Quadtree-accelerated Real-time Monocular Dense Mapping

Kaixuan Wang, Wenchao Ding, and Shaojie Shen

***Abstract*— In this paper, we propose a novel mapping method for robotic navigation. High-quality dense depth maps are estimated and fused into 3D reconstructions in real-time using a single localized moving camera. The quadtree structure of the intensity image is used to reduce the computation burden by estimating the depth map in multiple resolutions. Both the quadtree-based pixel selection and the dynamic belief propagation are proposed to speed up the mapping process: pixels are selected and optimized with the computation resource according to their levels in the quadtree. Solved depth estimations are further interpolated and fused temporally into full resolution depth maps and fused into dense 3D maps using truncated signed distance function (TSDF). We compare our method with other state-of-the-art methods using the public datasets. Onboard UAV autonomous flight is also used to further prove the usability and efficiency of our method on portable devices. For the benefit of the community, the implementation is also released as open source at `https://github.com/HKUST-Aerial-Robotics/open_quadtree_mapping`.**

I. INTRODUCTION

Cameras are widely used in robotic systems for they provide informative data about the environment while maintaining low power consumption and producing small footprints. Specifically, we are interested in Monocular Visual Inertial Systems [1] [2] because they can provide robots with the necessary information for navigation using the minimum sensor set of a single camera and an inertial measurement unit (IMU). However, the sparse or semi-dense maps built for localization are not sufficient for missions like obstacle avoidance. To perceive the surrounding environment, additional sensors like RGB-D cameras, Lidars or stereo cameras are usually used. However, the weight, size and power consumption of these sensors make them unsuitable for robots whose payload and power are limited. In this paper, we propose a monocular dense mapping method that can estimate high-quality depth maps and build dense 3D models in real-time using a single localized moving camera with no extra sensors as shown in Fig. 1.

Generating high-quality dense depth maps is the key step in the monocular dense mapping system. To be applied in robotic applications, the depth map estimation must be efficient in time and densely cover the image including low-texture regions. Our method is inspired by two observations in monocular depth estimation:

(1) Global optimizations [5], [6] are often required to estimate the depth of low-texture regions and smooth the depth map by minimizing one energy function considering both the patch matching cost and the depth map smoothness.

All authors are with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, SAR China. {kwangap, wdingae, eeshaojie}@ust.hk

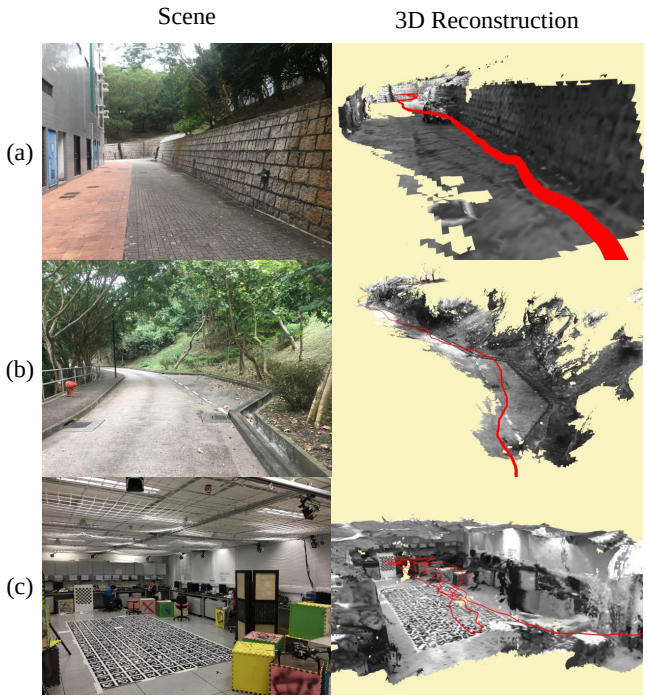
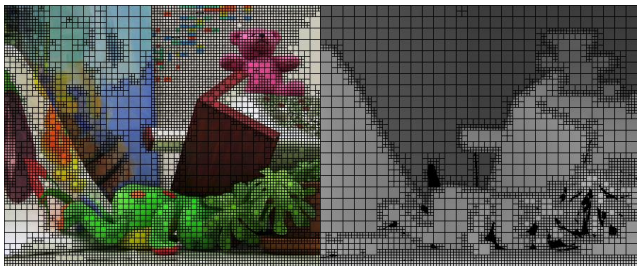


Fig. 1. Our method can generate dense 3D maps in real-time using a single localized moving camera obtained through a monocular visual-inertial state estimation [3]. The left column is a snap of the environment and the right column is the 3D map built in real-time. Red lines in the 3D maps represent the trajectory of the camera. A variety of outdoor and indoor environments are tested. Outdoor environments consist of repeated patterns in (a) and fine structures like trees in (b). Indoor environments consist of repeated patterns on the ground and textureless region like the floor in (c). More experiments are available in the supplementary video.

However, the expensive computation prevents them from being widely used in real-time applications. (2) Pixels that are within the same quadtree block of the intensity image share similar intensities and depth values. Quadtree structures are widely used (e.g., in image coding [7]) that pixels are organized in a tree structure according to their local texture: pixels with similar intensity values can be grouped together by assigning them to the same quadtree block and represented in the corresponding resolution. As illustrated in Fig. 2, in most of the cases, the quadtree level of a pixel in the intensity image is equal or finer than that in the depth image. This means that pixels which share the same quadtree block in the intensity image also belong to the same block in the depth image, and thus share similar depth values, but not vice versa. In other words, the depth of a pixel can be represented in the resolution corresponding to its quadtree level in the intensity image.

