REAL-TIME INDOOR SCENE RECONSTRUCTION WITH RGBD AND INERTIAL INPUT

Zunjie Zhu[1], *Feng Xu*[2], *Chenggang Yan*[1], *Xinhong Hao*[3], *Xiangyang Ji*[4], *Yongdong Zhang*[5] and *Qionghai Dai*[4]

[1]Department of Automation, Hangzhou Dianzi University
[2]BNRist and School of Software, Tsinghua University
[3]Science and Technology on Mechatronic Dynamic Control Laboratory, Beijing Institute of Technology
[4]Department of Automation, Tsinghua University
[5]School of Information Science and Technology, University of Science and Technology of China

ABSTRACT

Camera motion estimation is a key technique for 3D scene reconstruction. Previous works usually assume slow camera motions, which limit the usage in many real cases. We propose an end-to-end 3D reconstruction system which combines color, depth and inertial measurements to achieve robust reconstruction with fast sensor motions. Our framework utilizes extended Kalman filter to fuse the three kinds of information and involve an iterative method to jointly optimize feature correspondences, camera poses and scene geometry. We also propose a novel geometry-aware patch deformation technique to adapt the feature appearance in image domain, leading to a more accurate feature matching under fast camera motions. Experiments show that our patch deformation method improves the accuracy of feature tracking, and our 3D reconstruction framework outperforms the state-of-the-art solutions under fast camera motions.

Index Terms— 3D Reconstruction, Fast Motion, EKF, IMU.

1. INTRODUCTION

With the development of capture and computation devices, such as depth sensors and GPUs, real-time 3D reconstruction has got rapid development. In recent years, a lot of works have focused on indoor scene reconstruction. For example, InfiniTAM [1] uses depth information to reconstruct 3D models, and estimate camera poses by an Iterative Closest Point (ICP) algorithm [2]. However depth only methods are extremely brittle with large flat geometry, strong sun lights and depth sensor noises. And they suffer error accumulation a lot. In addition, drift-free camera tracking has got a breakthrough in RGB-based methods, including direct methods [3] and feature point-based methods [4], but these approaches can not reconstruct dense 3D geometry of a scene. BundleFusion [5], ElasticFusion [6] and some other methods [7, 8] use both color and depth information to estimate camera motions and dense geometries. Although these works exhibit reasonable results, they still require strong assumptions, like static scene without dynamic objects, sufficient texture and geometrical information, slow camera motions and invariant illumination. However, these assumptions can not be satisfied in many real applications.

Feng Xu and Chenggang Yan are corresponding authors. Email: fengxu@tsinghua.edu.cn, cgyan@hdu.edu.cn.

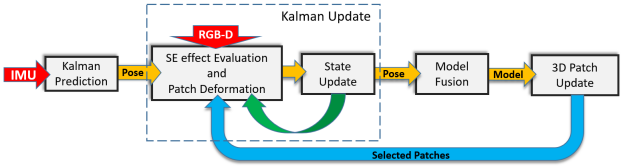

Fig. 1. Overview of our pipeline. The red, green and blue arrows represent the input acquired from current frame, iterative operation, and the patches from previous frame.

In this paper, we make a step further to handle fast camera motions. For both color and depth, fast camera motion leads to large inter-frame distance, which makes it difficult to perform image feature matching and ICP based depth alignment. We solve the issue by introducing IMU information, gathered by an accelerometer and a gyroscope. Further combining with color and depth information, robust camera pose estimation and geometry fusion of an indoor scene are jointly achieved. The main contributions of our work are as follows:

(1) We present a *RGB-D-inertial 3D reconstruction system*, which tightly combines the three kinds of information, and jointly achieves camera pose estimation and patch deformation in a Kalman filter framework.

(2) We present a *geometry-aware feature tracking method* for handling fast camera motions, which utilizes patch features to adapt blurry images and considers the deformation of patches in building feature matching for images with very different perspectives.

