SLAM3⁸), S-PTAM,⁹ and RTAB-Map¹⁰ are the best state-of-the-art feature-based visual SLAM approaches in static environments. To increase the performance of such feature-based method in dynamic environment, dynamic objects are considered generally as spurious data, and dynamic object is removed as outliers using RANdom SAMple Consensus (RANSAC) and robust cost function. On the other hand, targeted attempts are still being made to increase performance in dynamic scenes. For instance, DVO-SLAM¹¹ uses photometric and depth errors instead of visual features. The joint visual odometry scene flow¹² proposes an efficient solution to estimate the camera motion. However, odometry-based methods either cannot recover from inaccurate image registration or lacks a loop closure detection approach independent of pose estimate.

SDFs have long been studied to represent the 3D volumes in computer graphics.^{13–15} Newcombe et al.³⁸ proposed the SDF-based RGB-D mapping by generating precise maps in static environments. Elastic fusion (EF)¹⁶ is another method based on SDF, which can work in small scenarios. CoFusion (CF)¹⁷ is a contemporary method for reconstructing several moving objects, however, it works with slow camera motions only and its performance deteriorates significantly with increasing camera speed. Static fusion (SF)¹⁸ simultaneously estimates the camera motion together with dynamic segmentation of the image. However, it works only sequences without having high dynamics at the beginning. Palazzolo et al.⁴ propose refusion, where dynamics detection is done using the residuals obtained from the registration on SDF. This approach can create a consistent mesh of the environment, however, highly dynamical change deteriorates mapping performance.

Some methods use motion consistency to validate tracked points, where dynamic objects are segmented generally as spurious data since they conflict with the motion consistency of background over consecutive frames. For

instance, Wang and Huang¹⁹ segment dynamic objects using RGB optical flow. Nevertheless, the algorithm is still not robust enough for TUM high dynamic scenarios. Kim and Kim²⁰ propose to use the difference between depth images to eliminate the dynamics in the scene. However, this algorithm requires an optimized background estimator suitable for parallel processing. Azartash et al.²¹ use the image segmentation for discrimination of the moving region from the static background. Experimental results show that the accuracy remains almost the same in low dynamic scenarios. Tan et al.²² use an adaptive RANSAC for removing outliers. This method can work in dynamic situations with a limited number of slowly moving objects.

Other methods use classifiers to identify the dynamic objects. Kitt et al.²³ combine the motion estimation with object detection; however, this method requires a classifier, which makes this method inapplicable to online explorations. Bescos et al.²⁴ propose DynaSLAM, which combines a prior learning by mask region-based convolutional neural network (R-CNN)²⁵ and multiview geometry to segment dynamic content. Multiview geometry consists of region growth algorithm, which makes it unsuitable for real-time operation even running on NVIDIA Titan GPU. Mask fusion²⁶ also uses mask R-CNN for semantic segmentation. DS SLAM,²⁷ RDS-SLAM,²⁸ and semantic SLAM²⁹ are other semantic-based algorithms, which use the SegNet.¹ Pose fusion,³⁰ implemented on EF, uses open pose CNN³¹ for human pose detection, which limits this method in the nonhuman dynamic object scenes. Flow fusion³² uses optical flow residuals with PWC-Net³³ for dynamic and static human objects. However, such approaches are relying heavily on prior training methods. Therefore, if an unlearned dynamic occurs in camera view, estimation results are bigger. Furthermore, learning-based semantic information is time-consuming with heavy computational burden.

In our work, we reconstruct our scene geometry using SDF instead of surfels in contrast to EF and SF, and therefore, we can directly generate the mesh of the environment using such representation without using object tracking and object classifier. Moreover, a number of dynamic objects or their speeds do not limit our approach.

Our proposed methodology

Preliminaries and notations

In our approach, we denote a 3D point as $[X, Y, Z] \in \mathbb{R}^3$, rotation of the camera, and translation $R \in SO(3)$, $T \in \mathbb{R}^3$, respectively. At time t , RGB-D frame contains an RGB image and a depth map. The homogenous point $X = (x, y, z, 1)^T$ can be computed by assuming a pinhole camera model with intrinsic parameters f_x, f_y, c_x , and c_y (focal length and optical center) such as

$$\begin{bmatrix} \frac{x - c_x}{f_x} z, \frac{y - c_y}{f_y} z, z, 1 \end{bmatrix}^T \quad (1)$$

The 3D point corresponding to a pixel is reconstructed as

$$\begin{bmatrix} \frac{xf_x}{z} + c_x, \frac{yf_y}{z} + c_y \end{bmatrix}^T \quad (2)$$

In rigid body motion, the common representation matrix H consisting of a 3×3 rotation matrix and 3×1 translation vector T

$$H_{4 \times 4} = \begin{bmatrix} R_{3 \times 3} & T_{3 \times 1} \\ 0_{1 \times 3} & 1_{1 \times 1} \end{bmatrix} \quad (3)$$

is used in the transformation of a point \vec{X} under motion as

$$X' = H_{4 \times 4} X \quad (4)$$

The rotation matrix R has nine parameters and if we were to estimate the camera motion, we have to solve these nine parameters by forming a constrained optimization problem, which can be very slow to implement. The Lie algebra allows us lower dimensional linear space for rigid body motion representation, making it popular in computer vision problems.

We use a Lie algebra $SE(3)$ representation as twist coordinates ξ as in the literature³⁴ because the rigid motion has six degrees of freedom while transformation matrix T has 12 parameters. Using the Lie algebra representation, rigid body motion can be written as

$$\xi = \begin{bmatrix} 0 & -\omega_3 & \omega_2 & v_1 \\ \omega_3 & 0 & -\omega_1 & v_2 \\ -\omega_2 & \omega_1 & 0 & v_3 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (5)$$

Figure 1 depicts the important steps of our proposed method. We first apply a depth filter to eliminate significant amounts of noise in raw depth images. To eliminate redundant data in fusion process, we trim repeated camera views by measuring the similarity ratio of RGB images. We then perform pose estimation and continue the process by detecting the dynamic elements in the scene. The subsequent subsections provide the details of each block in our proposed system.

Depth smoothing and feature matching

Commercially available RGB-D cameras usually produce invalid depth measurements. In addition, there exist significant amounts of noise in raw depth images. In this study, we use a depth adaptive bilateral filtering⁴² method because it modifies the weighting to account for variation of intensity. Figure 2 depicts the original depth image, smoothed image, and filtered image, respectively. In addition, we change the zero values in the original depth images by neighboring 5×5 pixel mean value in smoothing process.

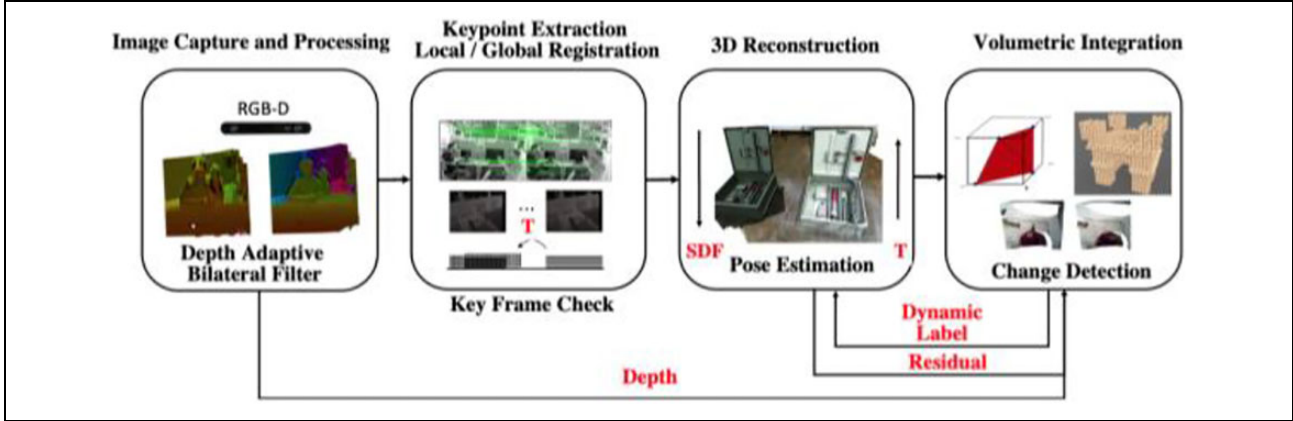


Figure 1. Our proposed scheme.

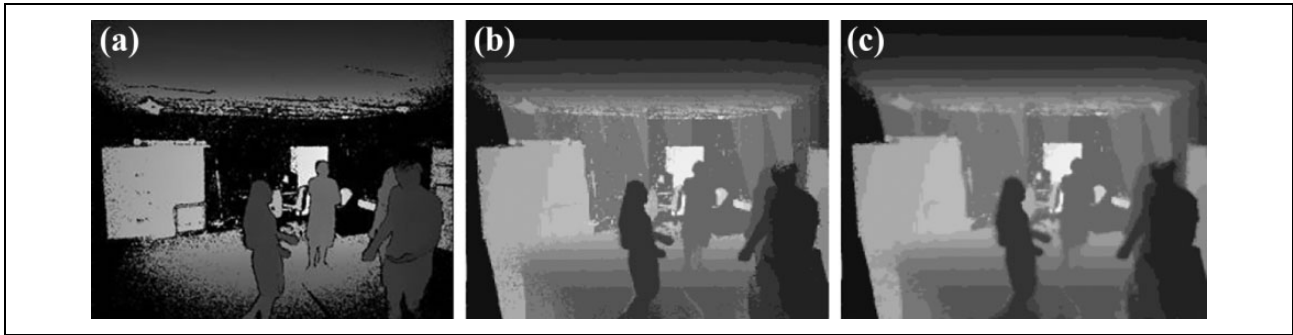


Figure 2. (a) Original image, (b) smoothed image, and (c) filtered image.

