2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Madrid, Spain, October 1-5, 2018

# Probabilistic Dense Reconstruction from a Moving Camera

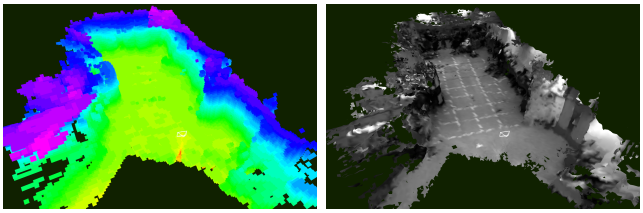
Yonggen Ling[1], Kaixuan Wang[2], and Shaojie Shen[2]

***Abstract*— This paper presents a probabilistic approach for online dense reconstruction using a single monocular camera moving through the environment. Compared to spatial stereo, depth estimation from motion stereo is challenging due to insufficient parallaxes, visual scale changes, pose errors, etc. We utilize both the spatial and temporal correlations of consecutive depth estimates to increase the robustness and accuracy of monocular depth estimation. An online, recursive, probabilistic scheme to compute depth estimates, with corresponding covariances and inlier probability expectations, is proposed in this work. We integrate the obtained depth hypotheses into dense 3D models in an uncertainty-aware way. We show the effectiveness and efficiency of our proposed approach by comparing it with state-of-the-art methods in the TUM RGB-D SLAM & ICL-NUIM dataset. Online indoor and outdoor experiments are also presented for performance demonstration.**

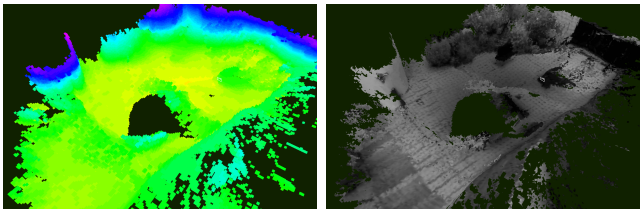
## I. Introduction

Accurate localization and dense mapping are fundamental components of autonomous robotic systems as they serve as the perception input for obstacle avoidance and path planning. While localization from a monocular camera has been well discussed in the past [1]–[5], online dense reconstruction using a single moving camera is still under development [6]–[9]. Since monocular depth estimation is based on consecutive estimated poses and images, main issues of it are: imprecise poses due to localization errors, inaccurate visual correspondences due to insufficient parallaxes and visual scale changes, etc. Depth estimation from traditional spatial stereo cameras (usually in the front-parallel setting), however, avoids the issues met with motion stereo. Thus many algorithms based on stereo cameras have been developed in the past decades [10, 11]. The significant drawback of spatial stereo is its baseline limitation: distant objects can be better estimated using longer baselines because of larger disparities; while close-up structures can be better reconstructed using shorter baselines because of larger visual overlaps. Moreover, for real world applications such as mobile robots, phones and wearable devices, it is impossible to equip them with long baseline stereo cameras because of the size constraint. If the baseline length, compared to the average scene depth of the perceived environment, is relatively small, images captured on stereo cameras will be similar. As a result, visual information from stereo cameras degrades to the same level as that obtained by a monocular camera.

[1]Tencent AI Lab, China. [2]The Hong Kong University of Science and Technology, Hong Kong, SAR China. Correspondence to: Yonggen Ling ylingaa@connect.ust.hk, Kaixuan Wang and Shaojie Shen {kwangap, eeshaojie}@ust.hk. This work was partially supported by HKUST institutional studentship.



(a) Dense indoor reconstruction for motion planning. (b) Meshing view of indoor reconstruction for visualization.



(c) Dense outdoor reconstruction for motion planning. (d) Meshing view of indoor reconstruction for visualization.

Fig. 1. Dense reconstruction of an indoor/outdoor environment from a single moving camera. (a)(c) Reconstruction for robotic applications, such as motion planning and obstacle avoidance. Colors vary w.r.t. the height to show the structure of the reconstructed dense environment. (b)(d) Meshing view by applying marching cubes [12] on TSDFs for visualization. More details can be found at: https://1drv.ms/v/s!ApzRxvwAxXqQmlW9ZOrp9hdA7ude.

Fundamentally different from passive cameras, time-of-flight (TOF) cameras as well as structure-light cameras, emit light actively. They are able to provide high accuracy depth measurements. With the advent of Microsoft Kinect and ASUS Xtion, dense reconstruction algorithms based on active depth cameras [13]–[15] have achieved impressive results in recent years. Unfortunately, active sensors do not work under strong sunlight, which limits their application to indoor environments.

This paper focuses on dense reconstructions using a single monocular camera, which adapts to both indoor and outdoor environments with various scene depth ranges. Comparing to existing methods [7, 9, 13]–[16], we make careful improvements to multiple sub-modules of the whole mapping pipeline, resulting in substantial gains in the mapping performance. The main contributions of this paper are as follows:

- A joint probabilistic consideration of depth estimation and integration.
- A detailed discussion of aggregated costs and their probability modeling.
- An online, recursive, probabilistic depth estimation scheme that utilizes both the spatial and temporal correlations of consecutive depth estimates.
- Open-source implementations available at

https://github.com/ygling2008/probabilistic_mapping.

To validate the effectiveness and efficiency of the proposed approach, we compare it with state-of-the-art methods on the TUM RGB-D SLAM & ICL-NUIM dataset. We also demonstrate its online performance on indoor and outdoor dense reconstructions.

The rest of this paper is structured as follows. Sect. II reviews the related work. Our proposed approach is presented in Sect. III, with experimental comparisons and validations demonstrated in Sect. IV. Sect V draws the conclusion and points out possible future extensions.

## II. Related Work

There has been extensive scholarly work on reconstructing a scene from images collected by a single moving camera. We only discuss the works most related to ours, that is, online monocular dense reconstruction systems.

Early live dense reconstruction systems are proposed by Stuhmer1 et al. [17] and Newcombe et al. [6], where the problem of dense reconstruction is formulated as an optimization problem. They solve for all depth values in multiple views by jointly minimizing the intensity difference and depth discontinuity. While Stuhmer1 et al. [17] rely on feature tracking for localization, Newcombe et al. [6] use the built dense reconstruction for pose tracking. Optimization-based methods are computationally intensive, thus they are usually run on high-performance GPUs.

To resolve the demanding computations, [9] ignores the spatial correlation between neighboring depth estimates, and computes each depth independently. [8], [18], and [7] decouple the constraints of photometric consistency and depth continuity. They firstly search for the optimal depth estimate for every pixel and then regularize the computed depths to enforce the consistency between neighboring depth estimates. Various filters are also included for outlier detection and removal. While these relaxations greatly reduce the algorithmic complexity, mapping results of these approaches are not as good as those of the optimization-based methods. Another relaxation is to narrow the depth searching range by merely evaluating depth values within a limited number of discrete depth samples [16]. [16] uses the dynamic programming scheme proposed in semi-global matching (SGM) [10] for cost minimization. It runs fast; however, its depth estimation contains many outliers as it neither makes use of the temporal correlation in image sequences nor deals with outliers.

The last algorithms to mention are those that reconstruct dense 3D models from sparse features or semi-dense mapping results [19]–[21]. [19] computes depth in multiple levels of images and then combines the obtained results into a final one. The density of mapping outputs from [19] depends on environments that are suitable for multi-level matching. Based on the local planar assumption, [20] and [21] use superpixels to expand the built semi-dense maps to dense mappings. [20] and [21] run fast; however, their superpixel extraction algorithms are not robust, with many ambiguities for superpixel segmentation. The effectiveness of the local planar assumption depends on the quality of superpixel extraction.