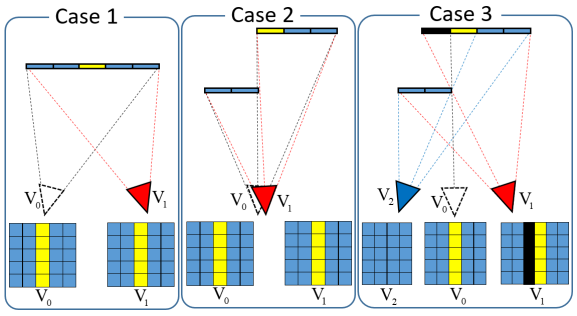

Fig. 2. This figure shows the patch SE effects caused by the camera motion and the 3D shape of a patch.

2. METHOD

The pipeline of our system is illustrated in Fig. 1. We introduce our method by the following three parts: geometry-aware feature tracking which solves the SE (Shrink and Extend) effect and executes patch deformation, filtering framework which explains the Kalman prediction and update steps, and finally the model fusion and patch update.

2.1. Geometry-aware Feature Tracking

Point-based feature tracking methods [4] extracts an insufficient number of features when images are blurred or with less texture. Thus, patch-based method, which considers larger image regions, is used to track features under these conditions [3]. However, large patches may contain objects with different depths, which change patch appearances in consecutive frames when camera motion is fast, leading to inaccurate feature tracking. To address this problem, we combine color and depth information to back project 2D patches into 3D and re-project them to the camera of the next frame by our initial camera motion. The projection helps to deform the original patches to model the appearance changes, and the patch tracking can be easily and accurately achieved by the deformed patches.

2.1.1. SE effect and Patch Deformation

When camera moves, a feature patch will be seen from different perspectives in different frames, and thus the 2D shape of the feature patch in image domain vary from different frames. To account for the patch deformation, different from Bloesch et al. [9] which only considers the 2D planar information of a patch, we use the 3D geometry of a patch to determine the 2D shape deformation of the patches between consecutive images recorded with fast camera motions.

Depending on different geometries of patches and irregularities of camera motions, patches may produce different deformations in consecutive frames. Fig. 2 shows three representative cases of patch deformations:

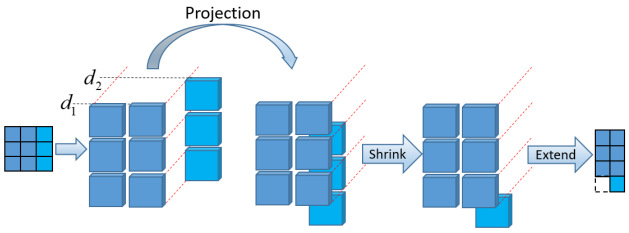

Fig. 3. Deformation process of SE patches. A 3×3 patch with depth value d_1, d_2 is projected into a new frame, then we detect the occlusion and reshape the projected patch.

Case 1. If there is no significant difference among pixel depths in a patch, no matter how aggressive the camera motion is, the general shape of the patch will remain unchanged in two consecutive frames.

Case 2. When camera moves slowly, the 2D shape of a patch will still remain unchanged even though there are large depth variances in the patch.

Case 3. Different from case 2, if camera moves aggressively, the intensity distribution and the shape of the patch will change. As shown in the right part of Fig. 2, if camera moves from $\mathbf{V}_0$ to viewpoint $\mathbf{V}_2$, the yellow region will be occluded and the patch shrinks in the frame of $\mathbf{V}_2$. In addition, the black region which is occluded in $\mathbf{V}_0$ is visible once the camera moves to $\mathbf{V}_1$, and thus the patch shape extends in the frame of $\mathbf{V}_1$. These phenomenons are called shrink and extend effect, so we call them together by SE effect.

We designed a unified deformation method to handle the SE effect. The details of patch deformation process are illustrated in Fig. 3 and formalized as follows. Each pixel i in a patch extracted from frame $k-1$ are defined as $P_i^{k-1} := (\mathbf{p}_i^{k-1}, I_i^{k-1}, d_i^{k-1}, \mathbf{n}_i^{k-1})$, where $\mathbf{p}_i$ denotes the image coordinates of pixel i in the patch. $I_i, d_i, \mathbf{n}_i$ denote the intensity, depth and 3D normal of pixel i. d_i and $\mathbf{n}_i$ are obtained from the depth image. We first back project each patch into 3D world coordinate system:

$$\mathbf{L}_i = \mathbf{T}_{k-1}\pi^{-1}(P_i^{k-1}). \tag{1}$$

Here, $\pi(\cdot)$ is to project a point from the 3D camera coordinate system to 2D pixel domain, and $\pi^{-1}(\cdot)$ represents the inverse operation. $\mathbf{T}_{k-1}$ is the camera pose of frame $k-1$ which transforms a 3D point in the camera coordinate system of frame $k-1$ to the world coordinate system which is assigned to be the camera coordinate system of the first frame. Thus $\mathbf{L}_i$ is in the world coordinate system but is indexed from a pixel i in frame $k-1$, so $\mathbf{L}_i := (\mathbf{P}_i, I_i^{k-1}, \mathbf{n}_i)$ where the intensity information does not change in the projection and $\mathbf{P}_i$ and $\mathbf{n}_i$ are in the world coordinate system. Then we project $\mathbf{L}_i$ to the pixel coordinate system of frame k.

$$P_i^{k'} = \pi(\mathbf{T}_k^{-1}\mathbf{L}_i). \tag{2}$$

Here $'$ means it is projected to this frame, not originally from this frame. Note that $\mathbf{T}_k$ is to be solved and affects the 2D position of the projection.

After the projection, if two projected pixels in P^{k-1} happened to be in the same pixel coordinate in $P^{k'}$, then we consider the shrink effect occurs in this patch. This usually happens when a region close to the camera covers the distant region. Therefore we remove the pixels corresponding to the distant region. Then we evaluate whether the extend effect happens or not. We set the shape of the projected patch to be the rectangle which covers all projected pixels. Then the extend effect is verified by checking whether the height or width of the projected patch is greater than the original one. No matter which effect happens, we use $P^{k'}$ as the deformed patch in the following feature tracking steps.

2.1.2. Objective

In feature tracking, patches affected by SE effect are replaced with the corresponding newly projected patches. Then We track the projected patch features by using both intensity and depth information.

Photometric error for each projected patch is computed as follows. We first extract the patch at the projected location of the current image as P^k, and then calculate the intensity difference between the extracted patch and the projected patch. The photometric error can be formalized as:

$$E_p = \sum_i^Y \left\| I[P_i^k] - I[P_i^{k'}] \right\|_2, \tag{3}$$

where Y denotes the number of pixels in the patch. I indicates the corresponding intensity information of a patch. Then we compute point-to-plane geometry errors as:

$$E_g = \sum_i^Y \left\| \left(\mathbf{P}[\mathbf{T}_k \cdot \pi^{-1}(P_i^k)] - \mathbf{P}[\mathbf{L}_i] \right) \cdot \mathbf{n}[\mathbf{L}_i] \right\|_2. \tag{4}$$

Given E_p and E_g, the cost function for patch tracking is formulated as:

$$E(\mathbf{T}_k) = \lambda \sum_j^M E_p^j + (1-\lambda) \sum_j^M E_g^j, \tag{5}$$

where j denotes the patch j, and M indicates the number of patches.

2.2. Filtering Framework

Our extended Kalman fitering (EKF) framework aims to tightly combine the color, depth and inertial measurements information. To be specific, we model the camera pose of each frame as the state of the EKF, and will solve it in the EKF. The observation of the EKF include both the color and depth images, and the relationship between the state and the observation is measured by the energy defined in Equation 5. If a state fits exactly to an observation, the energy is zero. On the other hand, the inertial information is used in the Kalman prediction step, which serves in building the motion prediction model.

We follow the traditional Kalman filter to define the variables. A nonlinear discrete time system with state $\mathbf{x}$, observation term z, process noise $\boldsymbol{\omega} \sim N(0, \mathbf{Q})$, and update noise $\boldsymbol{\mu} \sim N(0, \mathbf{U})$ in the kth frame can be written as

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \boldsymbol{\omega}_k), \tag{6}$$

$$\mathbf{z}_k = h(\mathbf{x}_k, \boldsymbol{\mu}_k). \tag{7}$$

In our framework, the state of the filter is composed of the following elements $\mathbf{x} : \mathbf{T} = (\mathbf{R}, \mathbf{t})$, with a 3×3 camera rotation matrix $\mathbf{R}$ and a 3×1 camera translation vector $\mathbf{t}$ in the world coordinate system. In the following, the superscript symbol '$+$' denotes a-posteriori estimate of a variable calculated from the Kalman update step and '$-$' denotes a-prior estimate from the Kalman prediction step.