SDF fusion is an averaging process, therefore, it is important not to use redundant data in the fusion process because small error makes the SDF model as unclear. To eliminate redundant camera views, we perform similarity ratio test based on feature matching. A typical feature matcher consists of the following steps: extracting local feature, matching features using nearest-neighbor approach, and selecting good correspondences.

In the literature, scale-invariant feature transform (SIFT) is being proposed for extracting keypoints and widely used in different applications. SIFT feature-matching works well for scaled images but fails some cases such as faces with pose changes.³⁶ Application of feature matching method FLANN with SIFT descriptor overcomes such disadvantages of SIFT. In similarity analysis, we use FLANN-based feature matching with SIFT descriptor and we use RATIO³⁷ to select good correspondences that compare the lowest feature distance and the second lowest feature distance for recognizing good ones. Similarity ratio of the VolumeDeform “boxing” sequence is depicted in Figure 3. Since the ratio is not high, which indicates low degree similarity, all the frames are included in the mapping process. On the other hand, BONN dataset “crowd2” sequence is a high dynamic sequence having 895 frames. If 80% similarity threshold is utilized, 78 frames are skipped, which results in 8.7% decrease in computational time.

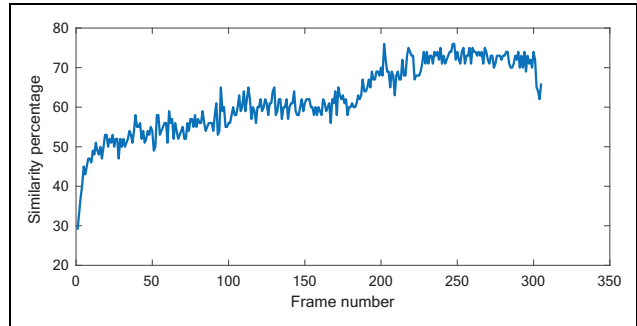


Figure 3. VolumeDeform boxing sequence similarity ratio.

Absolute translational error increases only 2.2%, while rotational relative pose error increases by 0.3%. In low dynamic sequences, the number of similar frames will be higher, which decreases the unnecessary computational power. This is the novel enhancement we provide to existing methods in the literature for the betterment of the performance. We use the 80% similarity threshold.

Pose estimation

We can represent the geometry using SDF. To reconstruct the scene, we fuse incrementally RGB-D data into SDF and geometry is stored in voxel grid (see Figure 4 for SDF calculation). First, camera pose is estimated using SDF,

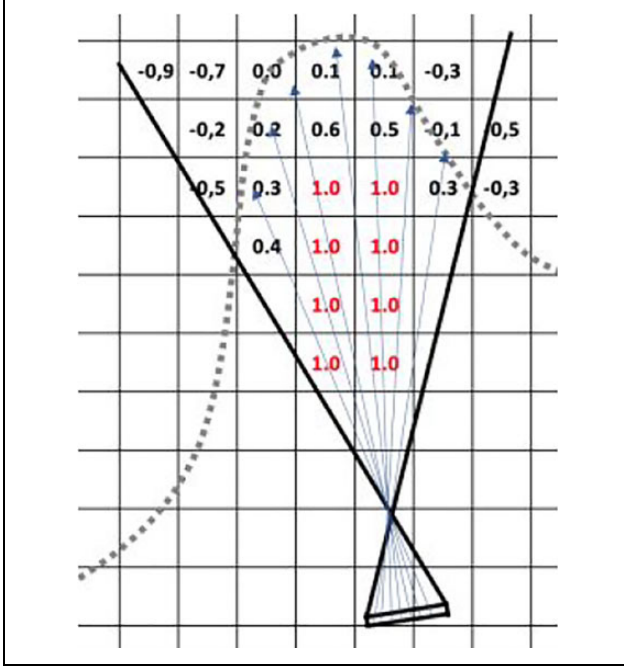


Figure 4. Illustration of SDF calculation and zero SDF function on the surface (grids represent the voxel border). SDF: signed distance function.

and SDF is updated based on newly computed camera pose. In the literature, most of the volumetric fusion techniques, for example, KinectFusion¹³ use synthetic depth images and align them using iterative closest point. However, we use the camera pose directly on the SDF because SDF encrypts the 3D geometry of the environment.

Assuming independent and identical distributed pixels with Gaussian noise in depth values, the likelihood of observing a depth image is

$$p(D|R, T) = \prod_{i,j} e^{-\psi(Rx_{i,j} + T)^2} \quad (6)$$

To find the camera poses that maximize this likelihood, we define a pose error function as

$$E_p(R, T) = \sum_{i,j} \psi(Rx_{i,j} + T)^2 \quad (7)$$

A rigid-body motion can be described in Lie algebra with the 6D twist coordinates $\xi = (\omega_1, \omega_2, \omega_3, v_1, v_2, v_3)$. If we rewrite the error function (7), then it becomes

$$E_p(\xi) = \sum_{i,j} \psi_{i,j}(\hat{\xi})^2 \quad (8)$$

$$\psi_{i,j}(\xi) = \psi(Rx_{i,j} + T) \quad (9)$$

If image registration is correct with the 3D model, the projected colors should be consistent as well. We incorporate this condition by adding an extra term. Since there is no absolute reference of the image for comparison, color value

Algorithm 1. Pose estimation algorithm.

Input : Joint error function

Output : Pose

```

1:   begin
2:     Initialize parameters
3:     Calculate Jacobian
4:     Initialize non-negative correction factor as Grammian
      of Jacobian
5:     while (pose difference) > 0.001 or iteration # < 5 do
6:       Find  $\delta$  increment for  $(J^T J + \lambda(J^T J))\delta = J^T f$ 
7:       Update pose with increment
8:       if objective function is minimum
9:         return pose
10:      else
11:        Update correction factor
12:        Increment iteration number
13:    end

```

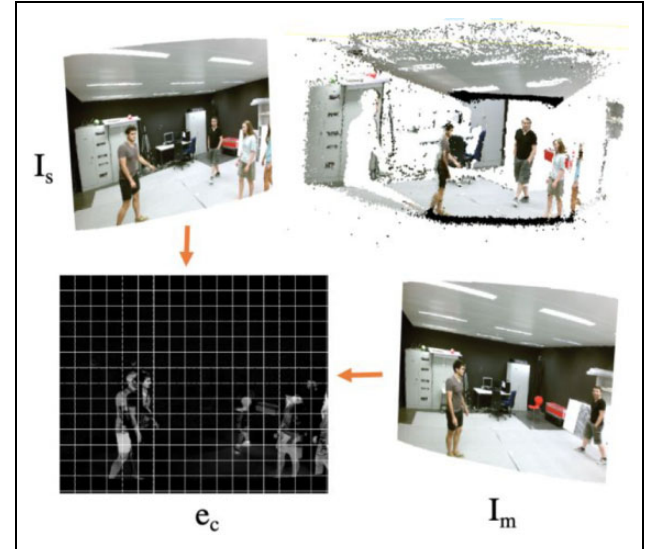


Figure 5. Inconsistency map of two images (EPFL RGB-D pedestrian dataset sequence frame 250 and 278).

stored in the voxels is used. Using color intensities of the pixels and corresponding voxels, the error function becomes

$$E_c(\xi) = \sum_{i,j} \left(V_I(\hat{\xi}) - I(p_i) \right)^2 \quad (10)$$

The joint error function is given in equation (11) with v intensity contribution with respect to the depth

$$E(\xi) = E_p(\hat{\xi}) + E_c(\hat{\xi}) \quad (11)$$

and start by linearizing ψ around initial pose estimate $\hat{\xi}$ using Jacobian matrix



Figure 6. (a) RGB image, (b) reconstruction error, and (c) dynamic label image.

$$J_p(\xi) = \frac{\partial \psi_{ij}(\hat{\xi})}{\partial \hat{\xi}} \frac{\partial (\hat{\xi})}{\partial} \quad (12)$$

$$\frac{\partial (\hat{\xi})}{\partial} = \begin{bmatrix} 0 & v3 & -v2 & 1 & 0 & 0 \\ -v3 & 0 & v1 & 0 & 1 & 0 \\ v2 & -v1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

In equation (12), $\frac{\partial \psi_{ij}(\hat{\xi})}{\partial \hat{\xi}}$ is computed numerically by evaluating the gradient. After computing the Jacobian $J_c(\xi)$ by following a similar procedure, we adopt to use Levenberg–Marquardt algorithm because Gauss–Newton cannot calculate the best optimal estimate, resulting in nonminimum function value. Levenberg–Marquardt algorithm can handle this problem in the form of

$$[A + \lambda I] \Delta = -b \quad (13)$$

where λ is the non-negative correction factor updated at each iteration. In our case, A matrix and b vector become

$$\begin{bmatrix} A = A_p + A_c \\ b = b_p + b_c \\ A_p = \sum_i J_{p,i}^T J_{p,i} \\ A_c = \sum_i J_{c,i}^T J_{c,i} \\ b_p = \sum_i J_{d,i} \psi_{ij}(\hat{\xi}) \\ b_c = \sum_i J_{c,i} (V_I(\hat{\xi}) - I(p_i)) \end{bmatrix} \quad (14)$$

Algorithm 1 summarizes the pose estimation process.

We solve equation (14) iteratively until difference $(\xi(k+1) - \xi(k))$ is small enough or maximum iteration number is reached. To increase real-time performance, we conduct all calculations on the GPU in parallel since vectors b and matrices A are independent of each other.

Algorithm 2. Dynamic labeling algorithm.

Input : Depth image, prior segmentation from residual error, initial label class

Output : Segmented depth image with label

```

1: Initialize parameters
2: Find maximal cliques
3: Construct k-neighborhoods
4: Partition into parallel threads
5: do each EM iteration
6:   for each neighborhood of the subgraph do in parallel
7:     E-step
8:     M-step
9:   end for
10:  Update parameters
11: while Likelihood increment < threshold
12: return Label set

```

Algorithm 3. DUDMap.

