Adaptive Baseline Monocular Dense Mapping with Inter-frame Depth Propagation

Kaixuan Wang and Shaojie Shen

Abstract— **State-of-the-art monocular dense mapping methods usually divide the image sequence into several separate multi-view stereo problems thus have limited utilization of the information in multi-baseline observations and sequential depth estimations. In this paper, two core contributions are proposed to improve the mapping performance by exploiting the information. The first is an adaptive baseline matching cost computation that uses the sequential input images to provide each pixel with wide-baseline observations. The second is a frame-to-frame propagated depth filter which integrates the sequential depth estimation of the same physical point in a robust probabilistic manner. Two contributions are integrated into a monocular dense mapping system that generates the depth maps in real-time for both pinhole and fisheye cameras. Our system is fully parallelized and can run at more than 25 fps on a Nvidia Jetson TX2. We compare our work with state-of-the-art methods on the public dataset. Onboard UAV mapping and handhold experiments are also used to demonstrate the performance of our method. For the benefit of the community, we make the implementation open source[1].**

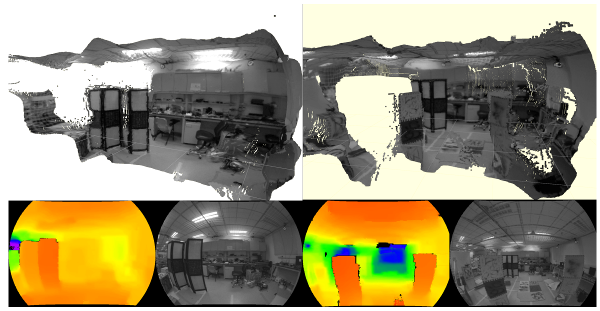
I. Introduction

Estimating the dense depth maps using cameras is important in robotic applications. Compared with widely used stereo systems, monocular dense mapping systems have multiple advantages. First, the size and power consumption of one camera are lower. Second, no extrinsic calibration between cameras is needed. On the contrary, the extrinsic relationship in stereo systems needs to be calibrated and maintained carefully. And lastly, the multi-baseline observations of a scene can be utilized for more robust and precise depth estimation. While short-baseline observations can remove the depth ambiguity caused by repeated patterns, large-baseline observations can improve the accuracy of the estimation [1]. These advantages of monocular dense mapping systems make them suitable for robots, especially for aerial robots, whose size, power and payload are limited.

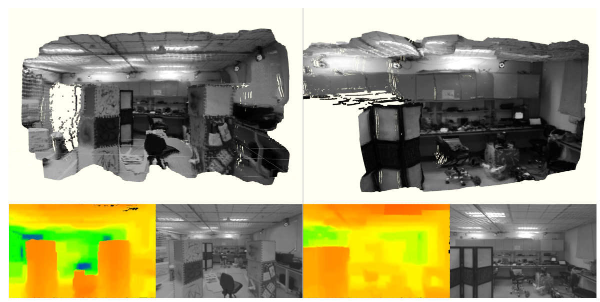
Recently, many methods (e.g., REMODE [2] and DTAM [3]) have been proposed to deal with the monocular dense mapping problem. These methods divide the input image sequence into several independent multi-view stereo problems. In these methods, the input image is compared with one or several selected images (a.k.a. measurement frames), and a depth estimation is generated if enough pixels are converged or the baseline is large enough. Although these

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

[1]https://github.com/HKUST-Aerial-Robotics/Pinhole-Fisheye-Mapping



(a) One Frame in UAV Fisheye Mapping



(b) One Frame in UAV Pinhole Mapping

(c) legend of colored depth map

Fig. 1. Fisheye and pinhole mapping on a UAV. Fig.1(a) and Fig.1(b) show mapping on a fisheye camera and pinhole camera in a lab environment. The input image and color-coded (using Fig.1(c)) depth image are shown on bottom. Corresponding point clouds are shown in different views. Our method can generate almost outlier free depth maps in real-time for every frame. Note that our method deals with depth discontinuity and high distortion well, for example, boxes, screen and celling regions.

methods generate dense depth maps, there are two problems that limit the utilization of the sequential information in the input images and estimated depth maps. First, there is a trade-off between the image overlap and the baseline length when selecting measurement frames. Wide-baseline measurement frames can improve the depth estimation, but only a small fraction of pixels can be estimated due to small overlaps. On the other hand, the depth cannot be accurately computed with short-baseline measurement frames although there exist large overlaps. Second, the sequential depth maps of the same scene are not fused to refine estimations and reject outliers. Depth consistency is used in some methods (e.g., [4]) to detect outliers. However, not all the previous depth estimations are exploited to detect outliers, and the depth maps are not refined in these methods.

In this paper, an adaptive baseline matching cost computation and an inter-frame propagated depth filter are proposed to address the above problems. An "age-map" is maintained in our system to record the number of previous frames each pixel is observed in. During the matching cost computation, the measurement frames are selected specifically for each pixel according to the pixel "age". Wide-baseline observations of pixels are utilized, and the overlap is ensured. A probabilistic depth filter is propagated from frame to frame to fuse the current depth estimation with previous estimations. The depth estimation is refined, and outliers are detected during the fusion. We integrate the proposed matching cost computation method and the depth filter into a monocular dense mapping system which can generate high-quality dense depth maps.