The most similar work to ours is [7]. While [7] adopts total variation smoothing for incorporating depth continuity, our work uses the dynamic programming scheme [10] instead. Moreover, we utilize both the spatial and temporal correlations inherent to image sequences in the whole probabilistic and recursive depth estimation process. [7] decouples them into two separate steps. We consider more cases of cost aggregation and probability modeling than [7], and we also introduce an uncertainty-aware depth integration for dense reconstruction, which is not covered in [7].

## III. Monocular Dense Reconstruction

Our reconstruction pipeline is shown in Fig. 3. It consists of three steps: depth estimation, hypothesis filtering, and uncertainty-aware depth integration.

### A. Depth Estimation via Motion Stereo

Our dense reconstruction system is built upon a feature-based SLAM pipeline, which provides camera poses in real time. This SLAM pipeline can be vision-based [4] or visual-inertial-based [2, 5, 22]. For each incoming keyframe image, we compute its corresponding depth estimation.

*1) Temporal Cost Aggregation:* We set the latest incoming keyframe as the reference frame, and aggregate information from past frames. $K_a$ ($K_a = 5$) frames spanning various parallax ranges are selected. They uniformly cover the average parallax deviation, ranging from 0 to $K_p$ ($K_p$ = 100) pixels, from the reference frame. This deviation is computed as the average corner location difference of the tracked features with rotation compensation. We select past frames based on the parallax deviation instead of the actual distance for adaption to environments with various scene depths.

For the benefit of online computation, we restrict every depth estimate to be one of $L$ ($L$ = 64) depth samples. These $L$ depth samples are not uniformly distributed within the feasible depth range. Instead they follow the principle of *depth from disparity*: each depth $d$ is a function of its disparity $disp$, baseline length $b$ and focus length $f$,

$$d = \frac{bf}{disp} = \frac{1}{disp \cdot \frac{1}{bf}} = \frac{1}{disp \cdot c_d} \tag{1}$$

where $c_d = \frac{1}{bf}$. Baseline length $b$ is set depending on the average depth of the perceived environment. We enumerate $disp$ from 0 to $L-1$, and obtain the set of $L$ depth samples $\Phi(L) = \{\frac{1}{63 \cdot c_d}, \frac{1}{62 \cdot c_d}, ..., \infty\}$. Given a pixel $\mathbf{u}_i$ in the reference image $i$ as well as its depth $d_{\mathbf{u}} \in \Phi(L)$, we project it on its aggregation frame $j \in K_a$ with pixel coordinate $\mathbf{u}_j$:

$$\begin{bmatrix}\mathbf{u}_j \\ 1\end{bmatrix} \simeq \mathbf{K}\mathbf{R}_w^j(\mathbf{R}_i^w d_{\mathbf{u}} \mathbf{K}^{-1} \begin{bmatrix}\mathbf{u}_i \\ 1\end{bmatrix} + \mathbf{t}_i^w - \mathbf{t}_j^w) = d_{\mathbf{u}}\mathbf{h}_i^j + \mathbf{c}_i^j \tag{2}$$

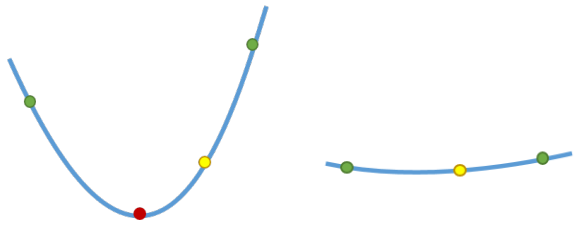
where $\mathbf{h}_i^j = \mathbf{K}\mathbf{R}_w^j\mathbf{R}_i^w\mathbf{K}^{-1}\begin{bmatrix}\mathbf{u}_i \\ 1\end{bmatrix}$, $\mathbf{c}_i^j = \mathbf{K}\mathbf{R}_w^j(\mathbf{t}_i^w - \mathbf{t}_j^w)$, $\mathbf{K}$ is the camera matrix, and $\mathbf{R}_i^w$, $\mathbf{R}_j^w$ and $\mathbf{t}_i^w$, $\mathbf{t}_j^w$ are rotations

and translations of images $i$ and $j$ w.r.t. the world frame respectively. The cost $e(\mathbf{u}_i, d_\mathbf{u}, \mathbf{u}_j)$ between $\mathbf{u}_i$ and $\mathbf{u}_j$ given $d_\mathbf{u}$ is the sum of the absolute differences between intensities within two $3\times3$ patches centered on $\mathbf{u}_i$ and $\mathbf{u}_j$. We define the cost of pixel $\mathbf{u}_i$ with depth estimate $d_\mathbf{u}$ as $e(\mathbf{u}_i, d_\mathbf{u})$, which is the aggregation of costs $e(\mathbf{u}_i, d_\mathbf{u}, \mathbf{u}_j)$ from $K_a$ selected frames:

$$e(\mathbf{u}_i, d_\mathbf{u}) = \frac{1}{N_a} \sum_{j \in N_a} e(\mathbf{u}_i, d_\mathbf{u}, \mathbf{u}_j) \tag{3}$$

where $N_a$ ($<= K_a$) is the number of $\mathbf{u}_j$ within the image size after projection.

*2) Spatial Regulation:* We notice that using a simple winner-takes-all strategy after the cost aggregation step does not produce reliable depth estimate, as it does not capture the piece-wise linear nature of depth images. In addition, in regions that are texture-less or with repetitive pattern, aggregated cost at a branch of depths are similar. As a result, the depth estimate from the winner-takes-all strategy is greatly affected by the image noise. We thus incorporate the spatial constraints between neighboring depths by using the semi-global optimization proposed in [10]. The 4-path dynamic programming is adopted for the balance between complexity and accuracy.



(a) (b)

Fig. 2. The local region around the optimal depth estimate (shown in yellow): (a) NOT flat, (b) flat. The previous and next depth sample of the optimal depth estimate are shown in green. Refined depths are shown in red.

*3) Local Region Discussion & Depth Refinement:* We define $S(\mathbf{u}_i, d_\mathbf{u})$ as the cost of pixel $\mathbf{u}_i$ with depth $d_\mathbf{u}$ after the 4-path aggregation [10] in the previous step. We can take $d^*_\mathbf{u} = \min_{d_\mathbf{u}} S(\mathbf{u}_i, d_\mathbf{u})$ as the output depth estimate. However, since $d_\mathbf{u}$ is one of discrete samples from $\Phi(L)$, its accuracy may not be high. We are going to refine the output depth estimate. We examine the local region around the optimal depth value $d^*_\mathbf{u} = \min_{d_\mathbf{u}} S(\mathbf{u}_i, d_\mathbf{u})$ (i.e. the yellow point in Fig. 2). Let $d^{*-}_\mathbf{u}$ and $d^{*+}_\mathbf{u}$ be the previous and next depth sample of $d^*_\mathbf{u}$ respectively (i.e. the green point in Fig. 2). There are two cases, as shown in Fig. 2. In case (a), the local region around the optimal depth value $d^*_\mathbf{u}$ is NOT flat, and we use parabola interpolation to improve the depth estimate accuracy:

$$S(\mathbf{u}_i, d^{*-}_\mathbf{u}) = c_0 {d^{*-}_\mathbf{u}}^2 + c_1 d^{*-}_\mathbf{u} + c_2 \tag{4}$$

$$S(\mathbf{u}_i, d^{*}_\mathbf{u}) = c_0 {d^{*}_\mathbf{u}}^2 + c_1 d^{*}_\mathbf{u} + c_2 \tag{5}$$

$$S(\mathbf{u}_i, d^{*+}_\mathbf{u}) = c_0 {d^{*+}_\mathbf{u}}^2 + c_1 d^{*+}_\mathbf{u} + c_2 \tag{6}$$

where $c_0$, $c_1$, and $c_2$ are three parabola parameters. Solving the above equations, we get the refined depth estimate (i.e. the red point in Fig. 2 (a)):

$$d^*_\mathbf{u} \leftarrow d^*_\mathbf{u} - \frac{1}{2} \frac{S(\mathbf{u}_i, d^{*+}_\mathbf{u}) - S(\mathbf{u}_i, d^{*-}_\mathbf{u})}{S(\mathbf{u}_i, d^{*+}_\mathbf{u}) + S(\mathbf{u}_i, d^{*-}_\mathbf{u}) - 2S(\mathbf{u}_i, d^{*}_\mathbf{u})}. \tag{7}$$