Inspired by these two observations, we propose a novel monocular dense mapping method that estimates dense depth maps and build 3D reconstructions using a single localized moving camera. Depth density and accuracy are improved by



(a) intensity image (b) depth image

Fig. 2. One image and the corresponding depth map from Middlebury Stereo Datasets [4] are divided into quadtree structures according to intensity and depth values. In most of the cases, pixels that share the same quadtree block in the intensity image also belongs to the same block in the depth map, but not vice versa. This means that pixels that are within the same quadtree block of the intensity image share similar intensity values and depth values.

global optimization and the computation burden is reduced by estimating the depth map in multiple resolutions corresponding to the quadtree structure of the intensity image. In specific, pixels are selected according to their quadtree levels with density proportional to the resolution they need to be estimated. Dynamic belief propagation is used to estimate the depth of selected pixels in a coarse-to-fine manner, where the depth is extracted at corresponding resolutions to gain efficiency. All the depth estimations are interpolated into a full dense depth map and fused temporally with former depth maps. Lastly, the depth maps are fused into a high-quality global 3D map.

We compare our work with other state-of-the-art monocular dense mapping methods using the public datasets. Onboard autonomous flights in different environments are also used to prove the usability of our method in UAV applications.

The contributions of our method are the following:

像素选择方法

- A novel pixel selecting method that selects pixels to estimate the depth according to their levels in the quadtree structure of the image. The selected pixels spread across the whole image and draw more attention to high-texture regions.

动态传播方式

- A dynamic belief propagation that estimates the depth of selected pixels in a coarse-to-fine manner. Pixels are optimized with the computation resource according to their quadtree levels. The optimization generates high-quality depth estimation while maintaining efficiency by skipping coarse resolution pixels.

基于已知位姿信息

- A monocular dense reconstruction system which generates high-quality dense depth maps in real-time using only a localized moving camera. As demonstrated in the experiments and the supplementary video, the depth maps can be directly fused into 3D maps for 3D reconstruction and UAV autonomous flight. The system is also released as open source for the community.

II. RELATED WORK

Many methods have been proposed to solve the monocular dense mapping problem.

In computer vision, the problem of reconstructing the environment using images is also known as Structure from Motion (SfM). Given a sequence of images, SfM reconstructs the structure of the environment and estimates camera motions. Although achieving impressive results, the result is usually sparse and need to be processed offline. On the other hand, our method uses images and corresponding camera motions to densely reconstruct the environment in real-time.

DTAM [8], VI-MEAN [9] and REMODE [10] are methods that solves the dense mapping problem using global or semi-global information. DTAM [8] builds cost volumes by accumulating multiple frames and extracts depth maps using a total variation optimization. VI-MEAN [9] uses a semi-global match [11] (SGM) to regularize the cost volume. However, the 4-path optimization causes the "streaking" artifacts in the estimated depth maps of VI-MEAN [9]. REMODE [10] uses a total variation to smooth the depth images estimated by a probabilistic update. Since the optimization does not search the depth in the cost volume like DTAM [8], REMODE [10] cannot handle low-texture regions well.

MonoFusion [12], MobileFusion [13] and 3D Modeling on the Go [14] build dense maps from two-view stereo matching. Every input image is compared with a selected measurement frame. Although two-view matching is fast, these methods contain many outliers in monocular cases. Post-processing techniques, such as Left-Right consistency check, large cost remove, consistency over time, are used to deal with the problem.

Multi-level mapping [15] is the method that is most similar to our work. In Multi-level mapping [15], quadtree structure is used to increase the local texture of patches so that more pixels can be estimated using local information. A total variation optimization is then carried out to smooth the depth map from multi-resolution. The method is efficient and increases the density of the depth maps. The most important difference between Multi-level mapping [15] and our method is the density of estimated depth maps. Using global optimization, our method can recover the depth of regions with low texture, even no texture. On the other hand, Multi-level mapping [15] estimates the depth of pixels using local intensity information. Thus, only the depth of pixels that have enough gradient can be estimated. The density of the estimated depth maps is important to the safety of robotic applications and the completeness of the 3D reconstruction.

III. SYSTEM OVERVIEW

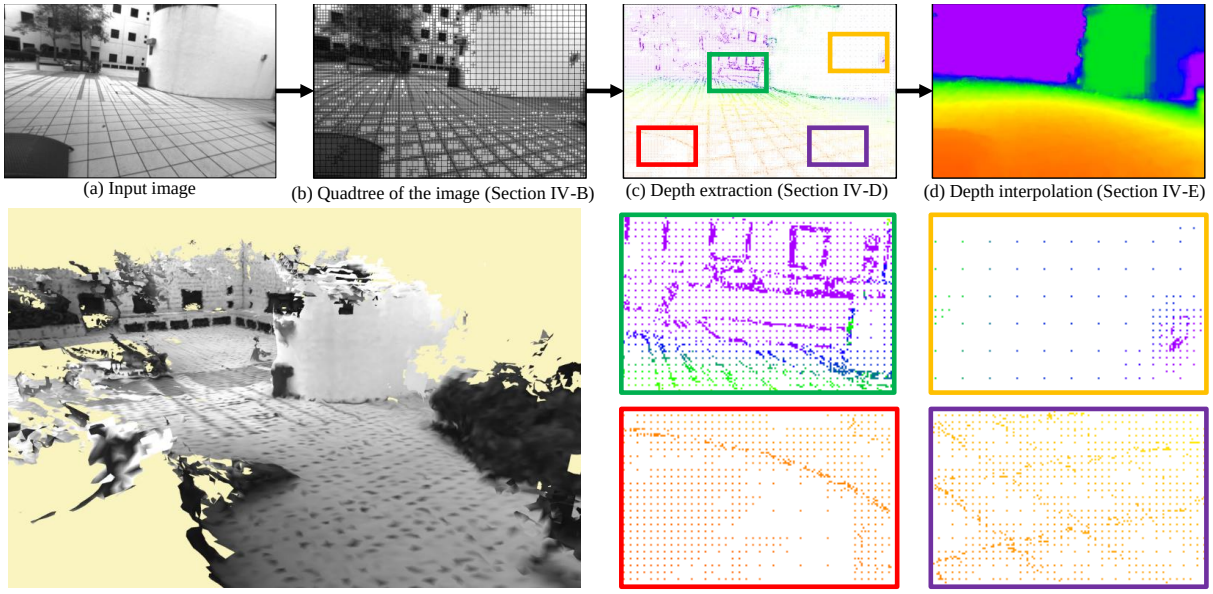
Inspired by the relationship between the quadtree structures of the image and the corresponding depth map, our method solves the depth estimation in multi-resolution and fuse them into high-quality full-resolution depth maps and dense meshes.