Input : Depth image, RGB image

Output : Artificial camera view, mesh (optional)

```

1: Initialize parameters for sensor, camera tracking, SDF
2: if frame number = initial frame
3:   Initialize poses as identity
4:   for i ∈ N (number of pixel)
5:     Initialize labels as static
6:   end for
7:   Pose estimation using matrix exponential
8: else
9:   RGB similarity check
10:  Pose estimation using matrix exponential
11:  Generate label set
12:  Re-pose estimation using matrix exponential with label set
13: Volume integration
14:  Update parameters
15: while Frame number < total number of frame
16:  Extract mesh
17: return Final mesh

```

Table 1. TUM dataset—translational RPE (RMSE, cm/s).

Mapping type		Dense Map							Sparse map				
CNN utilization		No CNN							Segmentation CNN				
Dynamic	Sequence	VO-SF ¹²	EF ⁴⁰	CF ¹⁷	RF ⁴	SF ¹⁸	ORB_SLAM ⁸	DUDMap	MF ²⁶	DS-SLAM ¹	Dyna-SLAM ²⁴	Semantic SLAM ²⁹	RDS-SLAM ²⁸
Low	Sit static	2.4	0.9	1.1	2.1	1.1	1	1.5	1.7	0.8	1.3	0.9	1.2
Low	Sit xyz	5.7	1.6	2.7	3.8	2.8	—	3.7	4.6	—	—	—	—
High	Walk stat	10.1	26.0	22.4	1.7	1.3	78	2.6	3.9	1.0	0.9	1.0	2.2
High	Walk xyz	27.7	24.0	32.9	11.8	12.1	42.6	10.3	9.7	3.3	2.1	2.2	4.3
High	Walk half	33.5	20.5	40.0	6.4	20.7	32.7	4.7	9.3	3.0	2.9	2.8	4.8
Mean error	All sequences	15.9	14.6	19.82	5.2	7.6	30.9 ^a	4.6	5.8	2.0 ^a	2.4 ^a	2.3 ^a	3.2 ^a
		18.4 ^a	17.9 ^a	24.1 ^a	5.5 ^a	8.8 ^a		4.8 ^a	6.2 ^a				

VO-SF: visual odometry scene flow; EF: elastic fusion; SF: static fusion; CF: CoFusion; RMSE: root mean square error; CNN: convolutional neural network; RPE: relative pose error.

^a Mean value excluding *sit/xyz*, all methods are given with corresponding paper in the reference set.

Table 2. TUM dataset—translational RPE (RMSE, °/s).

Mapping type		Dense Map							Sparse map				
CNN utilization		No CNN							Segmentation CNN				
Dynamic	Sequence	VO-SF ¹²	EF ⁴⁰	CF ¹⁷	RF ⁴	SF ¹⁸	ORB_SLAM ⁸	DUDMap	MF ²⁶	DS-SLAM ¹	Dyna-SLAM ²⁴	Semantic SLAM ²⁹	RDS-SLAM ²⁸
Low	Sit static	0.7	0.3	0.4	0.6	0.4	0.3	0.4	0.5	0.3	0.3	0.3	0.3
Low	Sit xyz	1.4	0.6	1.0	1.3	0.9	—	1.2	1.3	—	—	—	—
High	Walk stat	1.7	4.8	4.0	1.1	0.4	6	0.6	0.8	0.3	0.3	0.3	0.5
High	Walk xyz	5.1	4.8	5.6	2.7	2.7	7.9	2.3	2.0	0.8	0.7	0.6	0.9
High	Walk half	6.7	6.4	13	3.0	5.0	7.2	2.3	3.4	0.8	0.8	0.7	1.9
Mean error	All sequences	3.1	3.4	4.8	2.2	1.9	5.4 ^a	1.4	1.6	0.6 ^a	0.5 ^a	0.48 ^a	1.2 ^a
		3.6 ^a	4.1 ^a	5.8 ^a	1.9 ^a	2.1 ^a		1.4 ^a	1.7 ^a				

VO-SF: visual odometry scene flow; EF: elastic fusion; SF: static fusion; CF: CoFusion; RMSE: root mean square error; CNN: convolutional neural network.

^a Mean value excluding *sit/xyz*, all methods are given with corresponding paper in the reference set.

Signed distance function representation and 3D reconstruction

We use the discrete voxel grid to represent the SDF. Signed-distance value is calculated by trilinear interpolation of eight neighboring pixels. We project each voxel onto the image plane instead of ray casting because this process is suitable for parallel processing since each voxel is independent of its neighbors. Since the operation has to be carried out for each voxel, GPU is used for this operation. Finally, we implement marching cubes algorithm³⁵ to extract the triangle mesh. In RGB-D mapping approaches, storing the SDF in a 3D grid requires a large amount of memory. Therefore, we use a special memory allocation technique proposed by Nießner et al.³⁹ In this technique, we only allocate the voxels in required areas, which enable to scan the large areas with limited memory.

Dynamic detection

Let I_m and I_s be the instantaneous image of the generated model and source, respectively. The error in color map denoted by e_c as

$$e_c = |I_{s \leftarrow m} - I_s| \quad (15)$$

If the images I_m and I_s are accurately registered and if there is no change in the geometry, the resulting error would be zero (Figure 5). In general, minimizing equation (15) results in a sufficient image registration. SDF represents the distance to the nearest surface, and therefore, we select to use SDF as an error function. The error in the depth can be written as

$$e_p(\hat{\xi}) = \sum_{i=1}^N \left\| \psi_i(\hat{\xi}) \right\|^2 \quad (16)$$

In equation (16), N is the pixel number, ψ is the SDF, and $\hat{\xi}$ is the matrix exponential multiplied by the 3D point corresponding to the i 'th pixel p_i , computed using equation (1).

After performing an initial registration using equation (15), we compute for each pixel and its residual as defined in equation (17)

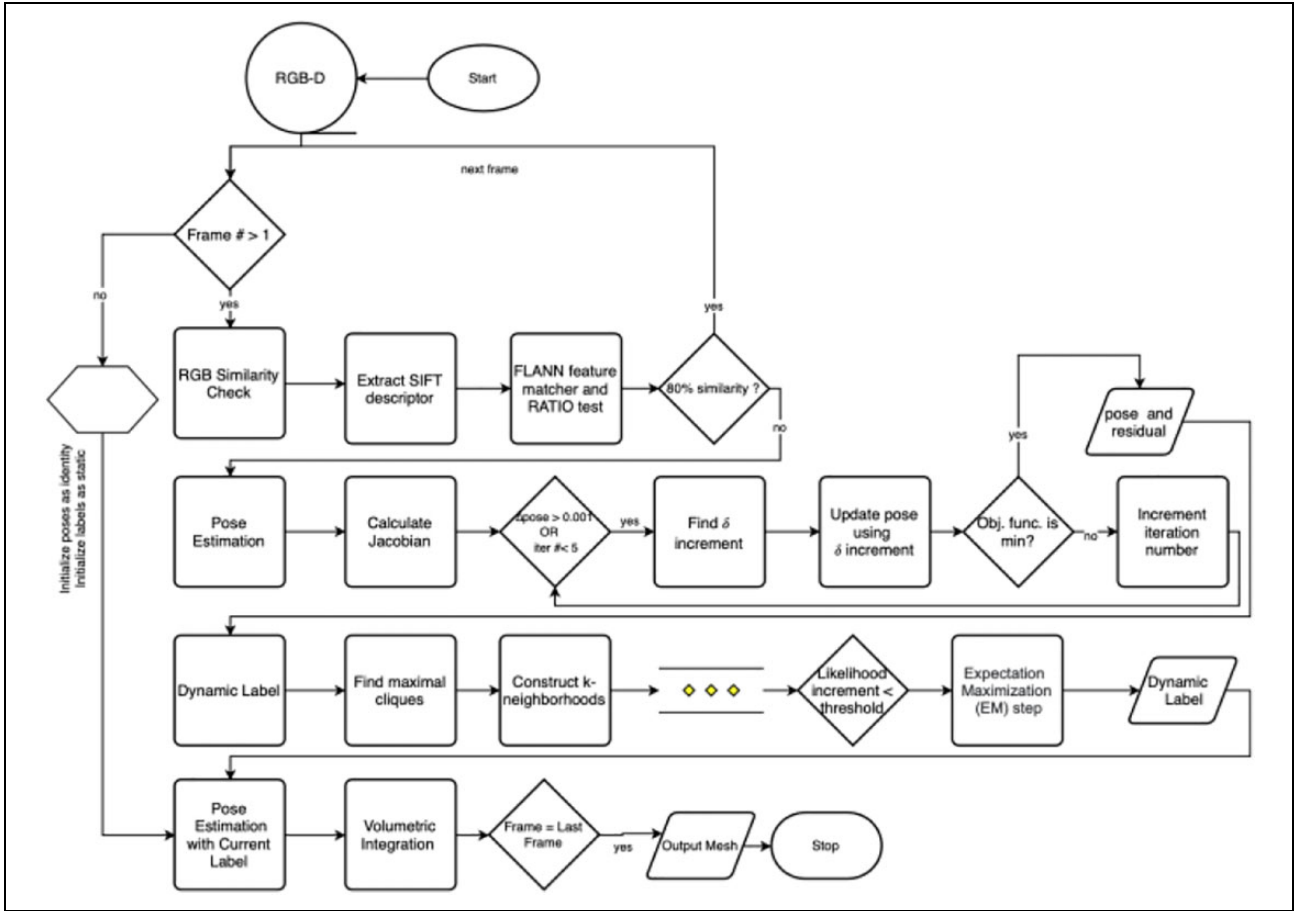
$$r_i = \left\| \psi_i(\hat{\xi}) \right\|^2 \quad (17)$$

Table 3. TUM dataset—translational ATE (RMSE, cm).