In case (b), where the local region around the optimal depth value $d^*_\mathbf{u}$ is flat, i.e., $2\times(1+\epsilon_d)\times S(\mathbf{u}_i, d^*_\mathbf{u}) > S(\mathbf{u}_i, d^{*-}_\mathbf{u}) + S(\mathbf{u}_i, d^{*+}_\mathbf{u})$ and $\epsilon_d = 0.05$ ($\epsilon_d$ can also be learned using opened RGB-D datasets), depth estimation is not reliable. We regard this depth estimate as an outlier depth estimate.

Note that different cases of local regions result in different probability modelings and update schemes (Sect. III-B).

### B. Hypothesis Filtering via Bayesian Gaussian Beta Process

*1) Preliminaries:* We observe that there are a few outlier depth estimates obtained in the previous step due to occlusion, lack of texture, violation of photometric consistency, etc. Different from [16] where outliers are not taken into account, we explicitly deal with outlier depth estimates. We assume that outlier depth estimates are uniformly distributed among the depth sample set $\Phi(L)$. We thus model the distribution of a depth estimate $d^t_\mathbf{u}$ ($d^*_\mathbf{u}$ at time instant $t$) as a Gaussian + uniform mixture model distribution [7, 9]: a good depth estimate is normally distributed around a correct depth $z_\mathbf{u}$ with probability $\pi_\mathbf{u}$, while an outlier estimate is uniformly distributed within an interval $[z_l, z_r]$ with probability $1-\pi_\mathbf{u}$. The depth estimate probability density function of cases (a) and (b) in Sect. III-A.3 is defined as

$$p(d^t_\mathbf{u}|\pi_\mathbf{u}, z_\mathbf{u}) = \begin{cases} \pi_\mathbf{u}\mathcal{N}(d^t_\mathbf{u}|z_\mathbf{u}, r^2_\mathbf{u}) + (1-\pi_\mathbf{u})\mathcal{U}(d^t_\mathbf{u}|z_l, z_r), & \text{(a)} \\ (1-\pi_\mathbf{u})\mathcal{U}(d^t_\mathbf{u}|z_l, z_r), & \text{(b)} \end{cases}$$

where $\mathcal{N}(d^t_\mathbf{u}|z_\mathbf{u}, r^2_\mathbf{u})$ is a Gaussian distribution with mean $z_\mathbf{u}$ and covariance $r^2_\mathbf{u}$, and $\mathcal{U}(d^t_\mathbf{u}|z_l, z_r)$ is a uniform distribution with $z_l$ and $z_r$ corresponding to the depth range of interest. The posterior of $z_\mathbf{u}$, $\pi_\mathbf{u}$ given $d^t_\mathbf{u}$ ($t \in [0\ 1\ ...\ n]$) is

$$\begin{aligned} p(\pi_\mathbf{u}, z_\mathbf{u}|d^n_\mathbf{u}, ..., d^0_\mathbf{u}) &\propto p(d^n_\mathbf{u}, ..., d^0_\mathbf{u}|\pi_\mathbf{u}, z_\mathbf{u})p(\pi_\mathbf{u}, z_\mathbf{u}) \\ &\propto p(d^n_\mathbf{u}, ..., d^0_\mathbf{u}|\pi_\mathbf{u}, z_\mathbf{u}) \\ &= p(d^n_\mathbf{u}|\pi_\mathbf{u}, z_\mathbf{u})p(d^{n-1}_\mathbf{u}, ..., d^0_\mathbf{u}|\pi_\mathbf{u}, z_\mathbf{u}) \\ &\propto p(d^n_\mathbf{u}|\pi_\mathbf{u}, z_\mathbf{u})p(\pi_\mathbf{u}, z_\mathbf{u}|d^{n-1}_\mathbf{u}, ..., d^0_\mathbf{u}) \end{aligned} \tag{8}$$

Similar to [9] and [7], we approximate $p(\pi_\mathbf{u}, z_\mathbf{u}|d^n_\mathbf{u}, ..., d^1_\mathbf{u}, d^0_\mathbf{u})$ using the product of a Gaussian distribution and a beta distribution for the sake of inference:

$$q(\pi_\mathbf{u}, z_\mathbf{u}|a_\mathbf{u}, b_\mathbf{u}, \mu_\mathbf{u}, \sigma_\mathbf{u}) = \mathcal{N}(z_\mathbf{u}|\mu_\mathbf{u}, \sigma^2_\mathbf{u})\mathcal{B}(\pi_\mathbf{u}|a_\mathbf{u}, b_\mathbf{u}) \tag{9}$$

where $\mu_\mathbf{u}$ and $\sigma^2_\mathbf{u}$ are the mean and variance of the depth estimate, while $a_\mathbf{u}$ and $b_\mathbf{u}$ are probabilistic counters of how many inlier and outlier measurements have occurred during the lifetime of the depth estimate. This leads to:

$$\begin{aligned} q(\pi_\mathbf{u}, z_\mathbf{u}|a^n_\mathbf{u}, b^n_\mathbf{u}, \mu^n_\mathbf{u}, \sigma^n_\mathbf{u}) &\approx p(\pi_\mathbf{u}, z_\mathbf{u}|d^n_\mathbf{u}, ..., d^1_\mathbf{u}, d^0_\mathbf{u}) \\ &\approx p(d^n_\mathbf{u}|\pi_\mathbf{u}, z_\mathbf{u})p(\pi_\mathbf{u}, z_\mathbf{u}|a^{n-1}_\mathbf{u}, b^{n-1}_\mathbf{u}, \mu^{n-1}_\mathbf{u}, \sigma^{n-1}_\mathbf{u}). \\ &\approx p(d^n_\mathbf{u}|\pi_\mathbf{u}, z_\mathbf{u})q(\pi_\mathbf{u}, z_\mathbf{u}|a^{n-1}_\mathbf{u}, b^{n-1}_\mathbf{u}, \mu^{n-1}_\mathbf{u}, \sigma^{n-1}_\mathbf{u}). \end{aligned} \tag{10}$$