Our method consists of five steps: (1) quadtree based pixel selection, (2) matching cost vector calculation, (3) dynamic belief propagation, (4) depth interpolation and temporal fusion, and (5) TSDF fusion. The pipeline and the intermediate results of the system are shown in Fig. 3. Input images are first transformed into a 3-level quadtree structure and each pixel will be assigned to a quadtree level. Pixels are selected across the image based on their quadtree levels and

动态置信度传播算法.

深度插值和时间融合

truncated signed distance function
截断有符号距离函数



(a) Input image (b) Quadtree of the image (Section IV-B) (c) Depth extraction (Section IV-D) (d) Depth interpolation (Section IV-E)

(e) Fused dense 3D meshes (Section IV-F) (f) Zoomed in details of extracted depth (c)

Fig. 3. Intermediate results of our method to generate dense depth maps and meshes. Depth values in (c) and (d) are rainbow color-coded with red representing $0.5m$ and purple representing more than $10.0m$. Our method first selects and estimates the depth of pixels according to the quadtree structure of the image and then fuses them into high-quality full-resolution depth maps and meshes. Extracted depth in (c) contains depth from dynamic belief propagation of Section IV-D and depth estimations from high-gradient estimation of Section IV-E. Note that the extracted depth in (c) spreads across the whole image including low-texture areas and captures texture and depth discontinuity regions well which is essential for further fusion.

the matching cost vector is computed for global optimization. Dynamic belief propagation extracts the depth estimation for each selected pixel in the corresponding resolution. Finally, the depth estimations are interpolated and fused temporally into full-resolution depth maps and are used to build global 3D maps.

IV. Monocular Dense Reconstruction

A. Notation

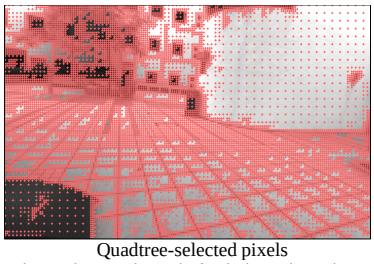
Let $\mathbf{T}_{w,k} \in \mathbb{SE}(3)$ be the pose of the camera with respect to the world frame w when taking the k-th image. We denote the k-th intensity image as $I_k : \Omega \subset \mathbb{R}^2 \mapsto \mathbb{R}$. A 3D point $\mathbf{x}_c = (x, y, z)^{\mathrm{T}}$ in the camera frame can be projected into the image as a pixel $\mathbf{u} := (u, v)^{\mathrm{T}} \in \Omega$ using the camera projection function: $\pi(\mathbf{x}_c) = \mathbf{u}$. A pixel can be back-projected into the camera frame c as a point: $\mathbf{x}_c = \pi^{-1}(\mathbf{u}, d)$ where d is the depth of the pixel.

In the following sections, the pose $\mathbf{T}_{w,k}$ is given by the monocular visual-inertial system or given by the datasets. We assume that the camera pose is accurate and fixed for each frame that will not change in the future which is the case for typical visual odometry systems. Reconstructing the environment with loop closure is beyond the scope of this paper.

The input of our system is a sequence of images I_i and their corresponding poses $\mathbf{T}_{w,i}$. For each input image I_k, the depth map is estimated using I_k as the reference frame and multiple former input images as measurement frames.

B. Quadtree Based Pixel Selection

We use the quadtree to find the most suitable resolution representation for each pixel. For each input image I_k, we compute a 3-level quadtree structure with the finest level corresponding to 4×4 pixel block and the coarsest level corresponding to 16×16 pixel block. As illustrated in Fig. 2, pixels in the same block of the quadtree share similar intensities and depth values thus can be estimated together. The first pixel in the quadtree block is selected to construct the matching cost vector and estimate the depth using dynamic belief propagation. Fig. 4 shows an example of selecting pixels based on the quadtree structure of the image.



Quadtree-selected pixels

Fig. 4. An example to show selected pixels based on the extracted quadtree of Fig. 3(b), red points are selected pixels to estimate the depth. The selected pixels cover the whole image in different densities according to the quadtree structure of the image.

For pixels that are not selected in this step, their depth estimation will be obtained by interpolation and fusion in Section IV-E. Selecting pixels according to the quadtree structure of the intensity image enables us to solve the depth maps in good efficiency without sacrificing much of the precision.

C. Matching Cost Vector Calculation

The matching cost vectors of quadtree-selected pixels are calculated for the use in dynamic belief propagation. Our method selects five previous nearby frames as measurement frames. In this work, we sample N_d depth values uniformly distributed on the inverse depth space. The sampled depth d

and the corresponding depth index l has the relationship:

$$\frac{1}{d} = (\frac{1}{d_{min}} - \frac{1}{d_{max}})\frac{l}{N_d - 1} + \frac{1}{d_{max}}, \quad (1)$$

where $l \in [0, N_d - 1]$, $l \in \mathbb{N}$. d_{max} and d_{min} are the maximum and the minimum sampled distance.

For each quadtree-selected pixel $\mathbf{u}_k$ in image I_k, the cost of depth d is defined as

$$\mathbf{C}(\mathbf{u}_k, d) = \frac{1}{|\mathcal{M}|} \sum_{I_m \in \mathcal{M}} SAD(I_m, I_k, \mathbf{u}_k, d), \quad (2)$$

where $\mathcal{M}$ is the set of selected measurement frames. For each measurement frame I_m and depth d, the corresponding pixel of $\mathbf{u}_k$ in the measurement frame I_m is given by $\mathbf{u}_m = \pi(\mathbf{T}_{w,m}^{-1}\mathbf{T}_{w,k}\pi^{-1}(\mathbf{u}_k, d))$. $SAD(I_m, I_k, \mathbf{u}_k, d)$ simply calculates the sum of absolute difference of two 3×3 patches located at $\mathbf{u}_m$ and $\mathbf{u}_k$ in image I_m and I_k.