Dynamic	Sequence	VO-SF ¹²	EF ⁴⁰	CF ¹⁷	RF ⁴	SF ¹⁸	ORB-SLAM ⁸	DUDMap	MF ²⁶	DS-SLAM ¹	Dyna-SLAM ²⁴	Semantic SLAM ²⁹	RDS-SLAM ²⁸
Low	Sit static	33	0.8	1.1	0.9	1.3	0.9	1	2.1	0.7	1.1	0.8	0.8
Low	Sit xyz	11	2.2	2.7	4	4	—	2.6	3.1	—	—	—	—
High	Walk stat	33	29	55	1.7	1.4	36	1.8	3.5	0.8	0.7	0.8	2
High	Walk xyz	87	91	69	9.9	12	92	7.4	10	2.5	1.6	1.6	5.7
High	Walk half	74	64	80	11	39	65	7.1	11	3	3	2.5	8
Mean error	All sequences	48	37	41	5.4	12	48.4 ^a	4.0	5.9	1.8 ^a	1.6 ^a	1.4 ^a	4.1 ^a
		56.8 ^a	46 ^a	51 ^a	5.9 ^a	13.4 ^a		4.3 ^a	6.7 ^a				

VO-SF: visual odometry scene flow; EF: elastic fusion; SF: static fusion; CF: CoFusion; ATE: absolute trajectory error; RPE: relative pose error; RMSE: root mean square error.

^aMean value excluding sit/xyz, all methods are given with corresponding paper in the reference set.

**Figure 7.** Flowchart of the proposed algorithm, DUDMap.

The residual obtained after image registration is used as for dynamic detection (Figure 6). Our aim is to compute the binary labeling for each element according to occurred changes. For example, $l_i = 0$ indicates consistency and $l_i = 1$ shows the presence of change in corresponding voxel i .

If $h(d)$ be the histogram of the image, our problem is in the form of binary classification problem using a dynamic label threshold. Then, probability density function can be defined as the combination of two density functions related class label as

$$p(D) = \sum_{i=1}^2 P_i p_i(D|l_i) \quad (18)$$

using class conditional densities and prior probabilities. To calculate an estimate of dynamic change, we maximize $p(l/D)$

$$l^* = \underset{l}{\operatorname{argmax}} \{L(D|l)\} \quad (19)$$

where $L(D|l)$ is the log likelihood of the two-component mixture and it can be written as

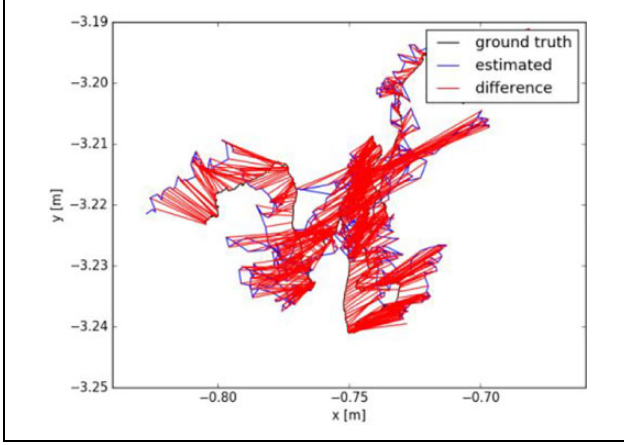


Figure 8. ATE/RPE of TUM fr3/walking static sequence. ATE: absolute trajectory error; RPE: relative pose error.

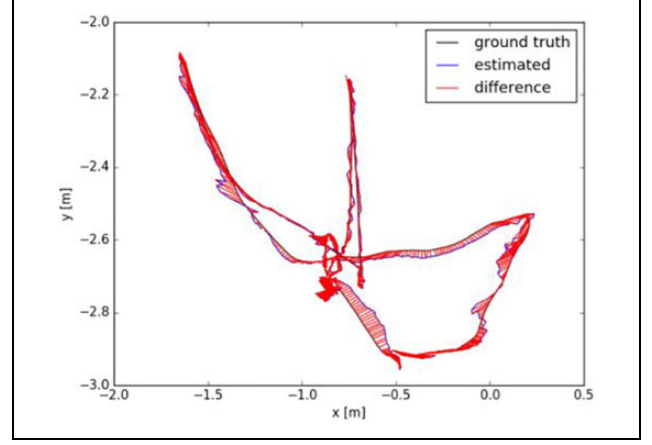


Figure 10. ATE/RPE of TUM fr3/walking halfsphere sequence. ATE: absolute trajectory error; RPE: relative pose error.

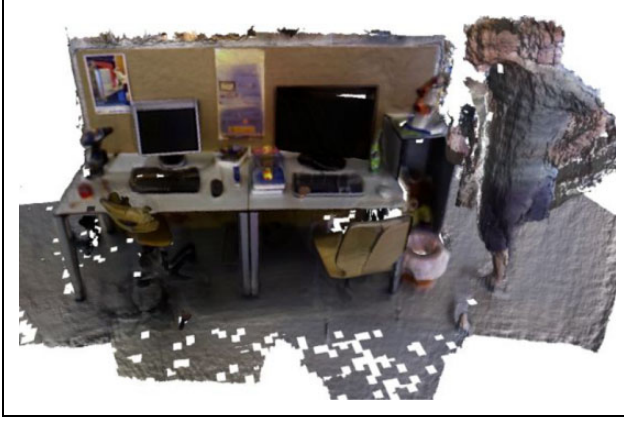


Figure 9. Mesh of TUM fr3/walking static sequence.



Figure 11. Mesh of TUM fr3/walking halfsphere sequence.

$$L(D, l) = \sum_{x=0}^{L-1} h(d) \ln p(D|l) \quad (20)$$

The final log-likelihood function is in the form of

$$L(D, U, l) = \sum_{i=1}^2 \sum_{d=0}^{L-1} h(d) u_i(d) \ln \{P_i p_i(d|l_i)\} \quad (21)$$

$$p(d|l) = H(y) \frac{2}{\rho_c \sqrt{2\pi}} e^{-\frac{y^2}{2\rho_c^2}}$$

In equation (21), $u(d)$ is the indication of static or dynamic component.

After dynamic label identification and updating the label grid (Algorithm 2), a second pose estimation and registration are performed using newly obtained label set (Algorithm 3). However, we must filter out dynamic labels that originated from noise. We compare the SDF value of new observation with the previous static reconstruction and compute the difference δ_L . Applying a threshold θ , we obtain the label grid such that

$$l_i = \text{dynamic if } \delta_L > \theta \quad (22)$$

Figure 7 shows the overall flowchart of our proposed methodology including RGB similarity check, pose estimation, and dynamic detection.

Experiments

Our proposed method is able to operate in dynamic environments without requiring any dynamic object detection and tracking. Our experiments support our main claims, which are as follows:

- Robustness to dynamic elements regardless of their quantity and speed of change in the environment.
- That approach requires no explicit object tracking, object classifier and generate a consistent a dense model of the environment.

The experiments were conducted on a workstation computer Intel i7 running at 3.20 GHz and a GeForce 1070 GPU using Ubuntu 16.04. Our default parameters have been determined empirically so that a sensitivity analysis is performed on change of parameters.

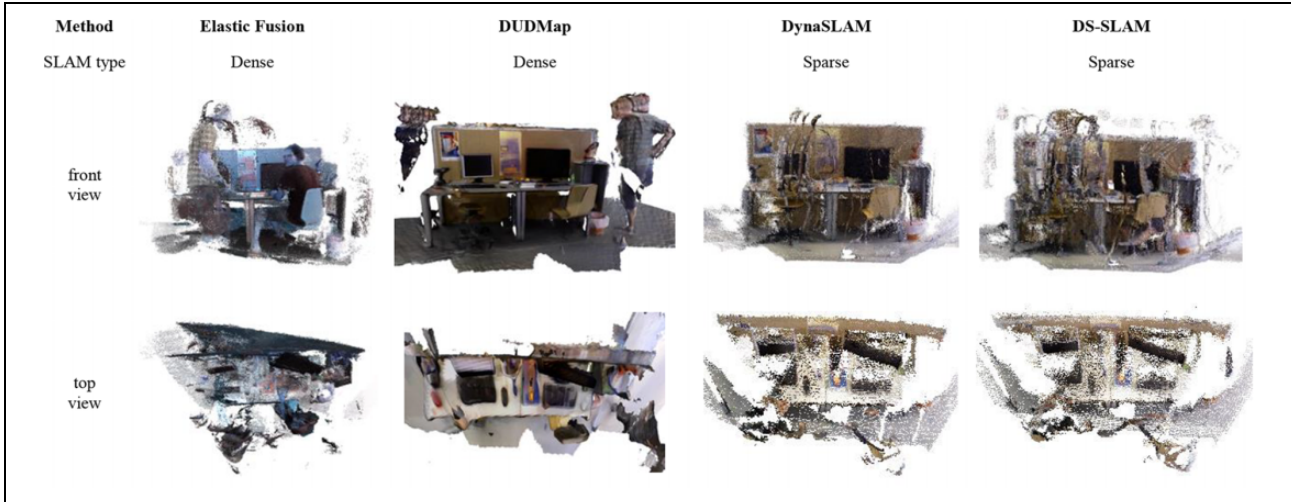


Figure 12. Scene reconstruction of fr3/walking xyz sequence.

Table 4. TUM dataset—execution time.