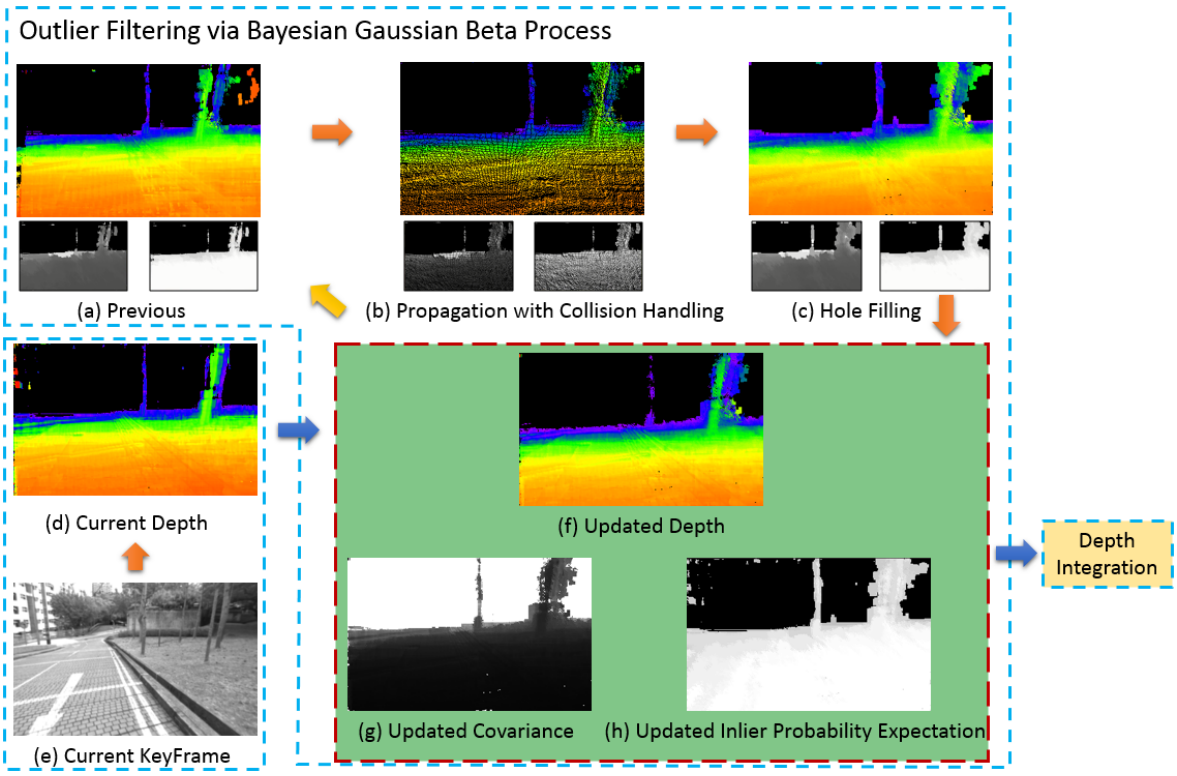



Fig. 3. An illustration of the Bayesian Gaussian beta process. (a) Previous depth hypotheses before propagation. (b) Depth hypotheses after propagation and collision handling. Holes appear due to scale changes (move forward). (c) We fill holes using neighboring depth hypotheses. (d) Current depth estimation (Sect. III-A) with (e) corresponding captured image. We update propagated depth hypotheses via Bayesian inference: (f) Updated depth with (g) corresponding covariance and (h) inlier probability expectation. Colors in depth vary according to the distance from the environment surface to the camera. For covariance, brighter intensities indicate larger covariances, while for inlier probability expectation, brighter intensities indicate higher expectation.

We refer readers to [9] for details of the posterior update for case (a). The posterior update for case (b), however, is novel and not covered in [9] or [7]. We present the mathematic derivation in the following. Recall that the definition of beta function is:

$$\mathcal{B}(\pi_{\mathbf{u}}|a_{\mathbf{u}},b_{\mathbf{u}}) = \frac{\Gamma(a_{\mathbf{u}}+b_{\mathbf{u}})}{\Gamma(a_{\mathbf{u}})\Gamma(b_{\mathbf{u}})}\pi_{\mathbf{u}}^{a_{\mathbf{u}}-1}(1-\pi_{\mathbf{u}})^{b_{\mathbf{u}}-1} \tag{11}$$

where $\Gamma(\cdot)$ is the gamma function. We increase $b_{\mathbf{u}}$ by 1, which leads to:

$$\begin{aligned}\mathcal{B}(\pi_{\mathbf{u}}|a_{\mathbf{u}},b_{\mathbf{u}}+1) &= \frac{\Gamma(a_{\mathbf{u}}+b_{\mathbf{u}}+1)}{\Gamma(a_{\mathbf{u}})\Gamma(b_{\mathbf{u}}+1)}\pi_{\mathbf{u}}^{a_{\mathbf{u}}-1}(1-\pi_{\mathbf{u}})^{b_{\mathbf{u}}} \\ &= \frac{(a_{\mathbf{u}}+b_{\mathbf{u}})\Gamma(a_{\mathbf{u}}+b_{\mathbf{u}})}{b_{\mathbf{u}}\Gamma(a_{\mathbf{u}})\Gamma(b_{\mathbf{u}})}\pi_{\mathbf{u}}^{a_{\mathbf{u}}-1}(1-\pi_{\mathbf{u}})^{b_{\mathbf{u}}} \\ &= \frac{a_{\mathbf{u}}+b_{\mathbf{u}}}{b_{\mathbf{u}}}(1-\pi_{\mathbf{u}})\mathcal{B}(\pi_{\mathbf{u}}|a_{\mathbf{u}},b_{\mathbf{u}}) \end{aligned} \tag{12}$$

where $\Gamma(a+1) = a\Gamma(a)$ is the property of gamma function. Substituting (9) and (12) into (10), we have

$$\begin{aligned}&\mathcal{N}(z_{\mathbf{u}}^n|\mu_{\mathbf{u}}^n,{\sigma_{\mathbf{u}}^n}^2)\mathcal{B}(\pi_{\mathbf{u}}|a_{\mathbf{u}}^n,b_{\mathbf{u}}^n) \\ \approx\ &(1-\pi_{\mathbf{u}})\mathcal{U}(d_{\mathbf{u}}^t|z_l,z_r)\mathcal{N}(z_{\mathbf{u}}|\mu_{\mathbf{u}}^{n-1},{\sigma_{\mathbf{u}}^{n-1}}^2)\mathcal{B}(\pi_{\mathbf{u}}|a_{\mathbf{u}}^{n-1},b_{\mathbf{u}}^{n-1}) \\ \propto\ &\frac{b_{\mathbf{u}}^{n-1}}{a_{\mathbf{u}}^{n-1}+b_{\mathbf{u}}^{n-1}}\mathcal{N}(z_{\mathbf{u}}|\mu_{\mathbf{u}}^{n-1},{\sigma_{\mathbf{u}}^{n-1}}^2)\mathcal{B}(\pi_{\mathbf{u}}|a_{\mathbf{u}}^{n-1},b_{\mathbf{u}}^{n-1}+1) \\ \propto\ &\mathcal{N}(z_{\mathbf{u}}|\mu_{\mathbf{u}}^{n-1},{\sigma_{\mathbf{u}}^{n-1}}^2)\mathcal{B}(\pi_{\mathbf{u}}|a_{\mathbf{u}}^{n-1},b_{\mathbf{u}}^{n-1}+1) \end{aligned} \tag{13}$$

which yields

$$\mu_{\mathbf{u}}^n = \mu_{\mathbf{u}}^{n-1},\ \sigma_{\mathbf{u}}^n = \sigma_{\mathbf{u}}^{n-1},\ a_{\mathbf{u}}^n = a_{\mathbf{u}}^{n-1},\quad b_{\mathbf{u}}^n = b_{\mathbf{u}}^{n-1}+1. \tag{14}$$

*2) Recursive Estimation:* In contrast to [9] and [7], where the temporal and spatial correlations of consecutive depth estimates are ignored, we make use of these correlations. Each depth hypothesis consists of three variables: mean, covariance and inlier probability expectation. We update depth hypotheses in a recursive way. An illustration of the proposed Bayesian Gaussian beta process is shown in Fig. 3. Details are as follows:

**Initialization**: For the first depth estimate, we initialize the depth hypothesis of pixel $\mathbf{u}$: $a_{\mathbf{u}}^0 = b_{\mathbf{u}}^0 = 10$, $\mu_{\mathbf{u}}^0 = d_{\mathbf{u}}^0$, and ${\sigma_{\mathbf{u}}^0}^2 = (\frac{\partial \frac{1}{disp\cdot c_d}}{\partial\ disp})^2\sigma_{disp}^2$ with $disp = \frac{1}{d_{\mathbf{u}}^0\cdot c_d}$ as well as $\sigma_{disp}^2 = 1$. This step is only performed once at the beginning of the Bayesian Gaussian beta process.