Benefitting from the proposed adaptive baseline matching cost computation and the propagated depth filter, our system outperforms state-of-the-art monocular mapping methods regarding accuracy on the public dataset [5] and achieves real-time performance on portable devices. More onboard UAV mapping (Fig. 1) and handhold mapping (Fig. 8) experiments are used to demonstrate the performance of our method.

The contributions of this paper are the following:

- an adaptive baseline matching cost computation that utilizes the information in sequential input images. The matching cost of each pixel is computed using available wide-baseline observations in the previous input images while the image overlap is ensured. The common dilemma between the large-baseline measurement frames and the image overlaps is solved by the proposed cost computation method.
- an inter-frame propagated depth filter that fuses the sequential depth estimations. Depth maps are fused in a robust probabilistic way in which estimations are refined, and outliers are detected.
- an open source monocular dense mapping system that generates depth maps using the sequential information of the input images by the proposed adaptive baseline cost computation and the propagated depth filter. The system supports both pinhole and fisheye cameras and runs in real-time even on portable devices, such as Nvidia TX2.

II. Related Work

Many methods have been proposed to solve the real-time monocular semi-dense and dense mapping problem.

LSD-SLAM [6] and Multi-level Mapping [7] are proposed to estimate the camera motion and build semi-dense maps. The estimated semi-dense maps cover areas with rich textures but are not sufficient for robot navigation.

DTAM [3] and VI-MEAN [8] are two methods that estimate the depth map by minimizing an energy function. Multiple images are selected as measurement frames to construct the cost volume for a robust and accurate estimation. However, the baseline of the measurement frames is limited to ensure enough overlaps. DTAM [3] uses a total variation to optimize the energy and requires a desktop graphics processing unit (GPU) due to the expensive computation. VI-MEAN [8] adopts the method of semi-global matching (SGM) [9] to generate the depth map. Because of the 4-path SGM, VI-MEAN [8] suffers from the "streaking" artifacts in the estimation.

3D Modeling on the Go [4] regard the reconstruction as a dense two-view stereo matching problem. The input image is compared with a selected frame to estimate the depth. The two-view stereo matching is efficient but generates noisy depth maps due to the depth ambiguity in low-texture regions and occlusions. [4] uses an extended Kalman filter to integrate the depth estimations over time. Although the sequential depth estimation is fused, the output still contains outliers and need a variety of heuristically designed filters (e.g., consistency over time) to detect.

REMODE [2] models estimated depth as a probabilistic distribution and carries out a $L1$-norm total variation smooth before the output. The probabilistic model of each pixel is robust to outlier update and can give out the certainty of each estimation. CNN-SLAM [10] combines LSD-SLAM [6] with predicted depth using deep learning methods. The combination enables CNN-SLAM to generate dense maps and robust odometry estimation. However, the application domain is limited to the training set of depth prediction networks.

Several aspects distinguish our algorithm from all the methods mentioned above. Firstly, our method utilizes the sequential information in both the matching cost computation and the depth refinement. Benefitting from the adaptive baseline matching, the baseline of the measurement frames for each pixel is not limited as long as the pixel is visible. The propagated depth filter fuses sequential estimations and detects outliers in a probabilistic way. Secondly, our method uses a parallelized belief propagation to extract the depth map from the computed matching cost. The 2D global optimization generates smooth and accurate depth estimations without the "streaking" artifacts. Lastly, our method is applied to both pinhole and fisheye cameras.

III. System Overview

Like the classic stereo matching pipeline, our monocular dense mapping system consists of three modules: adaptive baseline matching cost computation (Section IV-B), belief propagation-based depth extraction (Section IV-C), and depth refinement using inter-frame propagated depth filter (Section IV-D).

The pipeline of the whole system is shown in Fig. 2. For each input image, the "age-map" and the depth filter will propagate into the input frame. The matching cost is calculated for pixels using the specifically selected measurement frames based on the "age-map". The depth extraction module extracts the depth of pixels on regular grids from the cost

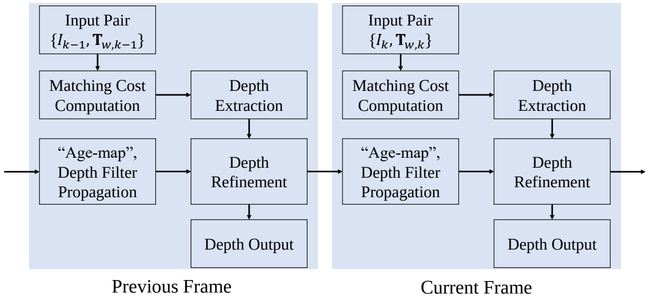


Fig. 2. An overview of our system. The "age-map" and the probabilistic depth filter are propagated from frame to frame maintaining the sequential information of the input images and the depth estimations.

volume using sped up belief propagation and interpolate it into full dense depth maps. Finally, the depth filter fuses the extracted depth with former depth estimations in a probabilistic way. Inlier pixels are refined, and outlier pixels are deleted.

IV. Monocular Dense Reconstruction

A. Preliminaries

Let the transformation $\mathbf{T}_{w,k} \in \mathbb{SE}(3)$ be the pose of the camera frame with respect to the world frame w when taking the k-th image. 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 pixel $\mathbf{u} := (u, v)^{\mathrm{T}} \in \Omega$ using the camera projection function: $\pi(\mathbf{x}_c) = \mathbf{u}$. Also, a pixel can be back-projected as a 3D point: $\mathbf{x}_c = \pi^{-1}(\mathbf{u}, d)$ where d is the depth of the pixel $\mathbf{u}$. The projection function $\pi(\cdot)$ and the back-project function $\pi^{-1}(\cdot)$ depends on the camera model applied in the system.