2.2.1. Kalman Prediction and State Propagation

Given an a-posteriori estimate $\mathbf{x}_{k-1}^+$ with covariance $\mathbf{P}_{k-1}^+$, the prediction step of the EKF yields a-priori estimate at the next frame:

$$\mathbf{x}_k^- = f(\mathbf{x}_{k-1}^+, 0), \tag{8}$$

$$\mathbf{P}_k^- = \mathbf{F}_k \mathbf{P}_{k-1}^+ \mathbf{F}_k^T + \mathbf{Q}_k, \tag{9}$$

with the Jacobians:

$$\mathbf{F}_k = \left. \frac{\partial f}{\partial \mathbf{x}} \right|_{\mathbf{x}_{k-1}^+, 0}. \tag{10}$$

The key in the Kalman prediction step is to define the function f. In our EKF framework, the inertial measurements are employed in the definition. Following [10], we get the actual sensor acceleration $\mathbf{a}$ and angular velocity $\mathbf{w}$ from the inertial measurements. We assume that the IMU is synchronized with the camera and acquires measurements with time interval τ which is much smaller than that of the camera. Hence, we denote N as the number of inertial measurements acquired in two consecutive camera frames, and then merge them together by a pre-integration method [11] to predict camera rotation $\Delta\mathbf{R}$ and translation $\Delta\mathbf{t}$ between two consecutive frames:

$$\Delta\mathbf{R} = \Phi \cdot \prod_{n=1}^{N} \mathrm{Exp}(\mathbf{w}_n \cdot \tau) \cdot \Phi^{-1} \tag{11}$$

$$\Delta\mathbf{v} = \sum_{n=1}^{N} \Delta\mathbf{R}_n \cdot \mathbf{a}_n \cdot \tau \tag{12}$$

$$\Delta\mathbf{t} = \Phi \cdot \sum_{n=1}^{N} (\Delta\mathbf{v}_n \cdot \tau + \frac{1}{2}\mathbf{g} \cdot \tau^2 + \frac{1}{2}\Delta\mathbf{R}_n \cdot \mathbf{a}_n \cdot \tau^2). \tag{13}$$

In the above three equations, the subscript 'n' denotes the corresponding variable at the nth IMU input in consecutive frames. Besides, $\Delta\mathbf{v}$ is the accumulated IMU linear velocity from the previous camera frame to the current camera frame, and Φ is the extrinsic matrix from the IMU coordinate to the camera coordinate. $\mathbf{g}$ is the gravity acceleration, and $\mathrm{Exp}(\cdot)$ denotes the exponential map from Lie-algebra to Lie-group. Details about this prediction step can be found in [10]. Finally, the states predicted in current frame k can be formulated as:

$$\begin{bmatrix} \mathbf{R}_k^- & \mathbf{t}_k^- \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \Delta\mathbf{R}_k \cdot \mathbf{R}_{k-1}^+ & \Delta\mathbf{t}_k + \mathbf{t}_{k-1}^+ \\ \mathbf{0} & 1 \end{bmatrix}. \tag{14}$$

2.2.2. *Kalman Update and Iteration*

In traditional extended Kalman update step, the measurement residual is modeled as:

$$\mathbf{y}_k = \mathbf{z}_k - h(\mathbf{x}_k^-, \mathbf{0}). \tag{15}$$

Here, $\mathbf{0}$ means we directly use $\mathbf{x}_k^-$ to calculate the residual without adding any Gausian noise. The updated state is formulated as:

$$\mathbf{x}_k^+ = \mathbf{x}_k^- + \mathbf{K}_k \cdot \mathbf{y}_k, \tag{16}$$

where $\mathbf{K}_k$ is the Kalman gain. In our method, we defined the residual as the photometric and geometric error of patches (equation 5), thus the residual can be formulated as:

$$\mathbf{y}_k = 0 - E(\mathbf{T}_k^-) = -E\left(\begin{bmatrix}\mathbf{R}_k^- & \mathbf{t}_k^- \\ \mathbf{0} & 1\end{bmatrix}\right). \tag{17}$$