**Propagation**: We propagate the depth hypothesis from the previous reference frame $a_{\mathbf{u}}^{n-1}, b_{\mathbf{u}}^{n-1}, \mu_{\mathbf{u}}^{n-1}, \sigma_{\mathbf{u}}^{n-1}$ to the new reference frame $a_{\mathbf{u}'}^{n-1}, b_{\mathbf{u}'}^{n-1}, \mu_{\mathbf{u}'}^{n-1}, \sigma_{\mathbf{u}'}^{n-1}$. Assuming the rotation is small, we have

$$\mu_{\mathbf{u}'}^{n-1} = \mu_{\mathbf{u}}^{n-1} - t_z,\quad {\sigma_{\mathbf{u}'}^{n-1}}^2 = {\sigma_{\mathbf{u}}^{n-1}}^2 + \sigma_{t_z}^2 \tag{15}$$

$$a_{\mathbf{u}'}^{n-1} = a_{\mathbf{u}}^{n-1},\qquad b_{\mathbf{u}'}^{n-1} = b_{\mathbf{u}}^{n-1} \tag{16}$$

where $t_z$ is the translation perpendicular to the camera plane and $\sigma_{t_z}^2$ is the variance of $t_z$. For simplicity, we set $\sigma_{t_z}^2$ to be $0.05^2$ in this work. We do not propagate depth hypotheses

whose inlier probability expectation $E[\mathcal{B}(\pi_{\mathbf{u}}|a_{\mathbf{u}}^{n-1}, b_{\mathbf{u}}^{n-1})] = \frac{a_{\mathbf{u}}^{n-1}}{a_{\mathbf{u}}^{n-1}+b_{\mathbf{u}}^{n-1}}$ is less than 0.4 (i.e., it is unlikely to be an inlier depth estimate).

**Collision Handling**: At all times, we allow at most one depth hypothesis per pixel. However, this is not the case for the scale change (i.e., move backward) as well as occlusion. If two or more depth hypothesis are propagated to the same pixel in the new keyframe, we save the depth hypotheses whose inlier probability expectation $E[\mathcal{B}(\pi_{\mathbf{u}}|a_{\mathbf{u}}^{n-1}, b_{\mathbf{u}}^{n-1})] = \frac{a_{\mathbf{u}}^{n-1}}{a_{\mathbf{u}}^{n-1}+b_{\mathbf{u}}^{n-1}}$ is larger than 0.5 (i.e., not likely to be an outlier depth estimate) as well as whose mean $\mu_{\mathbf{u}}^{n-1}$ is the smallest (for occlusion handling).

**Hole Filling**: Due to scale changes (i.e., move forward), holes may appear after propagation. We set each depth hypothesis in the holes to be the same as its nearest neighbors with distance less than $\tau_d$ pixels. Threshold $\tau_d$ balances the similarity between neighboring depth hypotheses against the variation: a large $\tau_d$ helps to fill more holes at the cost of less accurate depth hypotheses, while a small $\tau_d$ leads to more holes but accurate depth hypotheses. We empirically set $\tau_d$ to be 2 in this work.

**Update**: If the depth hypothesis of pixel $\mathbf{u}$ is null after propagation and hole filling, we initialize it as $a_{\mathbf{u}}^n = b_{\mathbf{u}}^n = 10$, $\mu_{\mathbf{u}}^n = d_{\mathbf{u}}^n$, and $\sigma_{\mathbf{u}}^{n\,2} = (\frac{\partial \frac{1}{disp \cdot c_d}}{\partial\ disp})^2 \sigma_{disp}^2$ with $disp = \frac{1}{d_{\mathbf{u}}^n \cdot c_d}$ as well as $\sigma_{disp}^2 = 1$. Otherwise, we update its posterior distribution of $q(\pi_{\mathbf{u}}, z_{\mathbf{u}}|a_{\mathbf{u}}^n, b_{\mathbf{u}}^n, \mu_{\mathbf{u}}^n, \sigma_{\mathbf{u}}^n)$ according to update formulations mentioned in Sect. III-B.1.

**Output**: For each pixel $\mathbf{u}$, we output its mean $\mu_{\mathbf{u}}^n$, variance $\sigma_{\mathbf{u}}^{n\,2}$ and inlier probability expectation $E[\mathcal{B}(\pi_{\mathbf{u}}|a_{\mathbf{u}}^n, b_{\mathbf{u}}^n)] = \frac{a_{\mathbf{u}}^n}{a_{\mathbf{u}}^n+b_{\mathbf{u}}^n}$ if $E[\mathcal{B}(\pi_{\mathbf{u}}|a_{\mathbf{u}}^n, b_{\mathbf{u}}^n)] > 0.6$. These outputs are needed for the uncertainty-aware depth integration to be discussed in Sect. III-C.

### C. Uncertainty-aware Depth Integration

To build compact and dense 3D models, we adopt the idea of volumetric fusion [13]–[15] to integrate all depth estimates obtained in the previous subsection. In contrast to [13]–[15], where dense reconstructions are based on light-emitting depth cameras that provide high-quality measurements, depth estimation from a moving camera contains noticeable outliers. This motivates us to explicitly model the inlier probability of each depth estimate in the previous subsection and take outliers into account in the depth integration step.

We represent the world as a 3D array of cubic voxels. Each voxel is associated with a signed distance function (SDF) $\phi(\mathbf{x}) : \mathbb{R}^3 \to \mathbb{R}$ and a weight $w(\mathbf{x}) : \mathbb{R}^3 \to \mathbb{R}$. SDF $\phi(\mathbf{x})$ denotes the signed distance between $\mathbf{x}$ and the nearest object surface, and it is positive if it is outside an object and negative otherwise. It can be easily seen that surfaces of objects are zero crossings of signed distance functions (i.e., $\phi(\mathbf{x}) = 0$). $w(\mathbf{x})$ represents the confidence of the sigined distance function. As shown in [23], averaging distance measurements with respective variances over time results in minimizing the weighted sum of the square distances to all ray endpoints for the zero isosurface of the SDF.