An "age-map" is maintained in our system recording the frame number a pixel can be tracked backwards the image sequence. Denote the "age-map" of I_k as $A_k : \Omega \subset \mathbb{R}^2 \mapsto \mathbb{N}$. $A_k(\mathbf{u})$ means that that pixel $\mathbf{u}$ is visible in every frame from $I_{k-A_k(\mathbf{u})}$ to I_k.

The input of our monocular dense mapping is a sequence of $\{I_k, \mathbf{T}_{w,k}\}$ pair and the output is a sequence of the depth estimation corresponding to I_k. We assume that the metric camera poses are obtained by the monocular visual-inertial system (e.g., VINS-MONO [11]). The bundle optimization of inertial measurements and visual observations makes the camera pose estimation accurate with the metric scale for monocular dense mapping.

B. Adaptive Baseline Matching Cost Computation

Different from widely used stereo mapping setup, monocular dense mapping systems benefit from multi-baseline observations of each pixel.

To maximally utilize the large-baseline observation for each pixel instead of simply accumulating input frames like DTAM [3], we designed a novel matching cost computation method that compares each pixel with former input images according to its visibility in the image sequence. For pixel $\mathbf{u}$ in frame I_k, the matching cost at depth d is defined as

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

where $M(\mathbf{u})$ is the measurement frame list selected for pixel $\mathbf{u}$. For each selected frame I_m and depth d, the pixel $\mathbf{u}$ in frame I_k is projected into frame I_m: $\mathbf{u}' = \pi(\mathbf{T}_{m,w}\mathbf{T}_{k,w}^{-1}\pi^{-1}(\mathbf{u}, d))$. $SAD(I_m, I_k, \mathbf{u}, d)$ stands for sum of absolute difference. It simply calculates the similarity of a 3×3 patch centered at $\mathbf{u}'$ and $\mathbf{u}$ in image I_m and I_k.

The depth is evenly sampled on the inverse depth space to construct the matching cost volume. Let d_{max} and d_{min} denote the maximum and the minimum distance sampled, respectively. And N_d is the total sampled depth number. Then the sampled depth d and the corresponding index l has relationship

$$\frac{1}{d} = (\frac{1}{d_{min}} - \frac{1}{d_{max}})\frac{l}{N_d - 1} + \frac{1}{d_{max}}, \quad (2)$$

where $l \in [0, N_d - 1], l \in \mathbb{N}$.

One of the core contributions that distinguishes our method from other monocular dense mapping methods is the frame select function $M(\mathbf{u})$. In our method, $M(\mathbf{u})$ evenly samples 10 frames from the oldest frame to the latest frame in which the pixel $\mathbf{u}$ is observed. For pixels that are observed in less than 10 frames, $M(\mathbf{u})$ contains the latest 10 frames for a robust estimation. Given the "age-map" A_k, 10 measurement frames are selected for pixel $\mathbf{u}$ as

$$M(\mathbf{u}) = \{I_{\lfloor k - A_k(\mathbf{u}) \times i/10 \rfloor} \mid i \in \{1, 2 \cdots 10\}\} \quad (3)$$

Since the pixel $\mathbf{u}$ is observed in every measurement frame $I_m \in M(\mathbf{u})$, the matching cost can always be computed. And $I_{k-A_k(\mathbf{u})}$, $I_{\lfloor k - A_k(\mathbf{u})/10 \rfloor}$ is the longest and shortest baseline observation respectively for cost computation if the camera moves in one direction.

Fig. 3 shows how the "age-map" propagates and helps the measurement frame selection. The maintenance of the "age-map" will be discussed in Section IV-D.4.

C. Belief Propagation-based Depth Extraction

In this section, a depth map is extracted from the cost volume built in Section IV-B. While texture areas may be estimated by the winner-takes-all strategy, textureless areas cannot be determined solely by the matching cost. To handle regions with low-texture or even no-texture, we adopt a belief propagation-based depth extraction to generate high-quality smooth depth maps. The belief propagation is parallel accelerated and extracts the depth of each pixel considering both the matching cost and depth map smoothness.

To further speed up the depth extraction, we extend a common interpolation idea from optical flow [12], [13] into the depth estimation. The depth values on regular grids with a step of 4 pixels are solved using the accelerated belief propagation. Then, the depth interpolation is performed to obtain the full resolution depth map. In the following sections, we show how these two steps help us extract depth maps.

