2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

DUI-VIO: Depth Uncertainty Incorporated Visual Inertial Odometry based on an RGB-D Camera

He Zhang and Cang Ye, *Senior Member, IEEE*

***Abstract*— This paper presents a new visual-inertial odometry, term DUI-VIO, for estimating the motion state of an RGB-D camera. First, a Gaussian mixture model (GMM) to is employed to model the uncertainty of the depth data for each pixel on the camera's color image. Second, the uncertainties are incorporated into the VIO's initialization and optimization processes to make the state estimate more accurate. In order to perform the initialization process, we propose a hybrid-perspective-n-point (PnP) method to compute the pose change between two camera frames and use the result to triangulate the depth for an initial set of visual features whose depth values are unavailable from the camera. Hybrid-PnP first uses a 2D-2D PnP algorithm to compute rotation so that more visual features may be used to obtain a more accurate rotation estimate. It then uses a 3D-2D scheme to compute translation by taking into account the uncertainties of depth data, resulting in a more accurate translation estimate. The more accurate pose change estimated by Hybrid-PnP help to improve the initialization result and thus the VIO performance in state estimation. In addition, Hybrid-PnP make it possible to compute the pose change by using a small number of features with a known depth. This improves the reliability of the initialization process. Finally, DUI-VIO incorporates the uncertainties of the inverse depth measurements into the nonlinear optimization process, leading to a reduced state estimation error. Experimental results validate that the proposed DUI-VIO method outperforms the state-of-the-art VIO methods in terms of accuracy and reliability**

I. Introduction

Nowadays, robots have found applications in agriculture, manufacturing, mining and self-driving vehicles, as well as other areas such as surveillance, rescue, deep sea and even space exploration [1]. To operate autonomously, a robot needs to map its environment and localize itself in the environment. This problem is generally known as simultaneous localization and mapping (SLAM). Visual SLAM (vSLAM) has gained extensive attention since the influential work [2] showing that vSLAM can run in real time on a personal computer. The state-of-the-art vSLAM methods have demonstrated their effectiveness with impressive results in various applications [3], [4]. Unfortunately, a vSLAM method cannot recover the absolute scale of the environment due to the use of a single monocular camera. Also, they cannot reliably track the camera pose when the images are impacted by illumination change, low-texture scene, and/or motion blur. To tackle these issues, an inertial measurement unit (IMU) is coupled with the camera for scale deterministic pose estimation. In the robotics community, a navigation system consisting of a camera and an IMU is called a visual-inertial navigation system (VINS) and the method to fuse the visual and inertial data for motion state estimation is termed visual-inertial odometry (VIO). Monocular-camera-based VINS has been a popular solution [5] to robust motion estimation for an aerial vehicle (UAV) or handheld applications. However, a monocular VINS may fail to estimate the scale under some motion conditions [6]. The scale problem can be eliminated if a stereo camera [7], [8] or an RGB-D camera [9], [10] is used instead. For an RGB-D camera-based VIO, there exists two approaches for state estimation. One is to use a visual odometry (VO) to compute the pose change by using the depth data of the associated visual features and fuse the VO-estimated pose change with that of the IMU's preintegration [9], [11]. Another is to compute the visual features' 3D positions from the depth measurements and minimize the features' reprojection residuals together with the IMU's preintegration residuals by an iterative process [10], [12]. The existing methods using either scheme assume an accurate depth measurement and treats the depth value of a visual feature as a constant in the iterative optimization process [10], [12]. However, the depth measurement of an RGB-D camera may incur a large uncertainty, which can lead to a large state estimation error if it is not taken into account.

In this work, we introduce a new VIO method, call DUI-VIO based on an RGB-D camera. It models the uncertainties for the depth and inverse depth measurements of the camera by a Gaussian mixture model (GMM) and incorporate the estimated uncertainties into the VIO's initialization and optimization processes to improve the accuracy of motion state estimation. A new PnP method is developed to provide a more accurate estimate for the VINS' transformation matrix for a more reliable VIO initialization. The method decouples the rotation and translation computations because the computation of the rotation does not require the depth data of the visual features but that of the translation does. This way, more visual features (w/ and w/o depth data) are used to compute a more accurate rotation, based which the translation is determined. This results in a more accurate estimate on the transformation matrix. The superiority of the proposed DUI-VIO has been validated by our experiments.