Since the major part of the 3D world is usually empty, we use a hash table to index voxels and only store SDFs, as well as their weights, that are near object surfaces [14, 24]. These SDFs are called truncated signed distance functions (TSDFs):

$$\phi_r(\mathbf{x}) = \begin{cases} \phi(\mathbf{x}), & \text{if } ||\phi(\mathbf{x})|| \leq r \\ \text{undefined}, & \text{otherwise} \end{cases}, \tag{17}$$

where $r$ is the truncated distance threshold. For a given depth measurement $d$ with corresponding ray vector direction $\mathbf{f}$, we classify segements of a ray into three regions [24]:

$$u \cdot \mathbf{f} \in \begin{cases} \text{hit region}, & \text{if } ||u - d|| \leq r \\ \text{space carving region}, & \text{if } u \leq d - r \\ \text{undefined}, & \text{otherwise} \end{cases}. \tag{18}$$

*1) Uncertainty-aware TSDF Update:* Voxels within the hit region are updated as:

$$\phi_r(\mathbf{x})' = \frac{\phi_r(\mathbf{x}) \cdot w(\mathbf{x}) + \delta d \cdot \alpha(\delta d)}{w(\mathbf{x}) + \alpha(\delta d)} \tag{19}$$

$$w(\mathbf{x})' = w(\mathbf{x}) + \alpha(\delta d) \tag{20}$$

where $\delta d = \mathbf{x} - d \cdot \mathbf{f}$, and $\alpha(\delta d)$ is the corresponding variance obtained in Sect. III-B. While $\alpha(\delta d)$ in [13]–[15] is a constant, we set it to be the variance obtained from hypothesis filtering to take the uncertainty of motion stereo into account. The initial condition of a TSDF is $\phi_r(\mathbf{x}) = constant$ and $w(\mathbf{w}) = 0$.

*2) Uncertainty-aware Ray Tracing:* Voxels within the space carving region are chiseled away. This operation can be viewed as removing potential depth outliers by visibility constraints (i.e., segments between two endpoints of a ray are empty). Free-space carving makes sense for the reason that we care more about which part of the scene does not contain surfaces (for motion planning) than what is inside objects. However, free-space carving with outlier depth measurements is harmful to the built model, as part of it may be wrongly chiseled away. Therefore, we only do ray tracing if the expectation of inlier probability obtained in Sect. III-B is large than 0.8.

While voxels are sufficient for motion planning [25, 26], colors and textures are more suitable for visualization and debugging. We include an optional step, marching cubes [12], to extract polygonal meshes of an isosurface from a three-dimensional discrete scalar field.

## IV. Experiments

The whole system is implemented in C++, with ROS as the interfacing robotics middleware. All testings are carried out in a commodity Lenovo laptop Y50 with an i7-4720HQ CPU and a mobile GTX-960M GPU. The depth estimation module is run on the GPU while the hypothesis filtering module and the uncertainty-aware depth integration module are implemented in the CPU. These modules are placed on different threads to utilize the multi-core CPU architecture.

TABLE I

COMPARISION OF THE AVARAGE COMPUTATION TIME ON THE TUM RGB-D SLAM & IC-NUIM DATASET.

| **Methods** | **Ours-T** | **Ours-T+S** | **Ours-T+S+D** | **Ours-T+S+D+H** | **REMODE** [7] | **VI-MEAN** [16] |
|---|---|---|---|---|---|---|
| Average computation time (ms) | 33.71 | 41.20 | 42.10 | 51.02 | 31.31 | 80.02 |

TABLE II

COMPARISION OF THE AVERAGE MAPPING DENSITY ON THE TUM RGB-D SLAM DATASET & ICL-NUIM DATASET..

| **Dataset Name** | **Sequence Name** | **Ours-T** | **Ours-T+S** | **Ours-T+S+D** | **Ours-T+S+D+H** | **REMODE** [7] | **VI-MEAN** [16] |
|---|---|---|---|---|---|---|---|
| TUM RGB-D SLAM | freiburg2_desk | 60.57 | 64.14 | 62.34 | 46.52 | 32.63 | 85.44 |
| | freiburg3_nostructure_texture_far | 65.42 | 76.68 | 74.62 | 71.40 | 44.62 | 69.13 |
| | freiburg3_sitting_halfsphere | 67.59 | 69.80 | 66.17 | 61.89 | 22.29 | 56.60 |
| | freiburg3_structure_texture_far | 80.26 | 87.05 | 86.72 | 80.88 | 34.16 | 76.26 |
| | freiburg3_structure_notexture_far | 76.68 | 85.49 | 84.05 | 82.28 | 43.85 | 80.30 |
| | freiburg3_sitting_xyz | 67.37 | 70.59 | 67.05 | 65.16 | 22.61 | 51.32 |
| ICL-NUIM | living room of kt0 | 87.26 | 91.82 | 90.01 | 88.93 | 68.80 | 96.05 |
| | living room of kt1 | 91.40 | 93.51 | 93.17 | 90.52 | 67.10 | 97.00 |
| | living room of kt2 | 88.58 | 90.00 | 89.46 | 88.49 | 67.13 | 94.45 |
| | living room of kt3 | 91.56 | 93.61 | 92.33 | 91.90 | 62.32 | 86.90 |
| | office room of kt0 | 89.67 | 93.34 | 91.37 | 87.45 | 32.08 | 90.57 |
| | office room of kt1 | 90.09 | 95.55 | 93.90 | 88.90 | 25.24 | 95.66 |
| | office room of kt2 | 89.53 | 92.08 | 91.66 | 87.85 | 39.35 | 94.15 |
| | office room of kt3 | 91.35 | 96.12 | 93.37 | 89.22 | 27.87 | 93.50 |

### *A. TUM RGB-D SLAM Dataset & ICL-NUIM Dataset*

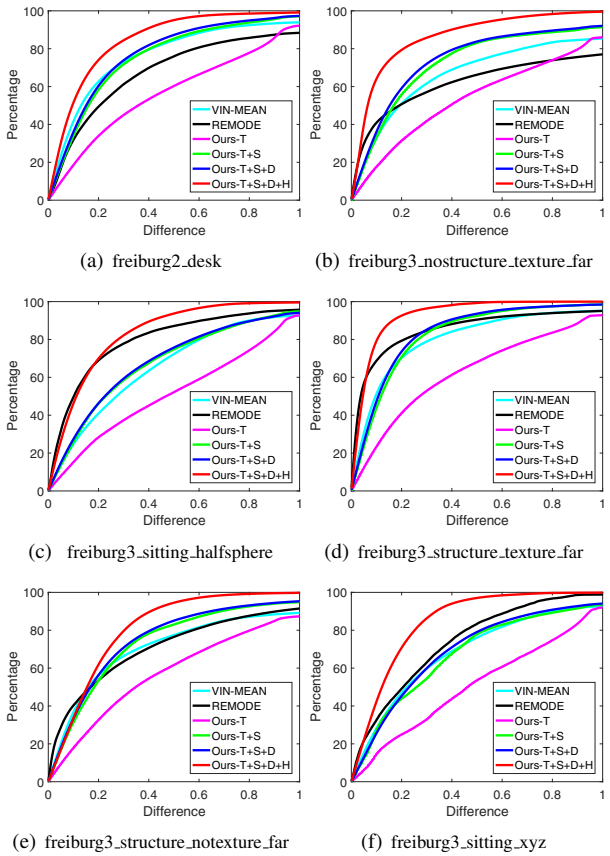
TABLE III

STATISTICS OF SELECTED SEQUENCES ON THE TUM RGB-D SLAM DATASET.

| **Sequence Name** | **Structure** | **Texture** | **Dynamic Objects** |
|---|---|---|---|
| freiburg2_desk | √ | √ | × |
| freiburg3_nostructure_texture_far | × | √ | × |
| freiburg3_sitting_halfsphere | √ | √ | √ |
| freiburg3_structure_texture_far | √ | √ | × |
| freiburg3_structure_notexture_far | √ | × | × |
| freiburg3_sitting_xyz | √ | √ | √ |