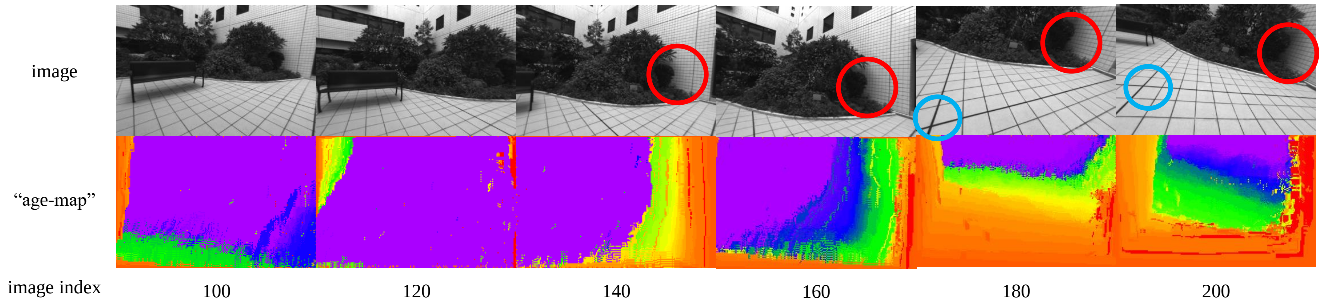


Fig. 3. An example to show the “age-map” propagates during the mapping process. The first row is the input images and the second row is the corresponding “age-maps” with rainbow color coded. Purple means the age is more than 60 frames and red is less than 10 frames. The last row is the index of images in the sequence. As shown in the figure, objects that last in the camera view have more “ages”. For example, the bushes in the red circle are seen in the images for about 60 frames, so large-baseline frames from I_{140} to I_{200} can be used to compute the matching cost of pixels in the red circle. On the other hand, the floor tiles in the blue circle only last for 20 frames, so the matching cost is computed with images from I_{180} to I_{200}. By maintaining the “age-map”, the baseline of the measurement frames selected for each pixel is adaptive to the visibility of the pixel in the input sequence.

1) Parallelized Belief Propagation: Belief propagation [14], [15] works by passing messages along four connected image grids carrying smooth information for neighbor pixels. Messages are vectors of the dimension given by the number of depth samples N_d. All messages are initialized as zero vectors and updated iteratively for each pixel. Let $m_{\mathbf{u}\to\mathbf{v}}$ be the message from pixel $\mathbf{u}$ to pixel $\mathbf{v}$, at each time, it is calculated as

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

$$m_{\mathbf{u}\to\mathbf{v}}(d_{\mathbf{v}}) = \min_{d_{\mathbf{u}}}(V(d_{\mathbf{u}}, d_{\mathbf{v}}) + m^{raw}_{\mathbf{u}\to\mathbf{v}}(d_{\mathbf{u}})), \tag{5}$$

where $\mathcal{N}(\mathbf{u})$ stands for the 4-neighbors of pixel $\mathbf{u}$. $V(d_{\mathbf{u}}, d_{\mathbf{v}})$ is the regularize function controlling the smoothness of the estimated depth maps. After enough iterations, the final belief cost $b_{\mathbf{u}}$ of pixel $\mathbf{u}$ at depth $d_{\mathbf{u}}$ combining both the matching cost and the neighbor messages is

$$b_{\mathbf{u}}(d_{\mathbf{u}}) = C(\mathbf{u}, d_{\mathbf{u}}) + \sum_{\mathbf{s}\in\mathcal{N}(\mathbf{u})} (m_{\mathbf{s}\to\mathbf{u}}(d_{\mathbf{u}})). \tag{6}$$

The depth with the smallest belief cost is assigned to the pixel as the estimation.

The standard message update [15] uses a forward and backward scan to compute the message in $O(N_d)$ time. Here we further reduce the update time to $O(log(N_d))$. Inspired by the success of SGM [9], which uses only two parameters to model the smoothness of the image, we define the function $V(d_{\mathbf{u}}, d_{\mathbf{v}})$ as

$$V(d_{\mathbf{u}}, d_{\mathbf{v}}) = \begin{cases} 0 & if\ |l_{\mathbf{u}} - l_{\mathbf{v}}| = 0 \\ P1 & if\ |l_{\mathbf{u}} - l_{\mathbf{v}}| = 1 \\ P2 & if\ |l_{\mathbf{u}} - l_{\mathbf{v}}| \geq 1 \end{cases}, \tag{7}$$

where $l_{\mathbf{u}}$ is the index of $d_{\mathbf{u}}$ as defined in Equation 2. Although the regularize function is the same with that in SGM [9], our method works in a 2D optimization way that the information of a pixel passes to the whole image along the image grids instead of some predefined 1D path. Thanks to the 2D optimization, the extracted depth maps are smooth and do not have the “streaking” artifacts as in VI-MEAN [8].

With the regularize function defined in Equation 7, the message can be calculated parallelly using a reduce-min operation on the GPU achieving $O(log(N_d))$ time. Pseudocode of updating message $m_{\mathbf{u}\to\mathbf{v}}$ is shown in Algorithm 1.

Algorithm 1 Accelerated Message Update with N_d threads

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

compute $m^{raw}_{\mathbf{u}\to\mathbf{v}}(d_i)$ using Equation 4
$m_{min}(d_i) \leftarrow m^{raw}_{\mathbf{u}\to\mathbf{v}}(d_i)$ {reduce-min begins}
$step \leftarrow \lfloor N_d/2 \rfloor$
$synchronize\ threads$
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
$min_{raw} \leftarrow m_{min}(0)$ {reduce-min ends}
$m_i \leftarrow \min(m^{raw}_{\mathbf{u}\to\mathbf{v}}(d_i), min_{raw} + P2)$
if $i > 0$ **then**
  $m_i \leftarrow \min(m^{raw}_{\mathbf{u}\to\mathbf{v}}(d_{i-1}) + P1, m_i)$
end if
if $i < N_d - 1$ **then**
  $m_i \leftarrow \min(m^{raw}_{\mathbf{u}\to\mathbf{v}}(d_{i+1}) + P1, m_i)$
end if
$m_{\mathbf{u}\to\mathbf{v}}(d_i) \leftarrow m_i$