D. Dynamic Belief Propagation

A dynamic belief propagation is proposed to estimate the depth of quadtree-selected pixels distributed across the image. Belief propagation works by passing messages along connected image grids to regularize the estimated depth maps. Messages are N_d dimensions vectors representing the belief of the depth distribution from one pixel to its neighbor. Every pixel in the image grid passes and receives four messages to and from its 4-neighbor. Combining the matching cost vector and the messages, the final belief vector $\mathbf{B}_{\mathbf{u}_k}(d)$ for pixel $\mathbf{u}_k$ in image I_k is given as

$$\mathbf{B}_{\mathbf{u}_k}(d) = \mathbf{C}(\mathbf{u}_k, d) + \sum_{\mathbf{s} \in \mathcal{N}(\mathbf{u}_k)} (m_{\mathbf{s} \to \mathbf{u}_k}(d)), \quad (3)$$

where $\mathcal{N}(\mathbf{u}_k)$ is the 4-neighbor of pixel $\mathbf{u}_k$ and $m_{\mathbf{s} \to \mathbf{u}_k}$ is the message passing from pixel $\mathbf{s}$ to $\mathbf{u}_k$. The depth that minimizes the belief vector is selected as the estimation of the pixel.

The core part of belief propagation is to iteratively update the messages. All the messages are initialized as zero vectors and updated in parallel as

$$m^{raw}_{\mathbf{p} \to \mathbf{q}}(d_\mathbf{p}) = \mathbf{C}(\mathbf{p}, d_\mathbf{p}) + \sum_{\mathbf{s} \in \mathcal{N}(\mathbf{p}) \setminus \mathbf{q}} m_{\mathbf{s} \to \mathbf{p}}(d_\mathbf{p}), \quad (4)$$

$$m_{\mathbf{p} \to \mathbf{q}}(d_\mathbf{q}) = \min_{d_\mathbf{p}}(V(d_\mathbf{q}, d_\mathbf{p}) + m^{raw}_{\mathbf{p} \to \mathbf{q}}(d_\mathbf{p})), \quad (5)$$

where $V(d_\mathbf{q}, d_\mathbf{p})$ regularizes the depth map by punishing the differece between $d_\mathbf{q}$ and $d_\mathbf{p}$.

The dynamic belief propagation is sped up in two ways. Firstly, the message update time is reduced to $O(log(N_d))$ by simplifying the regularize function $V(d_\mathbf{q}, d_\mathbf{p})$ and using parallel acceleration. Secondly, the update iterations are reduced by extracting the depth estimation at multi-resolution according to the quadtree levels of the selected pixels.

The time complexity of Equation 5 is $O(N_d^2)$ and Felzenszwalb *et al*. [6] reduces it to $O(N_d)$ by a forward and backward scan. We further reduce the message update time using parallel acceleration by parallel acceleration and adopting the regularization function from SGM [11]:

$$V(d_\mathbf{p}, d_\mathbf{q}) = \begin{cases} 0 & if\ |l(d_\mathbf{p}) - l(d_\mathbf{q})| = 0 \\ P1 & if\ |l(d_\mathbf{p}) - l(d_\mathbf{q})| = 1 \\ P2 & if\ |l(d_\mathbf{p}) - l(d_\mathbf{q})| \geq 1 \end{cases}, \quad (6)$$

where $l(d_\mathbf{p})$ is the depth index of $d_\mathbf{p}$ as defined in Equation 1, $P1$ and $P2$ controls the smoothness of the depth maps as that in SGM. Note that although the Equation 6 is the same to that in SGM [11], the message updating and passing is different. SGM aggregates the cost along several predefined 1D paths, while our method passes the messages on 2D image grid. 2D global optimization enables our method to generate smooth depth map without the "streaking" artifacts in SGM. Updating the message from pixel $\mathbf{p}$ to pixel $\mathbf{q}$ can be accelerated to $O(log(N_d))$ time with the help of parallel reduce operation. Algorithm 1 shows the way to update the messages of a pixel in $O(log(N_d))$ time. Messages are updated in parallel using Algorithm 1.

Algorithm 1 Message Update for pixel $\mathbf{p}$ with N_d threads

Require:
data term $\mathbf{C}(\mathbf{p}, d_i)$
message $m_{\mathbf{s} \to \mathbf{p}}(d_i), \mathbf{s} \in \mathcal{N}(\mathbf{p}) \setminus \mathbf{q}$
each thread's index $i \in [0, N_d - 1]$

compute $m^{raw}_{\mathbf{p} \to \mathbf{q}}(d_i)$ as Equation 4
$m_{min}(d_i) \leftarrow m^{raw}_{\mathbf{p} \to \mathbf{q}}(d_i)$
$step \leftarrow \lfloor N_d/2 \rfloor$
$synchronize\ threads$
for $step > 0$ **do**
  if $i < step$ **and** $m_{min}(d_i + step) < m_{min}(d_i)$ **then**
    $m_{min}(d_i) \leftarrow m_{min}(d_i + step)$
  end if
  $step \leftarrow \lfloor step/2 \rfloor$
  $synchronize\ threads$
end for
$min_{raw} \leftarrow m_{min}(0)$
$m_i \leftarrow \min(m^{raw}_{\mathbf{p} \to \mathbf{q}}(d_i), min_{raw} + P2)$
if $i > 0$ **then**
  $m_i \leftarrow \min(m^{raw}_{\mathbf{p} \to \mathbf{q}}(d_{i-1}) + P1, m_i)$
end if
if $i < N_d - 1$ **then**
  $m_i \leftarrow \min(m^{raw}_{\mathbf{p} \to \mathbf{q}}(d_{i+1}) + P1, m_i)$
end if
$m_{\mathbf{p} \to \mathbf{q}}(d_i) \leftarrow m_i$

To speed up the convergence and to extract the depth in corresponding resolutions, the messages are updated in a coarse-to-fine manner. Different from standard belief propagation [6] which works on 2D cost volume with a regular shape, the proposed dynamic belief propagation works on the cost volume built from quadtree-selected pixels. The cost volume is valid only on quadtree-selected pixels and equals to the matching cost vector computed in Section IV-C. For other pixels that are not quadtree-selected, the cost volume is zero vector. The cost volume is downsampled as traditional methods [6] but averaged according to the

number of quadtree-selected pixels. Although the coarse-to-fine manner speeds up the convergence of messages, the computation at fine levels of the image is still heavy for real-time applications. Since the depth of a pixel can be represented in the corresponding resolution as illustrated in Fig. 2, the depth estimation can be extracted after the optimization of the corresponding level is over. The messages of extracted pixels are passed into finer levels as [6]. However, these messages are not updated anymore but only used to infer the depth of other pixels. The whole process of dynamic belief propagation that extracts the depth estimation of quadtree-selected pixels in one optimization is shown in Fig. 5.