Notice that the deformations of the patches, which are used in calculating $\mathbf{y}_k$ by Equation 17, is heavily affected by the camera poses. After we obtained an updated camera pose by Equation 16, we use the newly updated camera pose to iteratively calculate the deformations of the patches and refine the camera poses by Equation 16 again. In this manner, we can estimate a more accurate $\mathbf{x}_k^+$. To be more specific, we use m to denote the iterations, and thus we have:

$$h(\mathbf{x}_{k,m}^+, 0) = E\left(\begin{bmatrix}\mathbf{R}_{k,m}^+ & \mathbf{t}_{k,m}^+ \\ \mathbf{0} & 1\end{bmatrix}\right), \tag{18}$$

and the Kalman gain respecting to each iteration is:

$$\mathbf{K}_{k,m} = \mathbf{P}_k^- \mathbf{H}_{k,m}^T \mathbf{S}_{k,m}^{-1} \tag{19}$$

$$\mathbf{S}_{k,m} = \mathbf{H}_{k,m} \mathbf{P}_k^- \mathbf{H}_{k,m}^T + \mathbf{U}_k. \tag{20}$$

As defined in the beginning of Section 2.2, $\mathbf{U}_k$ is the covariance matrix of noise $\boldsymbol{\mu}_k$. And the Jacobians updated in every iteration are formulated as:

$$\mathbf{H}_{k,m} = \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\mathbf{x}_{k,m}^+, 0} \tag{21}$$

Then the updated state of each iteration is calculated as follows:

$$\mathbf{x}_{k,m+1}^+ = \mathbf{x}_{k,m}^+ - \mathbf{K}_{k,m} \cdot h(\mathbf{x}_{k,m}^+, \mathbf{0}). \tag{22}$$

Notice that $\mathbf{x}_{k,0}^+$ is set to be $\mathbf{x}_k^-$ to initialize our computation. Finally, the iteration is terminated when the absolute value of $\mathbf{K}_{k,m} \cdot h(\mathbf{T}_{k,m}^+, \mathbf{0})$ is below a certain threshold and the covariance matrix is only updated once the process has converged after η iterations:

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_{k,\eta}\mathbf{H}_{k,\eta})\mathbf{P}_k^- \tag{23}$$

2.3. Model Fusion and Patch Update

We use the volumetric TSDF [12] to incrementally fuse each depth frame $\mathbf{D}_k$ into one 3D geometry model $\mathbf{M}_k(\mathbf{X})$, with the associated camera pose $\mathbf{R}_k^+, \mathbf{t}_k^+$. Details about depth fusion can be found in [13].

After the reconstruction, we should update patch features for the subsequent tracking. We exclude bad features by comparing average intensity error of patches, and re-extract square patch feature for those with non-square shapes affected by the SE effect. Then, we add new features with distinct intensity gradient and sufficient depth information, evaluated by the FAST corner detector [14] and the number of pixels with available depth information. Finally, the intensity information of patch features is updated by the current color image, and the depth information is acquired from the 3D geometry model which has better quality than the current depth image.

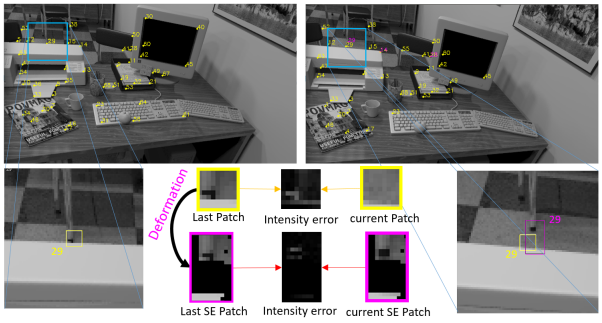

Fig. 4. This figure shows the feature tracking results of ours and the direct method.

Table 1. Comparison of Patch Feature Tracking

Type	Dataset	AIE	
		DM	Ours
slow	TUM_freiburg1_desk	13.3756	**9.53**
	ICL_NUIM_lr_kt2	4.8981	4.0825
	Dorm_slow	8.1312	7.8934
fast	ICL Fast Motion	17.219	**7.8328**
	Dorm_fast	13.9011	**7.9325**

3. EXPERIMENTS

We first demonstrate the effectiveness of our geometry-aware feature tracking method, which considers SE effect and deforms patches for accurate feature tracking in sequences with fast camera motion. Then, we evaluate the benefits of inertial information by comparing our system with and without IMU. Finally, our 3D reconstruction method is compared against the state-of-the-art systems in datasets with fast sensor motion.