Method	Semantic	GPU	Dynamic label	Time for per frame (ms)
ORB-SLAM3	—	—	—	22–30
DS-SLAM	SegNet	P4000	38 ms	Feature extraction: >9.3 Consistency check: >29 Segmentation: >38 Total: >75 ms
DynaSLAM	Mask R-CNN	Intel i7-8750 CPU only Tesla M40	2582 ms 200 ms	Total: >2600 ms ²⁷ Multiview geometry: >200 Background inpaint: >120
RDS-SLAM	SegNet	RTX 2080Ti	30 ms	Total: >300 ms
DUDMap	—	GTX 1070	8.4 ms	Similarity check: <7.1 Pose estimation: <10.3 Dynamic label: <8.4 Total: <50 ms
Elastic fusion	—	GTX 780Ti	—	<66 ms
Mask fusion	—	GTX TitanX	200 ms	<60 ms

TUM RGB-D dataset

In this dataset, walking sequences are highly dynamic and complex because moving objects cover almost all camera views. Sitting sequence is low dynamic and there exists a person sitting and moving their arms. In this dataset, the evaluation is performed through the metrics proposed by Sturm et al.³ as translational, rotational relative pose error (RPE), and translational absolute trajectory error (ATE). Obtained results of dense visual SLAM methods are listed in Tables 1 to 3. In the TUM dataset, the ground-truth trajectory is obtained from a high-accuracy motion-capture system with eight high-speed tracking cameras (100 Hz). Therefore, quantitative evaluation is possible regarding the accuracy of pose estimation. However, TUM dataset has no exact 3D model of the environment, therefore, we can evaluate the 3D reconstruction performance

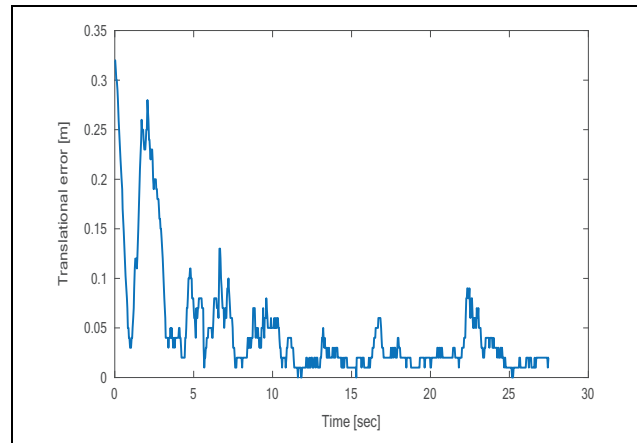


Figure 13. Relative translational error (walking-xyz) of our method.

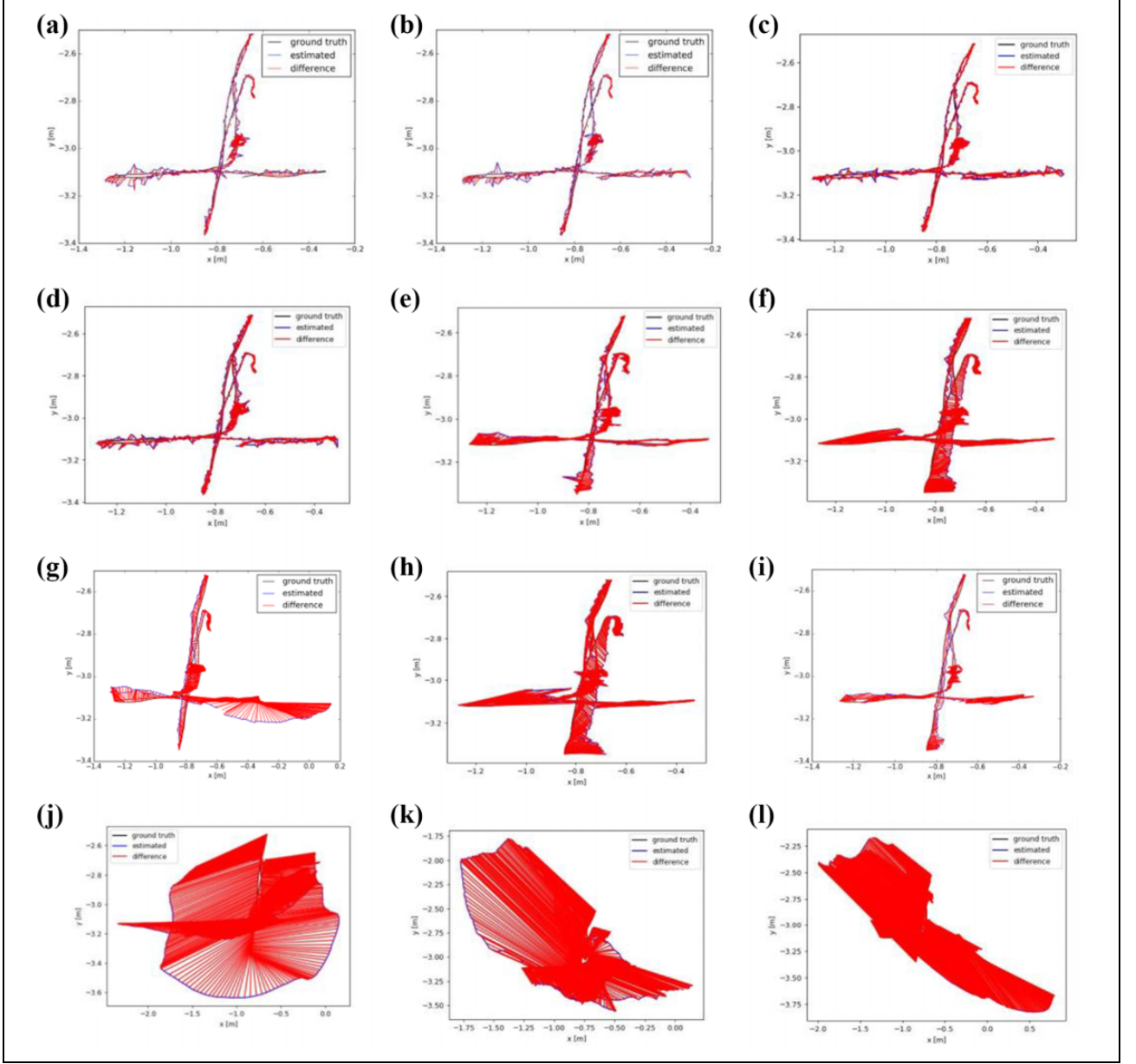


Figure 14. Comparison of estimated trajectories of TUM fr3/walking xyz sequence. (a) DynaSLAM: Sparse, mask R-CNN; (b) DS-SLAM: Sparse, SegNet CNN; (c) Semantic SLAM: Sparse, Blitznet CNN; (c) RDS-SLAM: Sparse, mask R-CNN/SegNet; (d) pose fusion: Dense, open pose CNN; (e) flow fusion: Dense, Pwc.Net CNN; (f) Refusion: Dense, no CNN; (g) Static fusion: Dense, no CNN; (h) DUDMap: Dense, no CNN; (i) Elastic fusion: Dense, no CNN; (j) ORBSLAM3: Sparse, no CNN; (k) VO-SF: Dense, no CNN. VO-SF: visual odometry scene flow.

results of our method qualitatively. Qualitative results are shown in Figures 9, 11, and 12. Figure 12 also shows the scene reconstruction result of fr3/walking xyz sequence obtained using EF, DynaSLAM, and DS-SLAM.

As given in Table 1, our proposed scheme achieves an average translation RPE of 0.045 m/s, which is considerably lower than other dense methods such as VOSF, EF, SF, and mask fusion. Our aim is to develop a dense RGB-D SLAM algorithm without using high computational power in dynamic environments. According to Tables 1 to 3, our method achieves smaller relative and translational error

than other dense methods. For all high dynamic sequences, our method reaches the lowest RPEs except for the “fr3/walk stat” sequence. In a highly dynamic scene, our proposed method produces better results for the following reasons:

EF is not capable of dynamics in the sequences. Hence, dynamic object deteriorates the 3D mesh and pose estimation. CF works well for slow camera motions but its performance deteriorates noticeably when the speed of the camera increases. SF works sequences with limited dynamics at the beginning, and therefore, it produces large

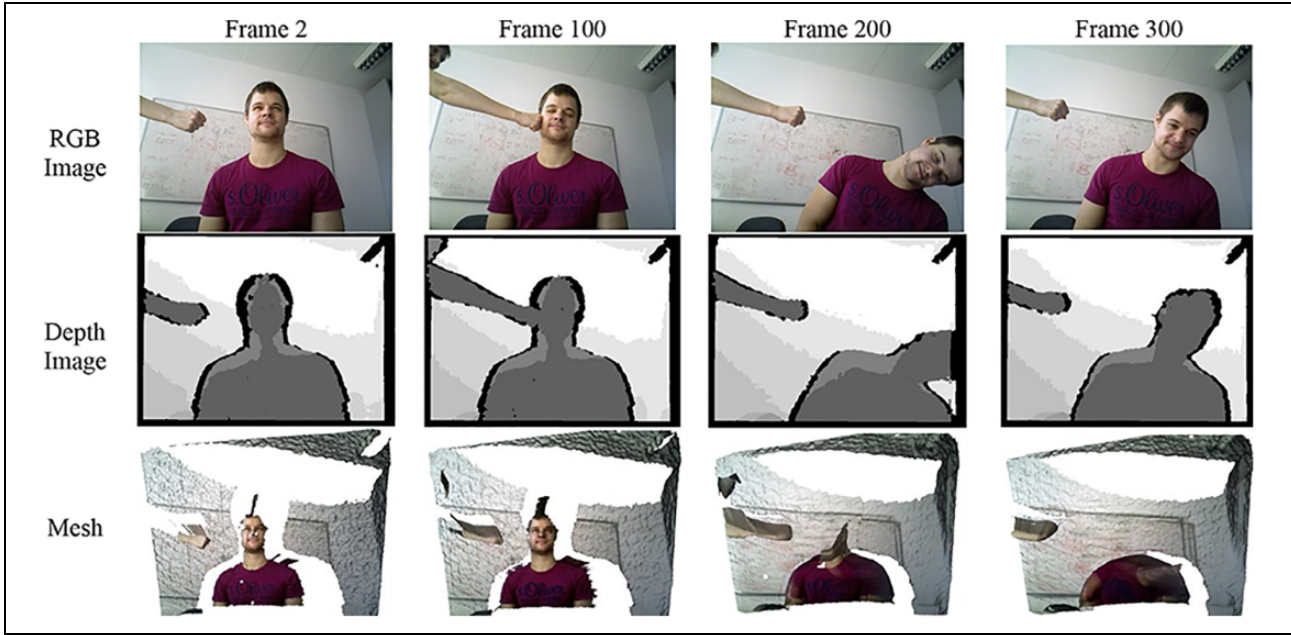


Figure 15. RGB-D image and mesh of VolumeDeform boxing sequence of our proposed method.