II. Related Work

In [9] an extended Kalman filter (EKF) is employed for localization by integrating the pose estimates by an RGB-D

This work was supported in part by the National Eye Institute (NEI) of the National Institute of Health under Award R01EY026275. The content is solely the responsibility of the authors and does not necessarily represent the official views of the funding agency. H. Zhang and C. Ye are with Computer Science Department, Virginia Commonwealth University, Richmond, VA 23284, USA. (e-mail: hzhang8@vcu.edu, cye@vcu.edu).

and an IMU. The EKF uses the IMU-estimated pose change to generate state prediction and treats the pose estimated by a visual SLAM as the observation. The innovation is used to update the state. Laidlow *et al*. [11] propose a dense RGB-D-inertial SLAM method. The method combines the residuals of the photometric per-pixel measurements, geometric point-to-plane distances, and IMU preintegration in a cost function. It then estimates the system's optimal motion state by minimizing the cost function by a Gauss-Newton iterative process. However, this method requires a GPU to process the dense camera data for real-time computation and is thus unsuitable to a resource-limited mobile platform [13], [14], [15]. Guo and Roumeliotis [16] propose a sparse feature-based RGBD-IMU calibration method. They analyze the observability of the VINS' state variables and design a strategy that updates the state variable only along their observable directions. The strategy results in a better filter consistency. Due to the use of sparse visual features, the method is computationally effective and does not require GPU speedup. But it does not explicitly estimate the VINS' initial state, which is crucial to the proper function of the subsequent state estimation process [17]. As revealed by the benchmark comparison study [15], VINS-Mono [5] has the most accurate result because of its robust initialization process. However, the initialization requires sufficient IMU excitement, without which an inaccurate scale estimation may be resulted, causing VINS-Mono to ill-perform. Recent RGB-D camera-based VIO methods [10], [12] address this issue by using the camera's depth data to obtain the absolute scale. The method proposed by Ling *et al*. [12] extracts ORB features [4] and uses a 3D-3D-perspective n-point (PnP) method [18] to compute the visual structure (including camera poses and feature positions). It then aligns the estimated camera poses with the IMU integration to estimate the VINS' initial state, including the IMU's poses, velocities, bias, and gravity direction. Shan *et al*. [10] use corner points [19] and employ a 3D-2D-PnP [20] method to build the visual structure. After initialization, their method estimates the VINS' motion state and the inverse depths of the tracked visual features through a nonlinear optimization process. If a feature's depth is provided by the RGB-D camera, the inverse depth value is treated as a constant. However, the reality is that depth measurement, particularly for a point near the edge of an object, may contain significant noise [21], making the value inaccurate. This underscores the need to estimate the uncertainty of the visual feature's inverse depth and factor it into the state estimation. Differing from the previous methods, our proposed DUI-VIO method explicitly computes the uncertainty of the RGB-D camera's depth data and incorporates the uncertainty into the initialization and nonlinear optimization processes for motion state estimation.

III. Structure Core and Coordinate Systems

In this work, the RGB-D camera we chose is the Structure Core (SC) produced by Occipital Inc. (Fig.1). It has a color camera and two Infrared (IR) cameras. The two IR cameras form a stereo imaging system that is capable of measuring a depth ranging from 0.4 to 5+ meters. Moreover, to improve the quality of depth data, it uses an infrared laser projector to project a static infrared pattern on the scene to ease stereo matching. A disparity map estimated from the stereo matching is used to generate the depth data by triangulation. Based on the known displacement between the color camera and the IR camera, the depth data is aligned to the color image. Therefore, the SC is able to provide aligned color and depth image (resolution: 640×480 pixels) at 30 fps. Also, the SC has an IMU sensor that is able to provide both accelerometer and gyroscope data with an update rate up to 800 fps. The inclusion of the IMU makes the SC an RGB-D camera-based VINS. The coordinate systems of the IMU and the camera are defined and labeled in Fig. 1.

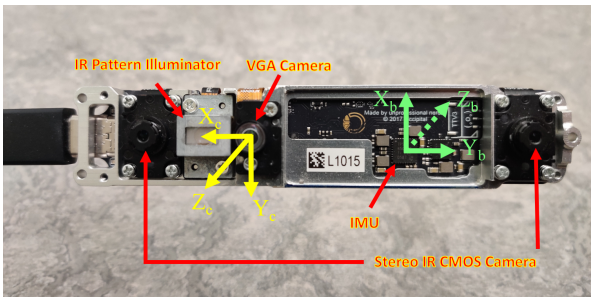

Fig. 1: Components of the Occipital Structure Core (SC): a color VGA camera, two IR cameras, an IR projector and an IMU. The body/IMU and camera coordinate systems are denoted by $\{B\}$ $(X_bY_bZ_b)$ and $\{C\}$ $(X_cY_cZ_c)$, respectively. The initial $\{B\}$ is taken as the world coordinate system $\{W\}$. In this paper, the super scripts b and c describe a variable in $\{B\}$ and $\{C\}$, respectively. The transformation matrix between $\{B\}$ and $\{C\}$ is precalibrated and denoted $T_c^b = [R_c^b;\ \mathbf{t}_c^b]$, where R_c^b is rotation and $\mathbf{t}_c^b$ is translation.