2) Depth Interpolation: Due to the smoothness of the depth map, we assume that the depth of an unoptimized pixel can be represented as a linear combination of surrounding optimized depths. The depth for an unoptimized pixel $\mathbf{p}$ can be represented using surrounding optimized pixels $\mathbf{q}$ as

$$d_{\mathbf{p}} = \frac{1}{W} \sum_{\mathbf{q}} W_{\mathbf{p},\mathbf{q}} d_{\mathbf{q}}. \tag{8}$$

And the weight is defined as

$$W_{\mathbf{p},\mathbf{q}} = exp(-\left\|\mathbf{p}-\mathbf{q}\right\|_2^2/\sigma_s^2 - \left\|I_{\mathbf{p}} - I_{\mathbf{q}}\right\|_2^2/\sigma_i^2), \quad (9)$$

depending on the spatial distance and the intensity similarity between pixel $\mathbf{p}$ and pixel $\mathbf{q}$. $W = \sum W_{\mathbf{p},\mathbf{q}}$ is the normalize factor. σ_s controls smoothness of the depth image, and σ_i allows discontinuity when the intensity changes.

Although more information can be used to extract the depth of unoptimized pixels, for example, combining the cost volume and the prior from surrounding optimized pixels $d_{\mathbf{p}} = \underset{d}{\operatorname{argmin}}(\sum W_{\mathbf{p},\mathbf{q}}(d_{\mathbf{q}} - d)^2/W + C(\mathbf{p}, d))$, we found that a simple combination of the optimized depth is good enough for further processing and gain much more efficiency.

D. Depth Refinement using Inter-frame Propagated Depth Filter

Rather than design heuristic filters as in [4] to detect outliers, we develop the robust depth model proposed in [16] into a depth filter. The depth filter propagates as the camera moves and fuses all the sequential extracted depth (Section IV-C) to estimate refined depth values and inlier probability. The result of the depth filter is also used to update the "age-map" which is used for adaptive baseline matching cost computation in Section IV-B.

Several aspects distinguish our method from [16] and REMODE [2] which also use the depth filter. First, in our method, only one depth filter is used to fuse all the extracted depth maps. During the fusion, the depth estimations are refined, and outliers are detected. On the other hand, [16] and REMODE [2] use multiple depth filters bound on selected keyframes to estimate the corresponding depth maps. Second, our filter is updated using high-quality depth maps extracted in Section IV-C. However, the other two methods use the depth estimation from local patch comparison which is noisy and cannot handle low-texture regions.

1) Depth Filter Model: Similar to [16], we model each pixel's depth estimation as a Gaussian distribution plus a uniform distribution

$$p(d_k|\hat{d},\rho) = \rho\mathcal{N}(d_k|\hat{d},\tau_k^2) + (1-\rho)\mathcal{U}(d_k|d_{min}, d_{max}), \quad (10)$$

where $\hat{d}$ is the ground truth depth, τ_k and ρ models the standard variance of inlier estimations and the probabilistic of outlier estimations, respectively. Given a sequence of independent observations $d_1, \ldots, d_k$, the posterior is

$$p(\hat{d},\rho|d_1,\ldots,d_k) \propto p(\hat{d},\rho)\prod_k p(d_k|\hat{d},\rho). \quad (11)$$

[16] shows that the posterior can be approximated by the product of a Gaussian distribution and a Beta distribution by matching the first and second moments of $\hat{d}$ and ρ

$$p(\hat{d},\rho|a_k,b_k,\mu_k,\sigma_k^2) \approx \mathcal{N}(\hat{d}|\mu_k.\sigma_k^2)Beta(\rho|a_k,b_k) \quad (12)$$

where μ_k and σ_k^2 is the estimated depth and the corresponding variance. a_k and b_k are parameters modeling the inlier ratio

$$p(\rho|a_k,b_k) \approx \frac{a_k}{a_k+b_k} \quad (13)$$

2) Depth Filter Propagation: The depth filter in our method is propagated from frame to frame estimating the current depth map. For each input frame $\{I_k, \mathbf{T}_{w,k}\}$, the depth filter corresponding to $\{I_{k-1}, \mathbf{T}_{w,k-1}\}$ is first propagated to the current frame.

Depth filter element at pixel $\mathbf{u}$ on image I_{k-1} with estimation parameter $(\mu_{k-1}, \sigma_{k-1}^2, a_{k-1}, b_{k-1})$ is projected at the new frame:

$$\begin{cases} \mathbf{u}' = \pi(\mathbf{T}_{w,k}^{-1}\mathbf{T}_{w,k-1}\pi^{-1}(\mathbf{u},\mu_{k-1})) \\ \mu' = \left\|\mathbf{T}_{w,k}^{-1}\mathbf{T}_{w,k-1}\pi^{-1}(\mathbf{u},\mu_{k-1})\right\|_2 \\ \sigma'^2 = \sigma_{k-1}^2 + \sigma_{prop}^2 \\ a' = a_{k-1} \\ b' = b_{k-1} \end{cases}, \quad (14)$$

where $\mathbf{u}'$ is the new position on I_k. μ', σ'^2 are the predicted depth and variance for pixel $\mathbf{u}'$ and σ_{prop}^2 comes from the prediction uncertainty of the depth filter propagation.

During the depth filter propagation, collision is handled that if more than one filter elements are projected to one pixel, only the element with the minimum predicted depth value μ' are kept, and others are removed as occlusion. After the propagation, elements are dilated one pixel to fill the holes caused by the forward-warping.