Figure 16. Mesh of BONN moving obstructing box sequence.

Table 5. BONN dataset—translational RPE (RMSE, cm/s).

Dynamic	Sequence	RF ⁴	SF ¹⁸	DynaSLAM ²⁴	DUDMap
High	Balloon tracking2	0.32	0.37	0.19	0.27
High	Obstruction box	0.34	0.33	0.54	0.17

SF: static fusion; RPE: relative pose error; RMSE: root mean square error.

Table 6. VolumeDeform dataset results.

Dynamic	Sequence	Trans, RPE RMSE (cm/s)	Trans, RPE RMSE (°/s)	Trans, ATE RMSE (cm)
High	Boxing	0.32	0.37	0.19
High	Sunflower	0.34	0.33	0.54

ATE: absolute trajectory error; RPE: relative pose error; RMSE: root mean square error.

errors on a highly dynamic environment. In general, existing high dynamics in the scene leads to blurry motion in the image, resulting inconsistent mesh.

In addition, according to Tables 1 to 3, there is no doubt that semantic-based visual SLAM methods have better results in ATE and RPE criteria. However, such method does not provide a dense model and it is relying heavily on the prior result from the learning techniques. If an unlearned condition exists in the camera view, the estimation result is highly influenced.

Table 4 compares the execution time of our proposed method with semantic-based SLAM algorithms. Most of the modern segmentation-based SLAM methods are built on ORBSLAM, therefore, it is included in timing analysis. The execution time data are obtained from the corresponding published papers. DynaSLAM has a good tracking performance, however, mask R-CNN makes this method unsuitable for real-time operation.

If a lightweight semantic segmentation such as Seg.Net is used, as in DS-SLAM and RDS-SLAM, the required time for per frame for segmentation decreases from 200 ms to 30 ms. However, an unlearned dynamics in the camera field-of-view results in pose error, leading to moving object to be mapped as a static object. Our method without using any semantic label criteria runs almost constant rate regardless of moving object type and speed. In addition, our method does not require high-end graphic units.

Figure 12 shows that a person remains in the model because the model built has artifact in the “walking xyz” sequences. This situation also occurs in “walking half-sphere” (Figure 11) and “walking static”(Figure 9) sequences because the camera is tracking a person initially, and finally, the camera never looks again, hence, it is not

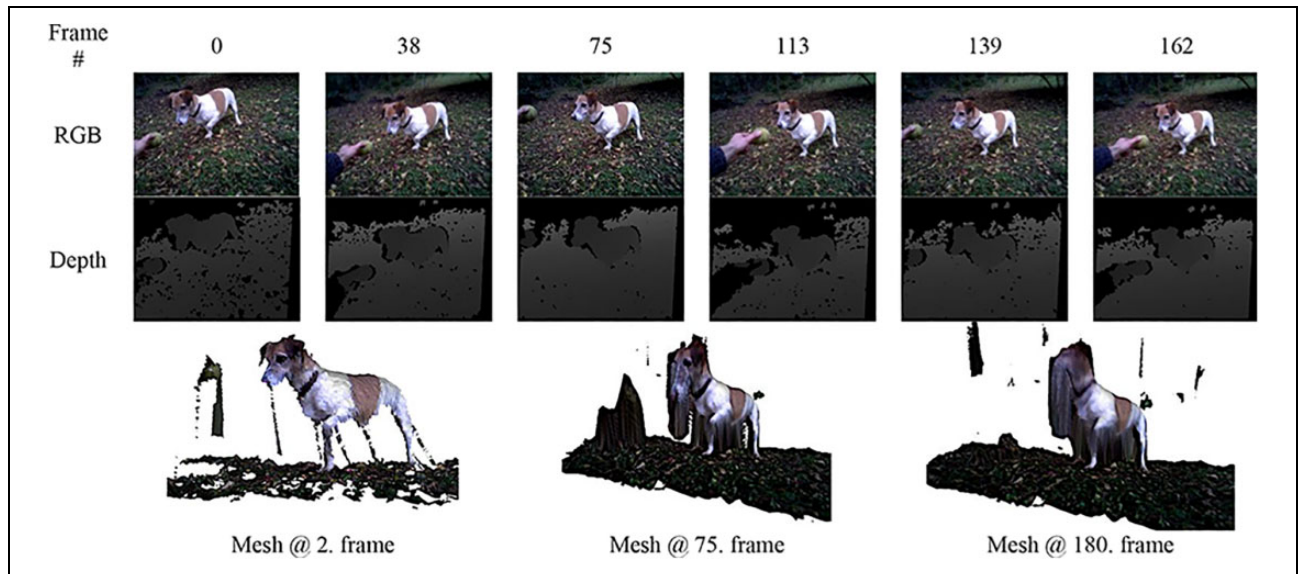


Figure 17. RGB-D image and final mesh of the CVSP “dog” sequence.

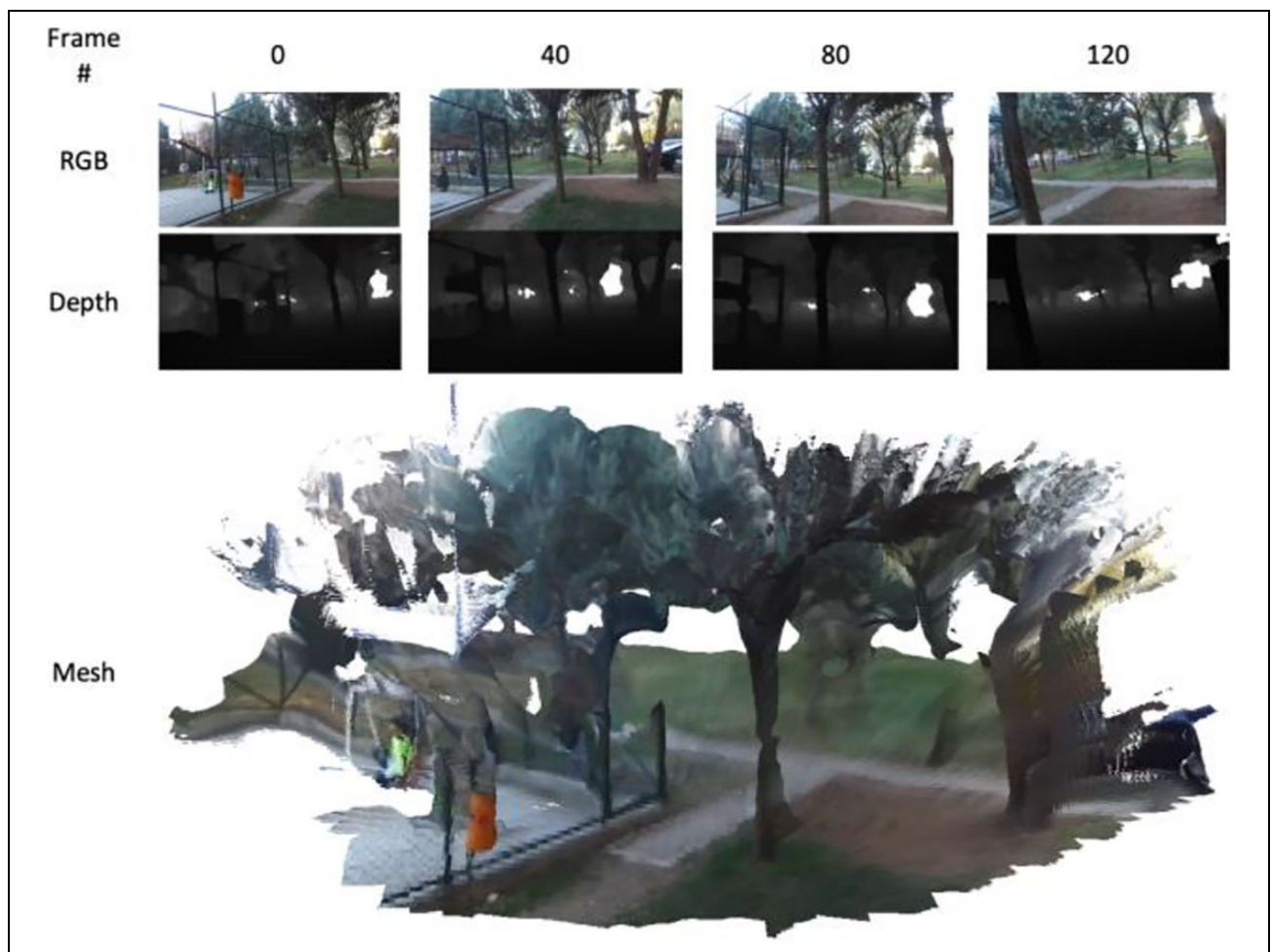


Figure 18. RGB-D image and final mesh of the “outdoor-1” sequence.

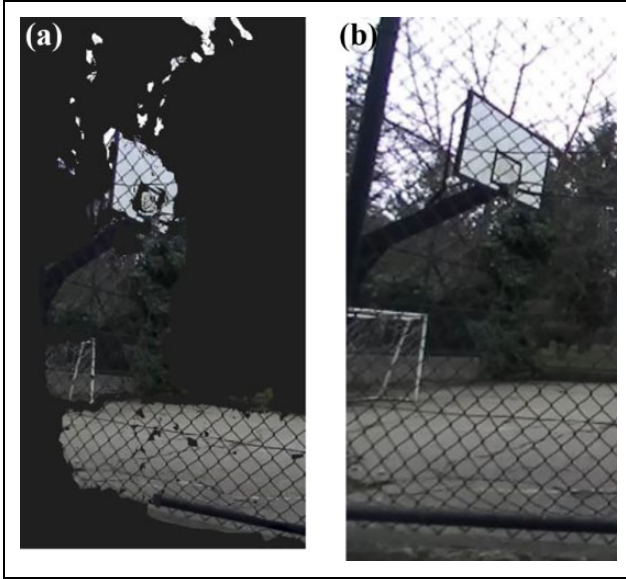


Figure 19. (a, b) Outdoor-I sequence grid fence mapping result.

possible to identify that the voxels are free. Figure 13 also confirms such a case. It is clear that translational error is higher at the beginning when the camera tracks the person.