We evaluate the mapping performance of our monocular depth estimation obtained after hypothesis filtering on the TUM RGB-D SLAM dataset [1] and the ICL-NUIM dataset [2]. We use ground truth poses from datasets as mapping pose inputs to ensure correct mapping metric for evaluation. Depths from Microsoft Kinect (TUM RGB-D SLAM dataset) or ray tracking (ICL-NUIM dataset) are used for mapping performance evaluation. Since the TUM RGB-D SLAM dataset is originally for odometry, we select some static sequences that are suitable for dense mapping. These selected sequences cover various environment conditions (Table III). We use an ablation study for analysis: T denotes temporal cost aggregation (Sect. III-A.1); S denotes spatial regulation (Sect. III-A.2); D denotes local region discussion & depth refinement (Sect. III-A.3); H denotes hypothesis filtering (Sect. III-B). We also compare our approach with state-of-the-art methods: REMODE [7] and VI-MEAN [16]. Three measurement metrics are used for comparison:

- Average computation time (ms): the average computation time of each depth computation. It evaluates the online performance of mobile applications.
- Average mapping density (%): the average density of depth estimates for each depth estimation. It plays a key role in the safety of mobile robots. A higher density helps better obstacle avoidance.
- Per-depth error percentage (% w.r.t. m): the percentage of depth difference, between the estimated depth values and the ground truth depth values, within the difference threshold $e_d$. It evaluates mapping accuracy. We prefer higher percentage of small estimation errors.




(a) freiburg2_desk (b) freiburg3_nostructure_texture_far

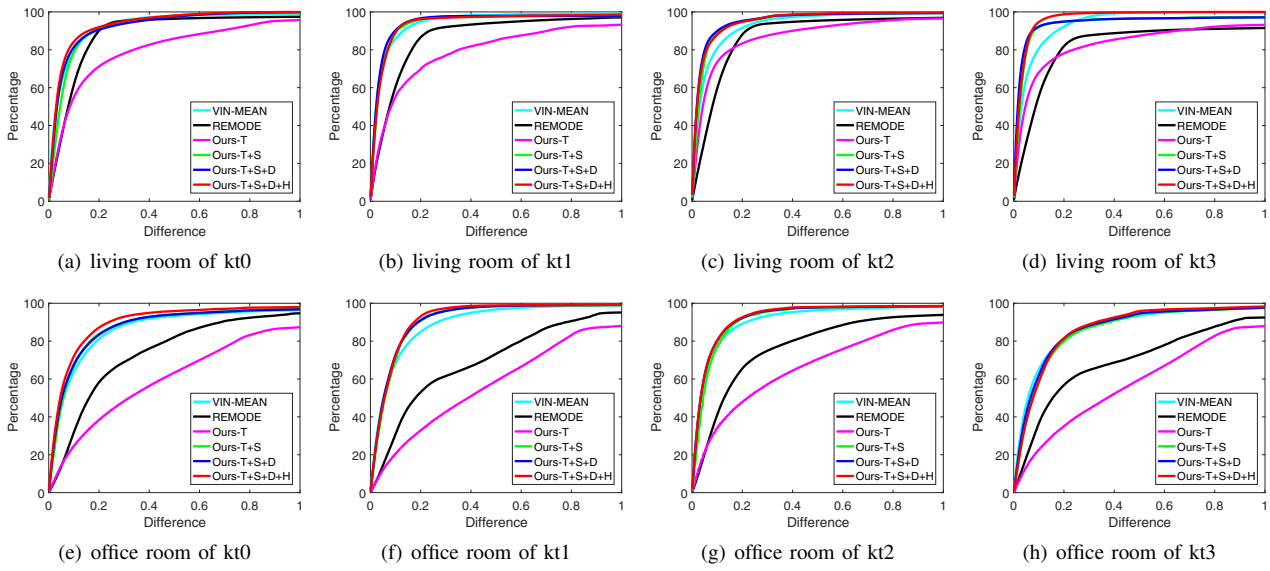
(c) freiburg3_sitting_halfsphere (d) freiburg3_structure_texture_far

(e) freiburg3_structure_notexture_far (f) freiburg3_sitting_xyz

Fig. 4. Comparision of per-depth error percentage (% w.r.t. m) in sequences of the TUM RGB-D SLAM dataset. We calculate the percentage (vertical axis) of depth difference, between the estimated depth values and the ground truth depth values, within the difference threshold $e_d$ (horizontal axis). Our approach (T+S+D+H) achieves higher mapping accuracy than state-of-the-art methods (REMODE [7] and VI-MEAN [16]).

[1] https://vision.in.tum.de/data/datasets/rgbd-dataset

[2] https://www.doc.ic.ac.uk/~ahanda/VaFRIC/iclnuim.html




(a) living room of kt0 (b) living room of kt1 (c) living room of kt2 (d) living room of kt3

(e) office room of kt0 (f) office room of kt1 (g) office room of kt2 (h) office room of kt3

Fig. 5. Comparision of per-depth error percentage (% w.r.t. m) in sequences of the ICL-NUIM dataset. We calculate the percentage (vertical axis) of depth difference, between the estimated depth values and the ground truth depth values, within the difference threshold $e_d$ (horizontal axis). Our approach (T+S+D+H) achieves higher mapping accuracy than state-of-the-art methods (REMODE [7] and VI-MEAN [16]).

Since the image resolution of sequences on both datasets are the same, the computation times of different approaches on different sequences are similar. We take the average of the computation times, and summarize them in Table I.

The comparison of average mapping density is shown in Table II. For REMODE [7], only converged depth estimates are used in the evaluation. Others (i.e. not converged) are not used since they are highly unreliable. Using these depth estimates leads to very low average mapping accuracy. Fig. 4 and Fig. 5 show detailed illustrations of the mapping accuracy on both datasets. We also give a visual comparison between different methods in Fig. 6 using snapshots from the depth estimations at one of the frames in the freiburg2_desk testing sequence of the TUM RGB-D dataset.

We firstly analyze the influence of different components on our approach. The step of temporal cost aggregation is the most time-consuming step. It forms the basis of all following calculations. Applying winter-takes-all strategy after temporal cost aggregation achieves more than 60 average mapping density. However, the corresponding mapping accuracy is very low. The step of spatial regulation, which utilizes the spatial correlation of neighboring depth estimates, not only increases the mapping density, but also increases the mapping accuracy. The local region discussion step slightly reduce the mapping density by rejecting unreliable depth estimates, while the depth refinement step slightly increase the mapping accuracy. The last step, hypothesis filtering, improves the mapping accuracy greatly at the cost of some mapping density reduction. Our hypothesis filtering strategy explicitly makes use of the temporal and spatial correlations of consecutive depth estimates. Consistent depth values are improved while inconsistent ones are removed.

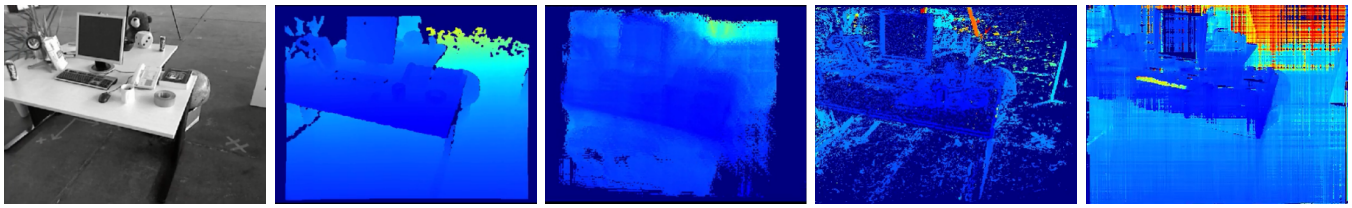
We then compares our approach (T+S+D+H) against REMODE [7] and VI-MEAN [16]. REMODE [7] runs fastest, as it estimates pixel depth independently, without taking the spatial correlation into consideration. It outputs depth estimates that are in well-textured regions (Fig. 6(d)). VI-MEAN [16] runs slowest and achieves mapping density usually higher than our approach. The main disadvantage of VI-MEAN [16] is that it does not model outliers, which is demonstrated by the noticeable outliers in its depth estimation (Fig. 6(e)). Our approach achieves a good balance between mapping density and mapping accuracy.