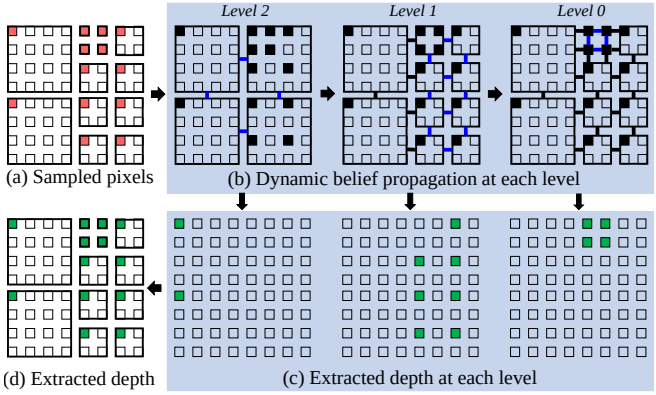


Fig. 5. Illustration of dynamic belief propagation that estimates the depth of quadtree-selected pixels. In (a), the image is converted into quadtree structure, quadtree-selected pixels are marked in red while the others are in white. Belief propagation works by updating messages between the image grid. As shown in (b), only messages (links colored in blue) of the corresponding or finer quadtree levels are updated using Algorithm 1 in parallel. Messages from coarse pixels (links colored in black) are terminated optimization. The depth estimations of quadtree-selected pixels are extracted at the corresponding levels of the optimization shown in (c) and combined as the output of dynamic belief propagation shown in (d).

E. Depth Interpolation and Temporal Fusion

Although the depth estimations of quadtree-selected pixels spread across the image, it is not full dense depth estimation. In this section, we first interpolate the estimations into full dense depth maps and then fuse them temporally with former results in a probabilistic way.

1) Depth Interpolation: During the interpolation, to further utilize the pixels that contain high-gradient texture but not quadtree-sampled, we estimate the depth of these pixels using the approach similar to Eigen *et al.* [16]. The high-gradient estimation is fully parallelized on GPU and takes about $2ms$ to update one frame.

The depth estimation from dynamic belief propagation and from high-gradient estimation are combined and interpolated spatially into full resolution depth map D_k corresponding to the incoming frame I_k. The depth estimated by the dynamic belief propagation and high-gradient estimation covers the whole image and draw more attention to texture-rich regions. Interpolating estimations to get full dense results is a common method to solve optical flow [17]. Here, we extract full dense depth maps by minimizing a least-square cost.

The cost of the interpolated full dense depth map D_k is defined as

$$E = \sum_{\mathbf{p} \in D_k} (h_{\mathbf{p}}(d'_{\mathbf{p}} - d_{\mathbf{p}})^2 + \lambda \sum_{\mathbf{q} \in \mathcal{N}(\mathbf{p})} w_{\mathbf{p},\mathbf{q}}(d'_{\mathbf{p}} - d'_{\mathbf{q}})^2), \quad (7)$$

where $d'_{\mathbf{p}}$ is the fused depth value of pixel $\mathbf{p}$, $d_{\mathbf{p}}$ is its corresponding depth estimation using belief propagation or high-gradient estimation if it exists. $h_{\mathbf{p}}$ is an index function, is 1 if $d_{\mathbf{p}}$ exists and is 0 otherwise. $w_{\mathbf{p},\mathbf{q}} = exp(-(\Delta I^2_{\mathbf{p},\mathbf{q}}/\sigma^2_{up})$ is the weight for depth difference based on intensity difference $\Delta I_{\mathbf{p},\mathbf{q}}$ of pixel $\mathbf{p}$ and $\mathbf{q}$ on image I_k. λ and σ^2_{up} control the smoothness of the interpolated depth map. Minimizing the cost E is computationally expensive. Here, we adopt the method from Min *et al.* [18] that approximates the problem in two 1D interpolate problems. Interpolating the depth in one dimension can be solved efficiently by Gaussian elimination. We first interpolate the depth in the row direction and then in the column direction.

2) Temporal Fusion: Full dense depth maps are further fused with former estimations in a probabilistic way to reject outlier estimations. The estimation of each pixel is modeled using an outlier robust model proposed by Vogiatzis *et al.* [19],

$$p(\hat{d}, \rho | a_k, b_k, \mu_k, \sigma^2_k) \approx \mathcal{N}(\hat{d} | \mu_k . \sigma^2_k) Beta(\rho | a_k, b_k), \quad (8)$$

where $\hat{d}$ is the ground truth depth, ρ is the inlier probability of the estimation. μ_k, σ^2_k are estimated depth and the corresponding variance, respectively, from frame I_k. a_k and b_k are the parameters modeling the inlier probability

$$p(\rho | a_k, b_k) \approx \frac{a_k}{a_k + b_k} \quad (9)$$

Different from REMODE [10] and Vogiatzis *et al.* [19] which estimate the depth maps of frames using the probabilistic model independent from each other, we maintain one probabilistic map throughout the runtime to fuse the sequential estimations. For every input image I_k, the probabilistic depth map of I_{k-1} is warped and fused with the extracted dense depth map. Depth estimations with inlier probability $p(\rho) > \rho_{inlier}$ are output as the result, while $p(\rho) < \rho_{outlier}$ are deleted as outliers.