3) Depth Filter Update: After the filter propagation, the depth prediction is aligned with the current input frame. The extracted depth of the current frame using the method discussed in Section IV-C is fused with the corresponding depth prediction. For a pixel $\mathbf{u}$ with extracted depth $d_{\mathbf{u}}$ that do not have a depth prediction, a new filter element is initialized using normal distribution with mean $d_{\mathbf{u}}$ and variance σ_{init}^2 times the Beta distribution with parameters a_{init} and b_{init} as Equation 12. If the correspondent depth filter exists, the depth filter is updated by matching the first and second moments in a close form [16].

Outlier elements are unavoidable in dense mapping systems that they will be projected into wrong positions and fused with other pixels. The estimated inlier ratio is used to handle inlier and outlier elements in the depth filter. Depth element with estimated inlier ratio $p(\rho) > \rho_{inlier}$ are output as the depth estimation for this pixel. The depth values from elements with $p(\rho) < \rho_{inlier}$ are considered as unreliable estimations and are masked in the depth maps. Elements with inlier ratio $p(\rho) < \rho_{outlier}$ are deleted as outliers and will be reinitialized in the fusion of next frame. Because of the outlier robust model used in our system, the depth map is robust to temporary outliers. As shown in Fig. 4, some outliers in the extracted depth are corrected and others are masked by the proposed depth refinement.

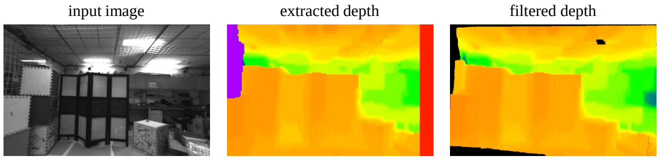


Fig. 4. Example to show refining a depth map using the probabilistic depth filter. The left image is the current input frame of the system. The middle is the extracted depth (Section IV-C). The right image is the result after the depth filter. Depth maps are color coded as Fig.1. The depth filter is robust to temporary outliers (purple and red estimation in the extracted depth) and gives out a better final depth map with outliers masked.

4) "Age-Map" Propagate and Update: The "age-map" tracks pixels from frame to frame enabling our method to exploit large-baseline observations during the adaptive baseline matching cost computation without sacrificing the image overlaps. The "age" of each pixel is initialized as zero. For every input image, the "age" of pixel $\mathbf{u}$ on I_{k-1} is propagated to $\mathbf{u}'$ on I_k using Equation 14 and increased by one. During the update of the depth filter, an outlier element deletion reset the corresponding pixel age to zero. Due to the memory limitation in GPU, the latest 60 frames are stored in our system and each pixel has the maximum "age" of 60.

V. Implementation Details

Our monocular depth reconstruction is fully parallelized using CUDA[2]. The efficiency makes the dense mapping run in real-time. Our method is also applied to fisheye cameras estimating the depth maps for highly distorted images.

A. Matching Cost Computation

To avoid potential numerical issues, input images are scaled between 0 and 1 before any calculation. $N_d = 64$ depth values are used to compute the matching cost and $d_{min} = 0.5$, $d_{max} = 50.0$ are used to determine the range of the depth values in Section IV-B. For fisheye cameras, the polynomial camera model [17] is used for projection and back-projection function.

B. Depth Extraction

In Section IV-C, $P1 = 0.2$ and $P2 = 2.0$ is used to control the smoothness of the depth maps. During the depth interpolation, each pixel is a combination of 25 optimized neighbor depths, with spatial weight $\sigma_s = 1.0$ and intensity weight $\sigma_i = 0.5$.

C. Depth Refinement

In Section IV-D, elements in the depth filter are initialized using $\sigma_{init}^2 = d_{max}^2$, $a_{init} = 10$, and $b_{init} = 10$. We assume each pixel with the extracted depth d using the method discussed in Section IV-C has standard deviation $sigma = 0.1 \times d$ when updating the filter element. During the propagation, $\sigma_{prop} = 0.05$ is used for prediction uncertainty. Inlier, outlier ratio threshold is set $\rho_{inlier} = 0.6$, $\rho_{outlier} = 0.4$.

[2]www.nvidia.com

VI. Experiments

We first compare our pinhole implementation with two state-of-the-art open source works REMODE [2] and VI-MEAN [8]. The mapping part of VI-MEAN [8] is separated experiments according to their further work [18]. All three methods need the distance prior of the environment. We set $d_{max} = 50.0$ and $d_{min} = 0.5$ all the methods. Parameters stay the same throughout the experiments. Only valid depth estimations are measured in the experiments. Both our method and VI-MEAN [8] have invalid detection. For REMODE [2], diverged estimations are masked as invalid points for a fair comparison.

A. Quality Evaluation

The TUM dataset [5] contains real-world RGB-D data and ground-truth camera poses for visual SLAM systems. The RGB-D data was recorded at 30 Hz with a resolution of 640×480. The camera pose was recorded at 100 Hz using a high-accuracy motion-capture. Here we select 6 sequences suitable for monocular mapping: *freiburg3 nostructure texture far*, *freiburg3 long office household*, *freiburg3 sitting halfsphere*, *freiburg3 structure texture far*, *freiburg3 structure notexture far*, and *freiburg3 sitting xyz*. Unlike synthetic datasets, the TUM dataset contains a variety of environments and movement patterns. All the methods are running on a Nvidia Jetson TX2 which is a portable device widely used in robotic applications. The TX2 is integrated with a 256-core GPU, a hex-core CPU, and 8 GB memory.