### B. Online Indoor and Outdoor Dense Reconstructions

We present online dense reconstructions with a monocular visual-inertial sensor suite. We use the method in [22] for online pose estimation. The voxel size in the depth integration step is 0.1 meters. Average computation times of depth estimation (T+S+D+H) and uncertainty-aware depth integration are 55.14 ms and 31.18 ms respectively. Snapshots of depths obtained after hypothesis filtering in both indoor and outdoor environments are shown in Fig. 7, while final dense reconstructions are shown in Fig. 1. More details of the online depth estimations and dense reconstructions are available at `https://1drv.ms/v/s!ApzRxvwAxXqQmlW9ZOrp9hdA7ude`.

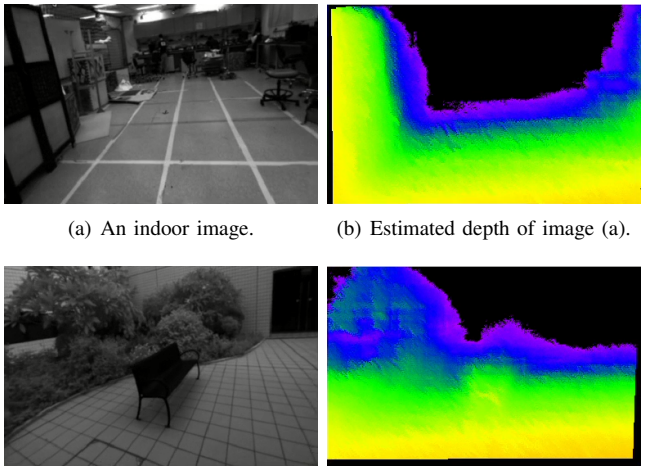
## V. Conclusion and Future Work

In this work, we present a probabilistic approach for monocular dense reconstruction in real time, which makes use of both the spatial and temporal correlations between consecutive depth estimations. In addition to the depth mean, we evaluate its confidence and inlier probability expectation simultaneously in a recursive and probabilistic way. We also take the uncertainty of the depth estimations into account in the depth integration step. Extensive experiments on the TUM RGB-D SLAM dataset and the ICL-NUIM dataset as well as online indoor and outdoor environments demonstrate the effectiveness and efficiency of our presented approach.



(a) A captured image. (b) Ground truth depth from Microsoft Kinect. (c) Depth from our approach (T+S+D+H). (d) Depth from REMODE [7]. (e) Depth from VI-MEAN [16].

Fig. 6. A visual comparison between our proposed approach and state-of-the-art methods (REMODE [7] and VI-MEAN [16]) at one of frames in the freiburg2_desk testing sequence. (a) A captured image. (b) Corresponding ground truth depth from Microsoft Kinect. (c) Depth estimation from our approach (T+S+D+H). (d) Depth estimation from REMODE [7]. (e) Depth estimation from VI-MEAN [16]. Colors vary w.r.t. the distances to the camera. Pixels in dark blue mean no depth estimates.



(a) An indoor image. (b) Estimated depth of image (a).

(c) An outdoor image. (d) Estimated depth of image (c).

Fig. 7. Snapshots of depths obtained after hypothesis filtering during indoor and outdoor experiments. (a)(c) are indoor and outdoor image and (c) (d) are their estimated depths. Colors vary w.r.t. distances to the camera. Dense reconstruction results can be found in Fig. 1.

In the future, we will apply our approach to real-world applications, such as autonomous navigation and AR.

## REFERENCES


[1] S. Shen, N. Michael, and V. Kumar, "Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft MAVs," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Seattle, WA, May 2015.

[2] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Consistency analysis and improvement of vision-aided inertial navigation," *IEEE Trans. Robot.*, vol. 30, no. 1, pp. 158–176, Feb. 2014.

[3] Li, M. and Mourikis, A.I., "High-precision, consistent EKF-based visual-inertial odometry," *Intl. J. Robot. Research*, vol. 32, no. 6, pp. 690–711, May 2013.

[4] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[5] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," in *Proc. of Robot.: Sci. and Syst.*, 2015.

[6] R. A. Newcombe, S. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *IEEE International Conference on Computer Vision*, 2011, pp. 2320–2327.

[7] M. Pizzoli, C. Forster, and D. Scaramuzza, "REMODE: Probabilistic, monocular dense reconstruction in real time," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, 2014.

[8] V. Pradeep, C. Rhemann, and S. Izadi, "MonoFusion: Real-time 3D reconstruction of small scenes with a single web camera," in *IEEE International Symposium on Mixed and Augmented Reality*, 2013.

[9] G. Vogiatzis and C. Hernandez, "Video-based, real-time multi-view stereo," *Image and Vision Computing*, vol. 29, pp. 434–441, 2011.

[10] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.

[11] A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching," in *Asian Conference on Computer Vision*, 2010.

[12] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 163–169, Aug. 1987.

[13] N. R. A., I. Shahram, H. Otmar, M. David, K. David, D. A. J., K. Pushmeet, S. Jamie, H. Steve, and F. Andrew, "Kinectfusion: Real-time dense surface mapping and tracking," in *The IEEE International Symposium on Mixed and Augmented Reality*, October 2011.

[14] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D reconstruction at scale using voxel hashing," *ACM Trans. Graph.*, vol. 32, no. 6, p. 169, 2013.

[15] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "Elasticfusion: Real-time dense SLAM and light source estimation," *Intl. J. of Robotics Research*, 2016.

[16] Y. Lin, F. Gao, T. Qin, W. Gao, T. Liu, W. Wu, Z. Yang, and S. Shen, "Autonomous aerial navigation using monocular visual-inertial fusion," *Journal of Field Robotics*, 2017.

[17] J. Stühmer, S. Gumhold, and D. Cremers, "Real-time dense geometry from a handheld camera," in *Proceedings of the DAGM Symposium on Pattern Recognition*, 2010.

[18] T. Schops, T. Sattler, C. Hane, and M. Pollefeys, "3D modeling on the go: Interactive 3D reconstruction of large-scale scenes on mobile devices," in *International Conference on 3D Vision*, 2015.

[19] W. N. Greene, K. Ok, and P. Lommel, "Multi-level mapping: Real-time dense monocular SLAM," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, 2016.

[20] A. Concha and J. Civera, "Dense piecewise planar tracking and mapping from a monocular sequence," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, 2015.

[21] L. Teixeira and M. Chli, "Real-time local 3D reconstruction for aerial inspection using superpixel expansion," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, 2017.

[22] Z. Yang and S. Shen, "Monocular visual-inertial state estimation with online initialization and camera-IMU extrinsic calibration," *IEEE Transactions on Automation Science and Engineering*, vol. 14, pp. 39–51, 2017.

[23] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, 1996.

[24] M. Klingensmith, I. Dryanovski, S. S. Srinivasa, and J. Xiao, "Chisel : Real time large scale 3D reconstruction onboard a mobile device using spatially-hashed signed distance fields," in *Proc. of Robot.: Sci. and Syst.*, 2015.

[25] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, May 2009.

[26] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online UAV replanning," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Oct 2016.