3.1. Evaluation

Feature tracking. We compare our feature tracking method against traditional direct method(DM) which does not take the SE effect into consideration. We use a patch-size 10×10 in both methods and extract no more than 100 patches in each frame. Fig. 4 shows the tracking results of a patch feature

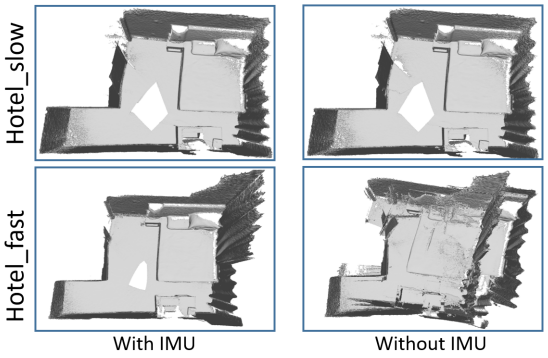

Fig. 5. The reconstruction results of a hotel room under slow and fast camera motion.

in two consecutive frames. The tracking result of traditional method is severely influenced by SE effect and get great intensity error, while our method eliminating the influence by deforming the patch and get smaller intensity error.

We compared on several datasets, including ICL datasets[15, 16], TUM datasets[17] and our datasets gathered by a hand-held sensor. The average intensity error (AIE) of patches are listed in Table 1. All datasets are divided into slow and fast depending on the quality of recorded images. To be more specific, as there is no explicit criteria to divide camera speed into slow and fast, thus we empirically set, based on the unified characteristics of most public datasets, the motion without creating image motion blur as slow camera motion, and the motion which creates severe image blur as fast motion. From the table 1, we find that our method gets smaller AIE in all datasets, especially in datasets with fast camera motion.

IMU. To verify whether the integration of IMU helps to reconstruct the scene geometry during fast camera motion, we compare the results with and without IMU on two datasets with slow and fast camera motions, respectively. As shown in Fig. 5, on the dataset with slow camera motion, the system without IMU works on-par with the complete system, while it fails to reconstruct the model for fast camera motions. Fig. 6 demonstrates the details of camera motions in the two datasets. From the figure, we found there exists some subsequences with large linear and angular velocities of camera in the fast dataset. And the system without IMU will fail to track camera poses during these sub-sequences. Notice that other fast datasets used in our experiments also contain this kinds of sub-sequences.

3.2. Comparison

We compare our 3D reconstruction systems with three public state-of-the-art methods: InfiniTAM[1], a typical voxel based scene reconstruction method, Bundlefusion[5] which proposes an efficient global pose optimization algorithm, and a surfel-based method ElasticFusion[6] which contains loop

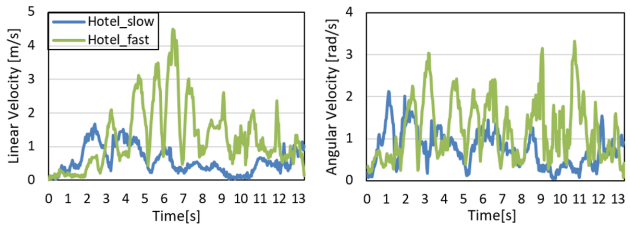

Fig. 6. Camera linear and angular velocity of the two sequences used in IMU evaluation.

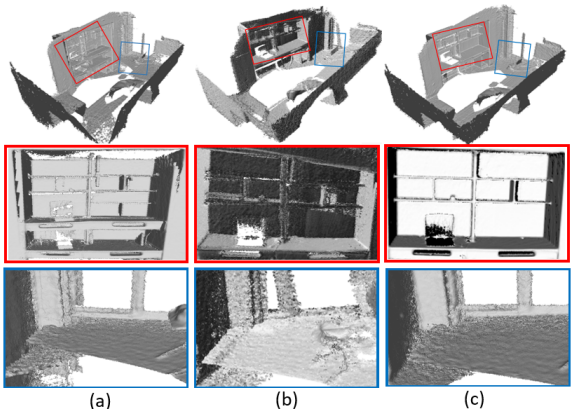

Fig. 7. Comparison of reconstruction with fast camera motion in (a)InfiniTAM (b)ElasticFusion and (c)Ours.

closure and executes model refinement through non-rigid surface deformations.