Figure 8 and 10 depict the ATE/RPE of TUM “fr3/walking static” and “fr3/walking halfsphere” sequences. In addition, Figure 14 shows the estimated trajectory result of fr3/walking xyz sequence obtained by the state-of-the-art visual SLAM system. Trajectory results are consistent with Tables 1 to 3. Semantic-based visual SLAM methods except pose fusion and flow fusion have better results in ATE and RPE criteria. Our proposed method can compete with semantic SLAM and RDS-SLAM, however, DynaSLAM and DS-SLAM have the best estimate. However, our method has the best result among the dense and CNN-free methods.

Bonn RGB-D dynamic dataset

We compare methods on the dynamic scenes of Bonn dataset published by Palazzolo et al.⁴ This dataset has a variety of sequences. For example, “moving_obstructing_box” scene assesses the kidnapped camera problem, where the camera is moved to a different location, whereas “balloon_tracking” has uniformly colored balloon having no features on it. Figure 16 shows the resulting mesh of BONN moving obstructing box sequence.

Table 5 presents that DynaSLAM outperforms the other methods in balloon tracking. However, it has poor performance on the obstructing box scene. Since DynaSLAM is the combination of neural network and geometric approach, the available semantic information on scene helps to increase the performance.

VolumeDeform dataset

VolumeDeform is an RGB-D dataset for the purpose of real-time nonrigid reconstruction and is used for evaluation of the nonrigid object reconstruction algorithms at real-time rates.⁵ Since dynamic datasets for evaluating RGB-D SLAM method with exact trajectory are limited, this dataset is used to measure the elimination capability of our method to handle dynamic parts in the scene. Figure 15 illustrates the moving object elimination capability of the proposed method by using VolumeDeform boxing sequence. In addition, results of pose error and trajectory error are listed in Table 6.

CVSSP RGB-D dataset

“CVSSP dynamic RGB-D dataset has RGB-D sequences of general dynamic scenes captured using the Kinect V1/V2 and two synthetic sequences.”⁶ This dataset is designed for nonrigid reconstruction. “Dog” sequence is selected because there exists little clearly distinct geometry in the environment with nonrigid dynamic object. In this sequence, the dynamic part is the movement of the arm of the person and the head of the dog. The exact value of the trajectory and reference 3D model of the environment are not provided, therefore, we evaluated the 3D mesh result qualitatively. As the frame number increases, our proposed method successfully eliminates dynamic in the frame (Figure 17).

Outdoor mapping performance

We used the ZED camera in a hand-held setup for acquiring RGB-D images. We captured the frame in a resolution of 1280×720 with a rate of 30 fps. To measure the 3D mapping performance of our proposed approach, default camera properties and standard settings are used without calibration or lens distortion correction. The voxel size of 0.01 m with a minimum of 0.3-m depth sensor setting is used. Our method successfully created the mesh of the environment with some distortions. For instance, 0.01-mm voxel size results in coarse map especially in missing wire grid fence and part of the fence door (Figure 18).

Using smaller voxel size increases the mapping performance helps to maintain grid fence as in Figure 19. If an autonomous robot is flying around thin branches, telephone lines, or chain link fencing, a detailed map is required to avoid from the collision because those are the main collision areas for outdoor autonomous drones.

In the second sequence, we captured the frame in a resolution of 1280×720 with a rate of 10 fps using default camera properties. The voxel size of 0.02 m and maximum depth of 16 m settings are used in this sequence. As can be seen from Figure 20, the final mesh has no artifact of the walking person in the scene. However, the result of EF has traces of the walking person.

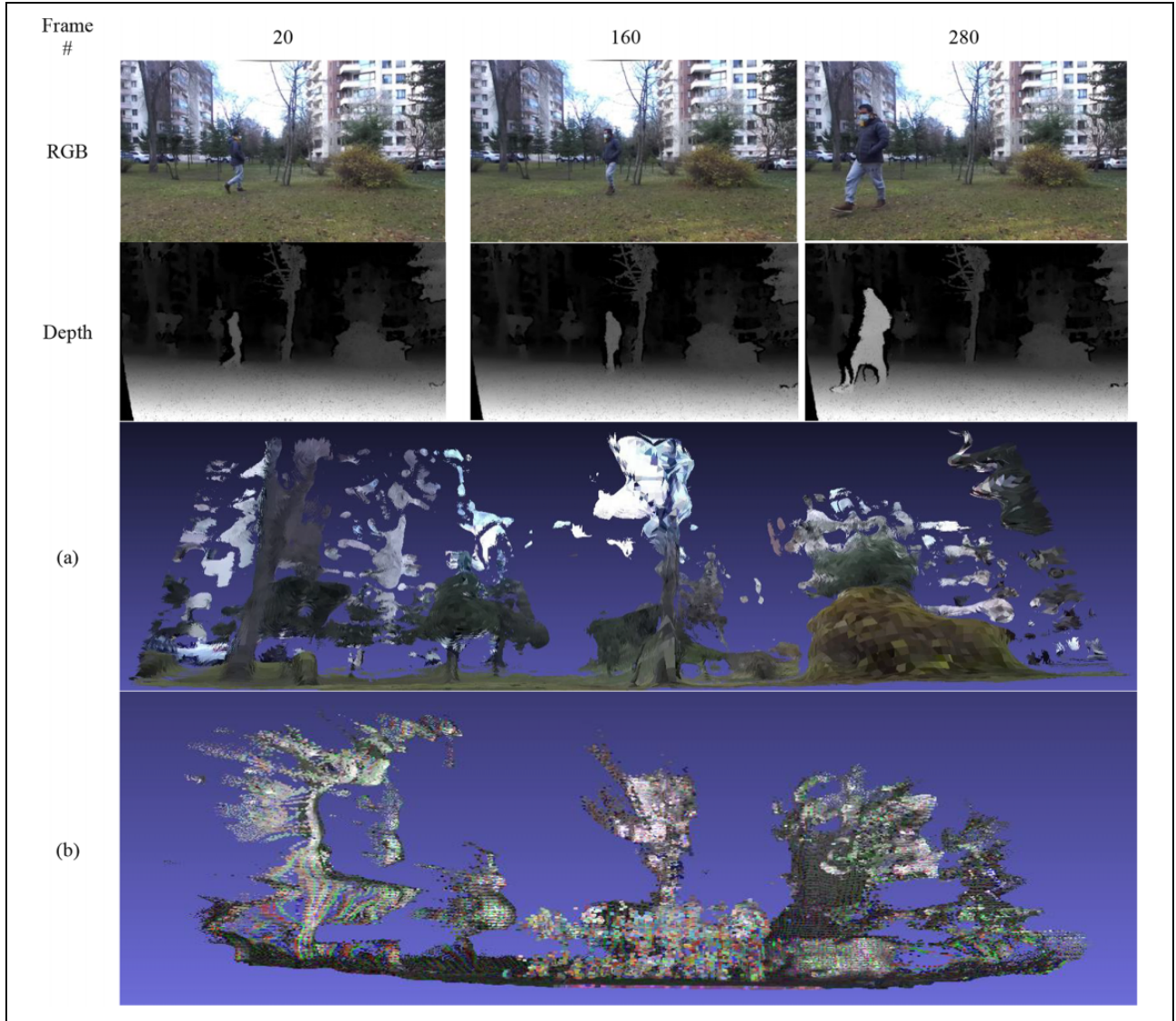


Figure 20. RGB-D image and final mesh of the “outdoor-2” sequence: (a) DUDMap and (b) elastic fusion.

Table 7. TUM “fr3/walking static” sequence translational ATE error (RMSE, cm).

Voxel size (m)	Weight (ψ)	Translational ATE error (cm)	Mean time for per frame (ms)
0.005	0.01	1.05	230
0.01	0.01	1.77	116
0.02	0.01	1.67	76
0.05	0.01	2.16	62
0.005	0.025	0.74	231
0.01	0.025	0.79	116
0.02	0.025	1.06	77
0.05	0.025	0.90	62
0.005	0.04	0.69	230
0.01	0.04	0.81	115
0.02	0.04	0.77	76
0.05	0.04	0.95	62

ATE: absolute trajectory error; RMSE: root mean square error.

Sensitivity analysis

In this section, the sensitivity of the proposed methodology to the voxel size and the contribution weight of the intensity with respect to the depth are examined (see Table 6). In addition, the required time for per image is analyzed. The fr3/walking static xyz dataset is selected for the error and timing analysis because, in this sequence, camera is tracking a person at the beginning, and finally, camera never revisits again, which results in artifact in resulting mesh. In addition, most of the state-of-the-art system use this sequence for performance analysis.

According to Table 7, in all cases, using larger voxel dramatically decreases the required calculation time, which makes that the proposed scheme is more suitable for real-time applications. However, using larger voxel increases the absolute translational error. Using larger ratio of the

intensity information with respect to the depth information decreases the RMSE error, however, such situation is not valid for all cases. Therefore, utilization of application-specific constant increases the performance of the proposed scheme.

Conclusion

Visual SLAM has been studied over the last years. The research efforts have addressed SLAM problem. However, most of the approaches assume a stationary environment.

Our proposed method, SDF-based dynamic mapping approach, can operate in environments, where high dynamics exist without depending on moving objects. In addition, a static object is moved, and the corresponding voxels are removed successfully from the mesh.

After performing a complete evaluation of our proposed method for several sequences of the TUM, Bonn, and VolumeDeform datasets, our method has an improved pose estimation capability even though there exist dynamic elements in the scene.

SDF is generally straightforward to split into independent tasks that may run in parallel, however, memory requirements are used for storing a given SDF volume scales cubically with the grid resolution. Hence, special care has to be taken for efficient memory usage considering the performance. In addition, the SDF encodes surface interfaces at subvoxel accuracy through interpolation, however, sharp corners and edges are not straightforward to extract from an SDF representation. Improvement using adaptive variable voxel size and implementing feature-preserving surface extraction on sharp corners is left for future work.