F. Truncated Signed Distance Function (TSDF) Fusion

Filtered depth is fused into a global map using the method proposed by Klingensmith *et al.* [20]. Since our depth estimation contains very few outliers thanks to the dynamic belief propagation, interpolation, and temporal depth fusion, we fuse the output directly without any extra filters.

V. Implementation Details

Our method is fully parallelized with GPU acceleration. The efficiency makes the 3D reconstruction run in real-time even for portable devices, for example, the Nvidia Jetson TX1. The values of parameters listed below are found empirically and we find that they work well in all experiments. This can be explained that depth maps are estimated using global optimization which is robust to different image contents.

A. Quadtree-based Depth Estimation

The input image intensity is scaled between 0 and 1 to avoid potential numerical issues. During the calculation of the matching cost vector in Section IV-C, $N_d = 64$ depth values are calculated with $d_{min} = 0.5m$ and $d_{max} = 50.0m$. In Section IV-D, $P1$ and $P2$ are set to 0.003 and 0.01, respectively, to balance the smoothness and discontinuity of the estimated depth maps.

B. Depth interpolation and Temporal Fusion

Full dense depth maps are obtained using the least-square method. λ is set to 10 and σ_{up} is set to 0.07 in Section IV-E.1 so that interpolated depth maps are smooth, and discontinuity is preserved. In Section IV-E.2, the full dense depth maps are fused temporally with former dense maps in a probabilistic model initialized using $a = 10$ and $b = 10$. $\rho_{inlier} = 0.5$ and $\rho_{outlier} = 0.45$ are used during the fusion for fast convergence and outlier rejection.

VI. Experiments

We first show the depth estimation ability using the quadtree-selecting and the dynamic belief propagation which are two main contributions of our method on Middlebury Stereo Datasets [4]. Then, our system is compared with other open-source dense mapping methods: REMODE [10] and VI-MEAN [9] on public datasets including the TUM Dataset [21] and the ICL-NUIM Dataset [22]. The mapping part of VI-MEAN [9] is extracted following their successor work [23]. In the following sections, only the mapping part of VI-MEAN [9] is evaluated using provided camera poses. Since Multi-level mapping [15] is not open-source, our method is compared with the results reported in their paper to demonstrate the performance of depth estimation using global optimization. Finally, our system is tested using onboard UAV navigation experiments in complex environments to prove the usability and real-time performance.

A. Depth Estimation on Middlebury Stereo Dataset

Here, we show that our method can estimate disparity maps in high efficiency without losing much accuracy. We compare our method with the standard multi-level belief propagation [6] (BP) using 2003 Middlebury Stereo Dataset [4]. A quadtree of four levels is built to guide the quadtree-selecting and the dynamic belief propagation. The matching cost vector is calculated using SAD of 3×3 image patches. All the parameters are the same for both methods with regularize terms $P1 = 0.5$, $P2 = 4.0$ and 4-level optimization with iterations of $(10, 5, 5, 2)$ from coarse to fine. Two outputs of our method are evaluated: (1) depth estimation of quadtree-selected pixels using dynamic belief propagation of Section IV-D, (2) interpolated depth of quadtree-selected pixels using Section IV-E. We denote these outputs as Quadtree-pixels and Quadtree-fused, respectively. The extracted depth and error map are shown in Fig. 6. Table I shows the evaluation results where the "error" is the average absolute error of estimated disparities and the "messages" is the number of updated messages during the optimization which consumes most of the computation time. As shown in the results, by estimating the disparity maps according to the quadtree structure of the reference image, our method generates very competitive results using less than half the computation of standard belief propagation. The result of Quadtree-fused on Teddy is even better than standard method in accuracy because of its subpixel results.

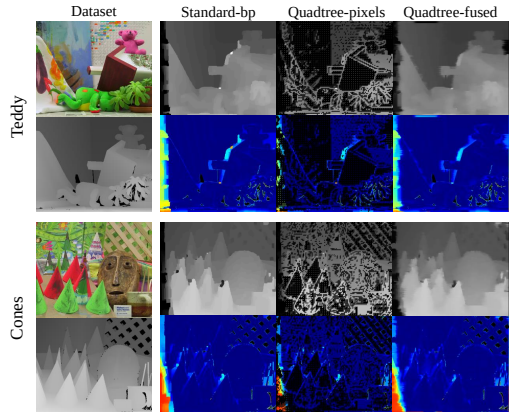


Fig. 6. Comparison between standard belief propagation and our method. Left column: images and ground truth depth maps from the dataset. For each output, the estimated depth and the corresponding error map are shown in top and bottom. The error map is coded in the Jet color map: blue means small errors and red means large errors. Note that the results of Quadtree-pixels are dilated for visualization. Evaluation results are shown in Table I.

TABLE I

Evaluation of Standard-BP and our method

		Standard-BP	Quadtree-pixels	Quadtree-filtered
Teddy	error	3.41	3.07	3.36
	messages	613K	283K	283K
Cones	error	3.28	2.64	3.29
	messages	613K	229K	229K

B. Comparison with State-of-the-art Methods

The TUM dataset [21] is designed to evaluate visual SLAM and RGB-D SLAM systems containing RGB-D images and camera poses captured by a high-accuracy motion-capture system. We select 6 sequences from the dataset suitable for monocular mapping: *nostructure texture far*, *long office household*, *sitting halfsphere*, *structure texture far*, *structure notexture far* and *sitting xyz*. The selected sequences contain a variety of environments from structure to no-structure, and from texture-rich to low-texture regions. The ICL-NUIM dataset [22] is generated synthetically with perfect ground truth poses and RGB-D data. Here, we use all the sequences in the living room scene.

All the parameters stay unchanged throughout the evaluation, and we set $d_{min} = 0.5$ and $d_{max} = 50$ for all the three methods. Other parameters are set to default values for VI-MEAN [9] and REMODE [10].

1) Quality Evaluation: Since both our method and VI-MEAN [9] have outlier detection, we treat diverged estimation reported by REMODE [10] as outliers and the others as valid estimations. Two standard measures are used to represent the quality of the method: density and relative error. The density of the depth map is measured by the percentage of the valid estimation to the whole image pixel number.