Three standard measures are used to evaluate the quality of each method. We define the relative error (RE) of a depth estimation as

$$RE(d_{\mathbf{u}}) = \frac{|d_{\mathbf{u}} - d_{\mathbf{u}}^{gt}|}{d_{\mathbf{u}}^{gt}}, \tag{15}$$

where $d_{\mathbf{u}}^{gt}$ is the ground truth corresponding to estimation $d_{\mathbf{u}}$. The first measure we use is mean relative error (MRE), which is the mean of all the relative errors of the valid estimations. The second measure is density, which is the density of valid estimation in the depth map. The first and second measures represent the quality and density of the depth maps. Lastly, we use RE-density to measure the relative error distribution of the estimations generated by each method. $RE\text{-}density(e)$ is defined as the percentage of valid estimations whose relative errors are within e.

In addition, we also evaluate the importance of the proposed matching cost computation and the depth refinement in our system. To evaluate the benefit of utilizing multi-baseline observations in the sequential input, we measure the performance of our method without the help of adaptive baseline matching cost computation in No-adaptive. In No-adaptive, for all the pixels in I_k, I_{k-3} is selected as the measurement frame for a balance between image overlap and baseline length. The depth refinement is evaluated in No-refinement, that the extracted full dense depth maps are evaluated directly without the depth filter fusion.

TABLE I
TUM DATASET [5] RESULT

	density(%)				MRE(%)				
dataset	REMODE	VI-MEAN	OURS	No-adaptive	REMODE	VI-MEAN	OURS	No-adaptive	No-refinement
nostructure texture far	49.46	**73.92**	64.24	41.85	93.19	97.27	**13.49**	14.53	292.98
long office household	41.47	**83.10**	77.86	69.57	73.50	58.25	**15.67**	17.28	160.05
sitting halfsphere	27.69	**75.81**	53.33	41.80	26.37	64.42	**23.10**	28.45	184.47
structure texture far	50.30	**90.55**	86.67	71.14	19.74	39.52	**12.01**	18.46	37.28
structure notexture far	37.06	**84.49**	84.13	58.70	38.44	69.87	**20.68**	24.23	192.79
sitting xyz	30.96	**75.72**	59.47	31.46	28.55	59.13	**24.94**	27.92	175.63

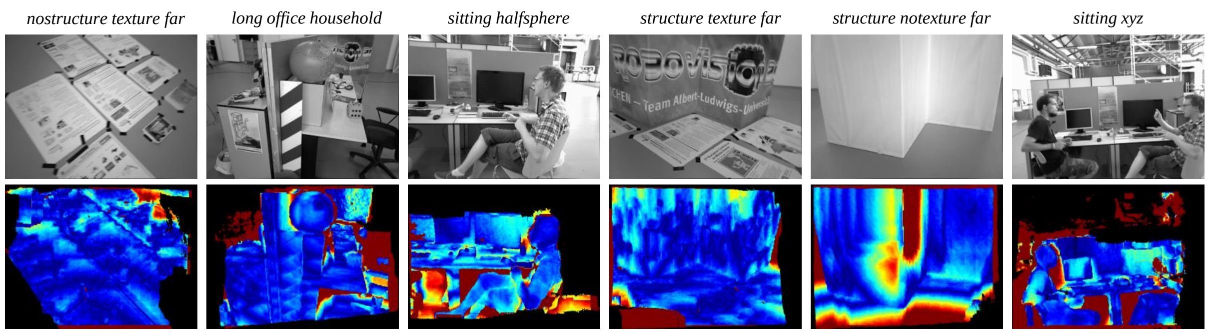


Fig. 5. Relative Error Map of our method in six scenarios from the TUM dataset. The first row is a snap of correspondent sequence. The following row is the error map of the depth image. Error maps are color coded in the Jet colormap: cold colors mean small error while warm colors mean large error. Black indicates outlier pixels masked by the depth filter. Note that our method generates smooth depth maps without the "streaking" artifacts and can handle textureless region well (for example, in *structure notexture far*). The probabilistic filter can detect outliers (e.g. caused by moving objects as in sequence *sitting xyz*) and keep inlier depth estimations.

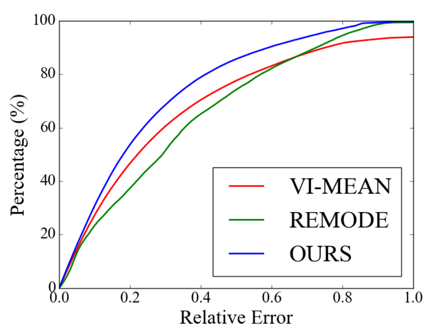


Fig. 6. RE-density of the three methods in six TUM dataset sequences. The estimation in our method is much more precise.

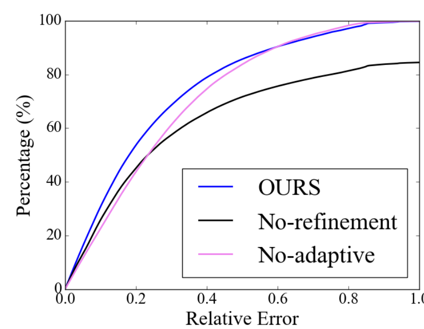


Fig. 7. RE-density to evaluate the importance of the proposed adaptive baseline matching cost computation and the propagated depth filter-based depth refinement.