The results of sequence Dorm_fast are exhibited in Fig. 7. As BundleFusion fails once the camera speeds up and subsequently restarts when the camera slows down, thus we only show its reconstruction process in our supplementary video. From Fig. 7, we find that InfiniTAM can not maintain consistency in the reconstructed geometry, which is mainly caused by the inaccurate camera pose estimation and large accumulated errors. Meanwhile, the loop closure function of ElasticFusion, aiming to eliminate accumulated errors, is always invalid in fast camera motion, and thus leads to the fail reconstruction of the parts shown with red and blue bounding boxes. In the opposite, our system reconstructs a good geometry of the scene even without loop closure.

We test our system on a laptop with Intel Core i7-7820HK CPU at 2.9GHz, 32GB of RAM and GPU GeForce GTX 1080 with 8GB memory. The computation of our method is accelerated by GPU and runs on 45FPS.

4. CONCLUSION AND FUTURE WORK

We present a real-time system for indoor scene reconstruction by tightly-coupling RGB-D-Inertial information with an extended Kalman filter, which estimates camera poses and reconstructs 3D scene model with fast camera motions. In ad-

dition, we explore the SE effect caused by fast camera motions and handle it by a geometry-aware patch deformation method. However, our system has not achieved loop closure with fast camera motions. The reason is that the degraded image information caused by fast camera motion, results in the difficulties in loop closure detection.

5. ACKNOWLEDGMENT

This work is supported by the National Key Research and Development Program of China under Grant (2017YFC0820604), the NSFC (No. 61671196, 61525206, 61701149, 61822111, 61727808, 61671268), Zhejiang Province Nature Science Foundation of China LR17F030006 and Beijing Natural Science Foundation (L182052).

6. REFERENCES

[1] Olaf Kähler, Victor Adrian Prisacariu, Carl Yuheng Ren, Xin Sun, Philip Torr, and David Murray, "Very high frame rate volumetric integration of depth images on mobile devices," *IEEE transactions on visualization and computer graphics*, vol. 21, no. 11, pp. 1241–1250, 2015.

[2] Szymon Rusinkiewicz and Marc Levoy, "Efficient variants of the icp algorithm," in *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*. IEEE, 2001, pp. 145–152.

[3] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, and Davide Scaramuzza, "Svo: Semidirect visual odometry for monocular and multicamera systems," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, 2017.

[4] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[5] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt, "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 76a, 2017.

[6] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger, "Elasticfusion: Real-time dense slam and light source estimation," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1697–1716, 2016.

[7] Chenggang Yan, Yongdong Zhang, Jizheng Xu, Feng Dai, Liang Li, Qionghai Dai, and Feng Wu, "A highly parallel framework for hevc coding unit partitioning tree decision on many-core processors," *IEEE Signal Processing Letters*, vol. 21, no. 5, pp. 573–576, 2014.

[8] Tristan Laidlow, Michael Bloesch, Wenbin Li, and Stefan Leutenegger, "Dense rgb-d-inertial slam with map deformations," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 6741–6748.

[9] Michael Bloesch, Michael Burri, Sammy Omari, Marco Hutter, and Roland Siegwart, "Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback," *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1053–1072, 2017.

[10] Anastasios I Mourikis and Stergios I Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Robotics and automation, 2007 IEEE international conference on*. IEEE, 2007, pp. 3565–3572.

[11] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2017.

[12] Brian Curless and Marc Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996, pp. 303–312.

[13] Hao Zhang and Feng Xu, "Mixedfusion: Real-time reconstruction of an indoor scene with dynamic objects," *IEEE Transactions on Visualization and Computer Graphics*, 2017.

[14] Edward Rosten and Tom Drummond, "Machine learning for high-speed corner detection," in *European conference on computer vision*. Springer, 2006, pp. 430–443.

[15] Ankur Handa, Richard A Newcombe, Adrien Angeli, and Andrew J Davison, "Real-time camera tracking: When is high frame-rate best?," in *European Conference on Computer Vision*. Springer, 2012, pp. 222–235.

[16] A. Handa, T. Whelan, J.B. McDonald, and A.J. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Hong Kong, China, May 2014.

[17] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.