IV. Uncertainty Analysis

In the literature, an RGB-D camera's depth measurement for a pixel in the color image is assumed independent of its neighboring points [22], [23]. Given an image pixel $\mathbf{q} = (u, v)$ with a depth measurement d_{uv}, the corresponding 3D point can be expressed in $\{C\}$ as $\mathbf{p} = [x, y, z]^T$, where:

$$x = \frac{z}{f_x}(u - c_x), y = \frac{z}{f_y}(v - c_y), z = d_{uv} \tag{1}$$

Here, (f_x, f_y), and (c_x, c_y) are the color camera's focal length and optical center, respectively. The depth can be computed by triangulation $d_{uv} = \frac{f_x b}{\rho_{uv}}$ [23], where b and ρ_{uv} are the baseline of the IR cameras and the image disparity for pixel $\mathbf{q}$, respectively. The inverse depth is computed by $\lambda_{uv} = \frac{\rho_{uv}}{f_x b}$ and the standard deviations for z and λ can be obtained by:

$$\sigma_z = \frac{d_{uv}^2}{f_x b}\sigma_\rho, \quad \sigma_\lambda = \frac{\sigma_\rho}{f_x b} \tag{2}$$

respectively, where σ_ρ is the standard deviation for ρ.

A. Depth and Inverse depth Uncertainties modeled by Gaussian Mixture Model

Dryanovski *et al*. [21] employ a Gaussian mixture model (GMM) to obtain a more accurate depth uncertainty estimation based on the assumption that the depth error of a point is dependent on those of its neighboring image

pixels. They also assume that the coordinates of a pixel have Gaussian distributed errors. In this work, we use the GMM [21] to compute the uncertainties of the depth and the inverse depth measurements. Specifically, assuming the u and v of are independent random variables with normal distributions $\mathcal{N}(\mu_u, \sigma_u)$ and $\mathcal{N}(\mu_v, \sigma_v)$, respectively, and the raw measurement z for pixel (u, v) is normally distributed $\mathcal{N}(\mu_z, \sigma_z)$ where $\mu_z = d_{uv}$ and σ_z is computed by equation (2), we define a random variable $\hat{z}$ as a mixture of the z values for all pixels within a local window $S = \{(i, j) : i \in [\mu_u - 1, \mu_u + 1], j \in [\mu_v - 1, \mu_v + 1]\}$. The weight of the mixture w_{ij} is computed by:

$$w_{ij} = exp\left(-\left(\frac{(i-u)^2}{2\sigma_u^2} + \frac{(j-v)^2}{2\sigma_v^2}\right) - \log\left(\frac{\eta(d_{ij} - d_{uv})^2}{\sigma_z^2 \sigma_S^2} + 1\right)\right) \quad (3)$$

where d_{ij} is μ_z for pixel (i, j) and η is a scale factor. σ_S^2 represents the variance of the depth measurements in the local window S. The first part of w_{ij} represents spatial correlation while the second part describes the correlation related to depth similarity. Then, the mean and variance of the resulting Gaussian mixture are computed as follows:

$$\mu_{\hat{z}} = \frac{\sum_{i,j} w_{ij} d_{ij}}{\sum_{i,j} w_{ij}}, \quad \sigma_{\hat{z}}^2 = \frac{\sum_{i,j} w_{ij}(d_{ij}^2 + \sigma_{z_{ij}}^2)}{\sum_{i,j} w_{ij}} - \mu_{\hat{z}}^2 \quad (4)$$

where $\sigma_{z_{ij}}$ is σ_z for pixel (i, j). For the inverse depth λ, we assume the raw measurement λ at (u, v) is also normally distributed $\mathcal{N}(\mu_\lambda, \sigma_\lambda)$, where $\mu_\lambda = \frac{1}{d_{uv}}$ and σ_λ is computed according to equation (2). Notice that σ_λ is constant that is independent of the location (u, v) [23]. We define $\hat{\lambda}$ as a mixture of the λ variables for those pixels in S, and the mean and variance of the Gaussian mixture for inverse depth $\hat{\lambda}$ are computed as:

$$\mu_{\hat{\lambda}} = \frac{\sum_{i,j} \frac{w_{ij}^{'}}{d_{ij}}}{\sum_{i,j} w_{ij}^{'}}, \quad \sigma_{\hat{\lambda}}^2 = \frac{\sum_{i,j} w_{ij}^{'}(\frac{1}{d_{ij}^2} + \sigma_\lambda^2)}{\sum_{i,j} w_{ij}^{'}} - \mu_{\hat{\lambda}}^2 \quad (5)$$

where $w_{ij}^{'}$ is similar to equation (3) by replacing d_{ij}, d_{uv}, σ_z^2 and σ_S^2 with $\frac{1}{d_{ij}}$, $\frac{1}{d_{uv}}$, σ_λ^2 and $\sigma_S'^2$, respectively. $\sigma_S'^2$ is the variance of the inverse depth measurements for those pixels in the local window S.