The results are shown in Table I, Fig. 5, Fig. 6, and Fig. 7 Our method outperforms the compared methods in MRE by a large margin in all the six sequences. The density of the estimated depth map is higher than that of REMODE [2] and close to that of VI-MEAN [8] in many of the sequences. Fig. 5 show that our method can deal with textureless regions and detect outlier estimations effectively. In Fig. 7, the error distributions of No-refinement and No-adaptive are compared with that of our method which utilizes the sequential information in both the input images and extracted depth maps. As shown in Table I and Fig. 7, the depth estimation is more dense and accurate when multi-baseline information in the input sequence is used. By fusing all the sequential depth maps, the outliers in No-refinement are removed and the overall accuracy is improved dramatically.

B. Efficiency Evaluation

The efficiency of each method is measured by the time to process one input image. Multi-resolution is used to evaluate the efficiency on different scales. As shown in Table II, our method achieves real-time performance on TX2 with the resolution of 512×384. Compared with REMODE [2] and VI-MEAN [8] which only estimate depth for some keyframes, our method generates depth estimation for every input frame in the processing time. The low-latency depth estimation makes our system suitable for robotic applications where fast perception is required for safety.

C. Onboard and Live Experiment

We test our method using pinhole and fisheye cameras in a lab environment on a UAV. Images are processed onboard in 376×240. camera poses are estimated using visual-inertial

TABLE II

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

	640×480	512×384	320×240
REMODE [2]	53.0(18.9)	22.0(45)	11.6(86.2)
VI-MEAN [8]	209.5(4.8)	93.8(10.7)	52.1(19.2)
OURS	127.6(7.8)	80.2(12.5)	33.4(29.9)

system (VINS) [11] at 10 Hz. All the parameters stay the same as in previous experiments. In fisheye mapping, we use a 180-degree field of view fisheye camera. Results are shown in Fig. 1. Our method generates nearly outlier free depth images by fusing the sequential information. Note that in the fisheye test, the ceiling is reconstructed right even it is highly distorted in the image and depth discontinuity is preserved between the screen and background.

More outdoor mapping experiments are tested using images with the resolution of 752×480. Both pinhole and fisheye cameras are used. As shown in Fig. 8, our method generates dense and smooth depth maps capturing fine structures (for example, trees, and poles) in the environments.

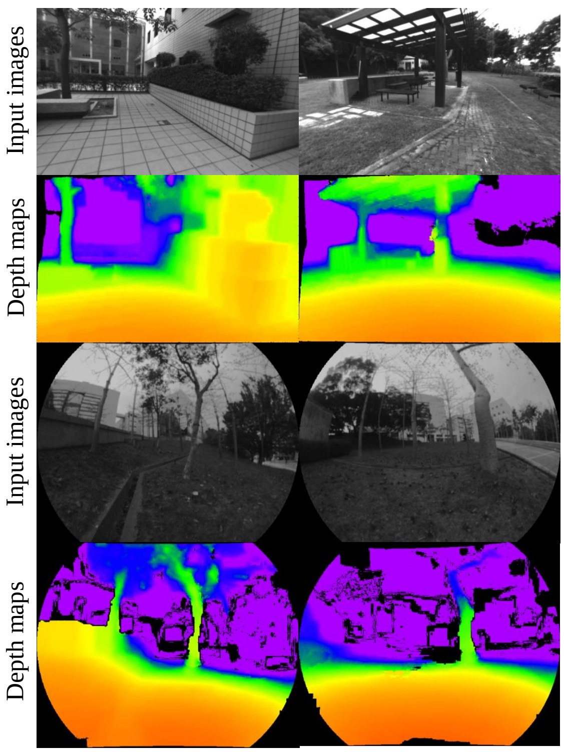


Fig. 8. We test the method in outdoor environments using pinhole and fisheye cameras. Depth maps are color coded the same as Fig. 1. As the results show, our method can capture fine structure as poles and trees. The depth maps generated are smooth and dense.

VII. CONCLUSION

We present a monocular dense mapping system that estimates high-quality dense depth maps in real-time for both pinhole and fisheye cameras. Two core contributions are proposed that significantly improve the performance of the system by utilizing the sequential information. The adaptive baseline matching cost computation is proposed to utilize the multi-baseline observations in the input sequence resulting in robust and accurate depth estimations. The propagated depth filter-based depth refinement is developed to refine all the sequential estimated depth maps and detect outliers. The public dataset, UAV mapping, and handhold experiments are used to demonstrate the performance of our method.

VIII. ACKNOWLEDGEMENT

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

REFERENCES

[1] M. Okutomi and T. Kanade. A multiple-baseline stereo. *IEEE Trans. Pattern Anal. and Mach. Intell.*, 15(4):353–362, April 1993.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[7] 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.

[8] 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.

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

[10] K. Tateno, F. Tombari, I. Laina, and N. Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

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

[12] Y. Hu, S. Rui, and Y. Li. Efficient coarse-to-fine patchmatch for large displacement optical flow. In *Proc. of the IEEE Int. Conf. on Pattern Recognition*, pages 5704–5712. IEEE, 2016.

[13] Q. Chen and V. Koltun. Full flow: Optical flow estimation by global optimization over regular grids. In *Proc. of the IEEE Int. Conf. on Pattern Recognition*, pages 4706–4714. IEEE, 2016.

[14] 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.

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

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

[17] W. Gao and S. Shen. Dualfisheye omnidirectional stereo. In *Proc. of the IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, Vancouver, Canada, September 2017.

[18] 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.