TABLE II

TUM DATASET AND ICL-NUIM DATASET RESULT

	nostructure texture far		long office household		sitting halfsphere		structure texture far		structure notexture far	
	density(%)	error(%)	density(%)	error(%)	density(%)	error(%)	density(%)	error(%)	density(%)	error(%)
REMODE [10]	49.46	93.19	41.47	73.50	27.69	26.37	50.30	19.74	37.06	38.44
VI-MEAN [9]	**73.92**	97.27	**83.10**	58.25	**75.81**	64.42	**90.55**	39.52	**84.49**	69.87
OURS	45.81	**4.79**	61.38	**7.97**	30.62	**13.00**	67.14	**5.12**	52.16	**10.23**
	sitting xyz		ICL-NUIM kt0		ICL-NUIM kt1		ICL-NUIM kt2		ICL-NUIM kt3	
	density(%)	error(%)	density(%)	error(%)	density(%)	error(%)	density(%)	error(%)	density(%)	error(%)
REMODE [10]	30.96	28.55	21.16	23.42	No Output	No Output	38.48	32.86	34.52	75.12
VI-MEAN [9]	**75.72**	59.13	**90.32**	35.89	**77.56**	76.96	**89.51**	52.08	**82.88**	91.54
OURS	33.79	**10.94**	43.52	**10.94**	17.66	**25.58**	33.85	**13.36**	33.20	**13.45**

TABLE III

COMPARISON WITH MULTI-LEVEL MAPPING [15] REPORTED RESULTS

	fr2/desk		fr3/long_office_household		fr3/nostructure_texture_near_withloop		fr3/structure_texture_far	
	density(%)	error(%)	density(%)	error(%)	density(%)	error(%)	density(%)	error(%)
Multi-Level Mapping [15]	26	17	23	20	**41**	6.2	**63**	**2.7**
Ours	**28**	**11**	**37**	**11**	32	**5.2**	49	5.3

TABLE IV

RUN-TIME PER FRAME(MS) (FPS) ON TX2

	640×480	512×384	320×240
REMODE [10]	53.0(18.9)	22.0(45)	11.6(86.2)
VI-MEAN [9]	209.5(4.8)	93.8(10.7)	52.1(19.2)
OURS	134.8(7.4)	81.6(12.3)	30.6(32.7)

Relative error is defined as

$$\frac{1}{|\mathcal{V}|}\sum_{\mathbf{i}\in\mathcal{V}}\frac{|d_{\mathbf{i}}-d_{\mathbf{i}}^{gt}|}{d_{\mathbf{i}}^{gt}}, \tag{10}$$

where $\mathcal{V}$ is the set of valid estimations, $d_{\mathbf{i}}$ and $d_{\mathbf{i}}^{gt}$ are estimated depth and corresponding ground truth, respectively.

The results are shown in Table II. Note that our method generates one depth map for every input frame while the other two only estimate the depth for some keyframes. By using the probabilistic model to detect outlier estimations, our method achieves the least relative error throughout all the sequences. The output density is lower than that of VI-MEAN [9] because the temporal fusion needs several frames to converge and the datasets contain lots of rotation movement that with small image overlaps. On the other hand, VI-MEAN [9] detects outliers by its matching cost thus cannot effectively detect all outliers leading to denser but inaccurate depth estimations. In robotic operations, where robots mostly move forward and image overlaps are large, our method generates depth maps with much higher densities as shown in Fig. 8 and the supplementary video. REMODE [10] estimates per-pixel depth values without global information before smoothing. In *ICL-NUIM kt0* sequence, the camera rotates rapidly and REMODE [10] cannot generate depth estimation because of limited per-pixel information.

2) Efficiency Evaluation: Efficiency on portable devices is important for mobile robots. We test the efficiency of dense mapping methods on a Nvidia Jetson TX2 with a 256-core GPU and 8 GB memory. Efficiency is measured by the average time to process one frame. All the parameters stay the same as in previous experiments and multi-resolution images are used to evaluate the scalability of each method.

Results are shown in Table IV. Note that our method generates dense depth estimation for every input image and achieves real-time performance on portable devices. Although our method uses global optimization, messages can be updated in parallel. The proposed quadtree based pixel selection and dynamic belief propagation enable our method to generate accurate depth maps in real-time. In SGM [11], although different cost aggregation paths can be calculated in parallel, pixels in the same path are updated sequentially, leading to a suboptimal usage of GPU parallel computing.

C. Comparison with Multi-level Mapping

Our method is compared with Multi-level mapping [15] algorithm since both methods use the quadtree to find the suitable resolution for depth estimation. We follow the definition of the error and the density as used in Multi-level mapping [15]. The error is defined as the relative inverse depth error and the density is the fraction of pixels with relative inverse errors less than 10%. The result is shown in Table III. For most of the cases, the accuracy of our method is higher than that of Multi-level mapping [15] because the depth of a pixel is estimated using global information. In *fr3/nostructure_texture_near_withloop* and *fr3/structure_texture_far*, the environments consist of rich texture and planar structure so that Multi-level mapping [15] performs well by using local patches.

D. Test on Onboard UAV

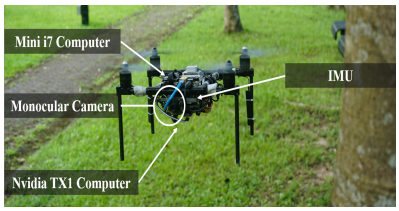


Fig. 7. The drone we use is equipped with the minimum sensor set of a monocular camera and an IMU that supports autonomous flights.