Given rising interest in developing visual odometry and SLAM algorithms for very dynamic environments, it is clear that a new RGB-D dataset containing fast and slow camera motions and varying degrees of dynamic elements would be greatly appreciated by researchers if made available.


Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by Roketsan Missiles Industries Inc.

ORCID iD

Özgür Hastürk  <https://orcid.org/0000-0002-7078-0482>

Supplemental material

Supplemental material for this article is available online.

References

1. Yu C, Liu Z, Liu X-J, et al. DS-SLAM: A Semantic Visual SLAM towards Dynamic Environments. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, 1 October 2018, pp. 1168–1174. Piscataway, NJ: IEEE, doi: 10.1109/IROS.2018.8593691.
2. Taneja A, Ballan L, and Pollefeys M. Geometric change detection in urban environments using images. *IEEE Trans Pattern Anal Mach Intell* 2015; 37(11): 2193–2206.
3. Sturm J, Engelhard N, Endres F, et al. A benchmark for the evaluation of RGB-D SLAM systems. In: *2012 IEEE/RSJ international conference on intelligent robots and systems*, Vilamoura, Portugal, 7–12 October 2012, pp. 573–580. Piscataway, NJ: IEEE, doi: 10.1109 IROS.2012.6385773.
4. Palazzolo E, Behley J, Lottes P, et al. Refusion: 3D reconstruction in dynamic environments for RGB-D cameras exploiting residuals. *CoRR*, abs/1905.02082, 2019.
5. Bagautdinov T, Fleuret F, and Fua P. Probability occupancy maps for occluded depth images. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Boston, MA, USA, 7–12 June 2015, pp. 2829–2837. Piscataway, NJ: IEEE, doi: 10.1109/CVPR.2015.7298900.
6. Dai A, Nießner M, Zollhöfer M, et al. BundleFusion: real-time globally consistent 3D reconstruction using on-the-fly surface reintegration. *ACM Trans Graph* 2017; 36(3): 1–18.
7. Mur-Artal R and Tardós JD. ORB-SLAM2: an open-source slam system for monocular, stereo, and RGB-D cameras. *IEEE Trans Robot* 2017; 33: 1255–1262.
8. Campos C, Elvira R, Rodríguez JGG, et al. ORB-SLAM3: an accurate open-source library for visual, visual-inertial and multi-map SLAM. <http://arxiv.org/abs/2007.11898>. (2020) (Accessed : 09 January 2021)
9. Pire T, Fischer T, Castro G, et al. S-PTAM: stereo parallel tracking and mapping. *Rob Auton Syst* 2017; 93: 27–42.
10. Labbe M and Michaud F. Appearance-based loop closure detection for online large-scale and long-term operation. *IEEE Trans Robot* 2013; 29: 734–745.
11. Kerl C, Sturm J, and Cremers D. Dense visual SLAM for RGB-D cameras. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, Japan, 3–7 Nov. 2013, pp. 2100–2106, Piscataway, NJ: IEEE, doi: 10.1109/IROS.2013.6696650.
12. Jaimez M, Kerl C, Gonzalez-Jimenez J, et al. Fast odometry and scene flow from RGB-D cameras based on geometric clustering. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, 29 May–3 June 2017, pp. 3992–3999. Piscataway, NJ: IEEE, doi: 10.1109/ICRA.2017.7989459.
13. Ma Y, Soatto S, Kosecka J, et al. *An invitation to 3D vision: from images to geometric models*. New York, NY: Springer Verlag, 2003.
14. Innmann M, Zollhöfer M, Nießner M, et al. VolumeDeform: real-time volumetric non-rigid reconstruction. In: *14th European conference on computer vision (ECCV)*, Amsterdam,

- the Netherlands, 11–14 October 2016, pp. 362–379. Cham: Springer.
15. Steinbrücker F, Sturm J, and Cremers D. Volumetric 3D mapping in real-time on a CPU. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, 31 May–7 June 2014, pp. 2021–2028. Piscataway, NJ: IEEE, doi: 10.1109/ICRA.2014.6907127.
 16. Whelan T, Salas-Moreno RF, Glocker B, et al. ElasticFusion: real-time dense SLAM and light source estimation. *Int J Robot Res* 2016; 35: 1697–1716.
 17. Rünz M and Agapito L. Co-fusion: Real-time segmentation, tracking and fusion of multiple objects. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, 29 May–3 June 2017, pp. 4471–4478. Piscataway, NJ: IEEE, doi: 10.1109/ICRA.2017.7989518.
 18. Scona R, Jaimez M, Petillot YR, et al. StaticFusion: Background Reconstruction for Dense RGB-D SLAM in Dynamic Environments. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD, Australia, 21–25 May 2018, pp. 3849–3856. Piscataway, NJ: IEEE, doi: 10.1109/ICRA.2018.8460681.
 19. Wang Y and Huang S. Motion segmentation based robust RGB-D SLAM. In: *WCICA*, Shenyang, China, 29 June–4 July 2014, pp. 3122–3127. IEEE.
 20. Kim H and Kim JH. Effective background model-based RGB-D dense visual odometry in a dynamic environment. *IEEE Trans Robot* 2016; 32(6): 1565–1573.
 21. Azartash H, Lee K, and Nguyen TQ. Visual odometry for RGB-D cameras for dynamic scenes. In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy, 4–9 May 2014, pp. 1280–1284. Piscataway, NJ: IEEE, doi: 10.1109/ICASSP.2014.6853803.
 22. Tan Wei, Liu Haomin, Dong Z, et al. Robust monocular SLAM in dynamic environments. In: *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, Adelaide, SA, Australia, 1–4 Oct 2013, pp. 209–218. Piscataway, NJ: IEEE, doi: 10.1109/ISMAR.2013.6671781.
 23. Kitt B, Moosmann F, and Stiller C. Moving on to dynamic environments: Visual odometry using feature classification. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, 18–22 Oct 2010, pp. 5551–5556. Piscataway, NJ: IEEE, doi: 10.1109/IROS.2010.5650517.
 24. Bescos B, Fácil JM, Civera J, et al. DynaSLAM: tracking, mapping, and inpainting in dynamic scenes. *IEEE Robot Autom Lett* 2018; 3(4): 4076–4083.
 25. He K, Gkioxari G, Dollár P, et al. Mask R-CNN. In: *2017 IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, 22–29 Oct. 2017, pp. 2980–2988. Piscataway, NJ: IEEE, doi: 10.1109/ICCV.2017.322.
 26. Runz M, Buffier M, and Agapito L. MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects. In: *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, Munich, Germany, 16–20 Oct. 2018, pp. 10–20. Piscataway, NJ: IEEE, doi: 10.1109/ISMAR.2018.00024.
 27. Wu Y, Luo L, Yin S, et al. An FPGA based energy efficient DS-SLAM accelerator for mobile robots in dynamic environment. *Appl Sci* 2021; 11(4): 1828.
 28. Liu Y and Miura J. RDS-SLAM: real-time dynamic SLAM using semantic segmentation methods. *IEEE Access* 2021; 9: 23772–23785.
 29. Fan Y, Zhang Q, Liu S, et al. Semantic SLAM with more accurate point cloud map in dynamic environments. *IEEE Access* 2020; 8: 112237–112252.
 30. Zhang T and Nakamura Y. PoseFusion: dense RGB-D SLAM in dynamic human environments. In: Xiao J, Kröger T, and Khatib O (eds) *Proceedings of the 2018 international symposium on experimental robotics*. Cham: Springer, 2018, pp. 772–780.
 31. Cao Z, Hidalgo G, Simon T, et al. OpenPose: real-time multi-person 2D pose estimation using part affinity fields. *IEEE Trans Pattern Anal Mach Intell* 2021; 43(1): 172–186.
 32. Zhang T, Zhang H, Li Y, et al. FlowFusion: Dynamic Dense RGB-D SLAM Based on Optical Flow. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, 31 May–31 Aug 2020, pp. 7322–7328. Piscataway, NJ: IEEE, doi: 10.1109/ICRA40945.2020.9197349.
 33. Sun D, Yang X, Liu M, et al. PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 18–23 June 2018, pp. 8934–8943. Piscataway, NJ: IEEE, doi: 10.1109/CVPR.2018.00931.
 34. Badrinarayanan V, Kendall A, and Cipolla R. SegNet: a deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans Pattern Anal Mach Intell* 2017; 39: 2481–2495.
 35. Lorensen WE and Cline HE. Marching cubes: a high resolution 3D surface construction algorithm. *Comput Graph* 1987; 21(4): 163–169.
 36. Diaz-Chito K, Hernandez-Sabatnd A, and López AM. A reduced feature set for driver head pose estimation. *Appl Soft Comput* 2016; 45: 98–107.
 37. Lowe DG. Distinctive image features from scale-invariant keypoints. *Gonput Vis* 2004; 60(2): 91–110.
 38. Newcombe RA, Izadi S, Hilliges O, et al. KinectFusion: Real-time dense surface mapping and tracking. In: *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, Basel, Switzerland, 26–29 Oct. 2011, pp. 127–136. Piscataway, NJ: IEEE, doi: 10.1109/ISMAR.2011.6092378.
 39. Niessner M, Izadi S., Stamminger M, et al. Real-time 3d reconstruction at scale using voxel hashing. *ACM Trans Graph* 2013; 32(6): 1–169.
 40. Whelan T, Leutenegger S, Salas-Moreno RF, et al. ElasticFusion: dense SLAM without a pose graph. In: *Robotics: science and systems (RSS)*, Rome, Italy, 13–17 July 2015.
 41. Malleson C, Guillemaut J, and Hilton A. Hybrid modeling of non-rigid scenes from RGBD cameras. *IEEE Trans Circuits Syst Video Technol* 2019; 29(8): 2391–2404.
 42. Li JW, Gao W, and Wu YH. Elaborate scene reconstruction with a consumer depth camera. *Int J Autom Comput* 2018; 15: 443–453.