To evaluate the usability of our method on a UAV, we use the TSDF fused dense map to support autonomous flight. Fused voxels which contain the estimated surface are used for obstacle avoidance during the flight. Both the indoor and outdoor experiments are tested to prove the accuracy and

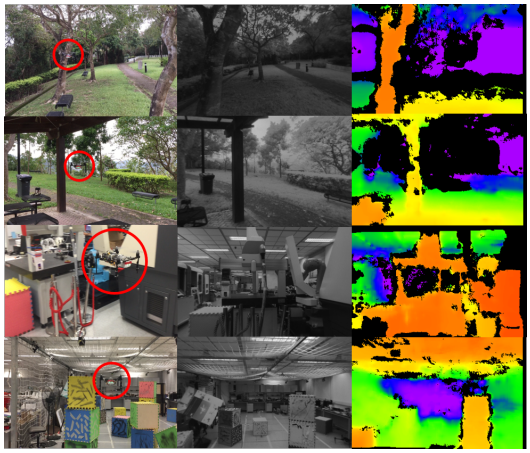


Fig. 8. We test our method onboard a UAV in a variety of environments from outdoor to indoor. The left column is the third view during the flight and the drone is highlighted in red circles. The middle and the right column are onboard cameras and the corresponding depth estimation. Depth maps are color coded as Fig. 3(d), with black pixels indicating invalid estimations masked by probabilistic depth fusion.

density of our dense mapping system. The UAV platform is shown in Fig. 7 that consists of an i7 computer and a Nvidia Jetson TX1. Our dense mapping system runs on TX1 and the other components including VINS [3] and planning run on the i7 computer. A camera and an IMU are all the sensors used for the autonomous flight. Input images are downsampled to 376×240 for real-time performance with a 10Hz output.

The indoor environment consists of repeated patterns like the safety net, and textureless surfaces like walls. Outdoor experiments are conducted in a small woods area where obstacles are thin and are hard to reconstruct. Snapshots during the flight are shown in Fig. 8 and more results can be found in the supplementary video. As can be seen in the results, our method generates depth maps that are dense and accurate for autonomous flight.

VII. CONCLUSION

In this paper, we propose a monocular dense mapping method that generates high-quality depth maps and dense 3D maps using images and corresponding camera poses. The output of the system can be used to reconstruct the environment in real-time and support autonomous navigation. Quadtree-selecting and dynamic belief propagation are proposed to speed up the mapping and generate accurate depth estimations. Both the public datasets and real-time onboard experiments show that our method generates depth maps in high quality and with high efficiency.

In the future work, we will focus on building large-scale, drift-free 3D dense maps by fusing depth maps with odometry systems that have loop closure abilities. Such a system is important for long-term robotic applications.

VIII. ACKNOWLEDGEMENT

This work was supported by the Hong Kong PhD Fellowship Scheme.

REFERENCES

[1] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *Int. J. Robot. Research*, 34(3):314–334, 2015.

[2] A. Concha, G. Loianno, V. Kumar, and J. Civera. Visual-inertial direct SLAM. In *Proc. of the IEEE Int. Conf. on Robot. and Autom.*, Stockholm, Sweden, May 2016.

[3] T. Qin, P. Li, and S. Shen. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. *arXiv preprint arXiv:1708.03852*, 2017.

[4] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2003.

[5] J. Sun, N Zheng, and H. Y. Shum. Stereo matching using belief propagation. *IEEE Transactions on pattern analysis and machine intelligence*, 25(7):787–800, 2003.

[6] P.F. Felzenszwalb and D.P. Huttenlocher. Efficient belief propagation for early vision. *International journal of computer vision*, 70(1):41–54, 2006.

[7] G. J. Sullivan and R. L. Baker. Efficient quadtree coding of images and video. *IEEE Trans. Image Process*, 3(3):327–331, May 1994.

[8] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *Proc. of the IEEE Int. Conf. Comput. Vis.*, Barcelona, Spain, November 2011.

[9] Z. Yang, F. Gao, and S. Shen. Real-time monocular dense mapping on aerial robots using visual-inertial fusion. In *Proc. of the IEEE Int. Conf. on Robot. and Autom.*, Singapore, May 2017.

[10] M. Pizzoli, C. Forster, and D. Scaramuzza. REMODE: Probabilistic, monocular dense reconstruction in real time. In *Proc. of the IEEE Int. Conf. on Robot. and Autom.*, Hong Kong, May 2014.

[11] H. Heiko. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2008.

[12] V. Pradeep, C. Rhemann, S. Izadi, C. Zach, M. Bleyer, and S. Bathiche. MonoFusion: Real-time 3D reconstruction of small scenes with a single web camera. In *Proc. of the IEEE and ACM Int. Sym. on Mixed and Augmented Reality*, Adelaide, SA, Australia, December 2013.

[13] P. Ondruska, P. Kohli, and S. Izadi. Mobilefusion: Real-time volumetric surface reconstruction and dense tracking on mobile phones. *IEEE transactions on visualization and computer graphics*, 21(11):1251–1258, 2015.

[14] T. Schops, T. Sattler, C. Häne, and M. Pollefeys. 3D modeling on the go: Interactive 3d reconstruction of large-scale scenes on mobile devices. In *Proc. of the Int. Conf. on 3D Vis.*, pages 291–299, ENS Lyon, France, October 2015.

[15] G.W. Nicholas, K. Ok, P. Lommel, and N. Roy. Multi-level mapping: Real-time dense monocular SLAM. In *Proc. of the IEEE Int. Conf. on Robot. and Autom.*, Stockholm, Sweden, May 2016.

[16] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *Proc. of the IEEE Int. Conf. Comput. Vis.*, Sydney, Australia, December 2013.

[17] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1164–1172. IEEE, 2015.

[18] D. Min, S. Choi, J. Lu, B. Ham, K. Sohn, and M. N. Do. Fast global image smoothing based on weighted least squares. *IEEE Transactions on Image Processing*, 23(12):5638–5653, 2014.

[19] G. Vogiatzis and C. Hernández. Video-based, real-time multi-view stereo. *Image and Vision Computing*, 29(7):434–441, 2011.

[20] 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.*, Rome, Italy, July 2015.

[21] 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 IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, Oct. 2012.

[22] A. Handa, T. Whelan, J. MacDonald, and A. J. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *Proc. of the IEEE Int. Conf. on Robot. and Autom.*, Hong Kong, May 2014.

[23] 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. *J. Field Robot.*, 00:1–29, 2017.