B. 3D covariance matrix

Deriving from equation (1) and replacing z with $\hat{z}$, the 3D covariance matrix $\Sigma_{\mathbf{p}}$ of a point $\mathbf{p}$ in the camera coordinate system can be computed by:

$$\Sigma_{\mathbf{p}} = \begin{bmatrix} \sigma_x^2, \sigma_{xy}, \sigma_{xz} \\ \sigma_{yx}, \sigma_y^2, \sigma_{yz} \\ \sigma_{zx}, \sigma_{zy}, \sigma_z^2 \end{bmatrix} \quad (6)$$

where $\sigma_x^2 = \frac{\sigma_{\hat{z}}^2(\mu_u')^2 + \sigma_u^2 g}{f_x^2}, \sigma_y^2 = \frac{\sigma_{\hat{z}}^2(\mu_v')^2 + \sigma_v^2 g}{f_y^2}, \sigma_{xz} = \sigma_{zx} = \frac{\sigma_{\hat{z}}^2 \mu_u'}{f_x}$, $\sigma_{yz} = \sigma_{zy} = \frac{\sigma_{\hat{z}}^2 \mu_v'}{f_y}, \sigma_{xy} = \sigma_{yx} = \frac{\sigma_{\hat{z}}^2 \mu_u' \mu_v'}{f_x f_y}$ with $\mu_u' = \mu_u - c_x$, $\mu_v' = \mu_v - c_y$, $g = \mu_{\hat{z}}^2 + \sigma_{\hat{z}}^2$. The approximations of the mean and covariance of the 3D point $\mathbf{p}$ are used in the initialization phase of the DUI-VIO method.

V. RGB-D Camera-based VIO

The proposed DUI-VIO consists of two components: frontend feature tracking and backend state estimation. Similar to [5] and [10], the feature tracking part extracts visual features [19] from an image and tracks them across images by using KLT [24]. A fundamental-matrix-based RANSAC process is implemented to remove the outliers. It also selects keyframes based on the average parallax difference. If the average parallax of the tracked features between the current frame and the latest keyframe is larger than a threshold (10 pixels), this frame is treated as a keyframe. The tracked features in all frames are passed to the backend process to estimate the VINS' motion state. The backend state estimation starts with a sophisticated initialization process and then proceeds with a nonlinear optimization process for state estimation.

A. Initialization

The initialization procedure first implements a vision-only SfM to estimate the camera poses and feature positions. To reduce computational complexity, we process visual-inertial data inside a sliding window. . The details of the initialization are illustrated in Algorithm 1. At first, feature correspondences are checked between the latest camera frame $\mathcal{F}_M$ and all previous frames. If enough parallax (more than 30 pixels) is found between $\mathcal{F}_l$ and $\mathcal{F}_M$, we estimate the pose change T between them by using a PnP method (line A1.3-4). $\mathcal{F}_l$ is selected as reference frame and T is used to triangulate all features (with no depth data from the depth camera) observed in these two frames (line A1.6). The output of the triangulation is a set of 3D feature points denoted as Γ which is represented in the reference frame. Given Γ, we estimate the poses of all other frames in the sliding window by the iterative procedure (Function $PnPAllFrames$ in line A1.12): 1) given (Γ, S_p), employ the OpenCV function $solvePnP$ to estimate the pose of the current frame i; 2) triangulate the features at $\mathcal{F}_i$ and add them to Γ; 3) repeat steps 1 and 2 for the next frame until all frames are processed. Next, a global bundle adjustment is employed to optimize the camera poses by minimizing the feature reprojection residuals (line A1.16). Finally, visual-inertial alignment is applied to estimate the IMU's poses, velocities, biases and gravity direction by aligning the camera poses with the IMU pre-integration (line A1.17). More details about visual-inertial alignment can be found in [10].

The success of the function $PnPAllFrames$ depends on the accuracy of Γ which is estimated by feature triangulation (line A1.6). The accuracy of the triangulation is determined by the pose change estimate T. Therefore, the reliability of the initialization is dependent on the accuracy of T (line A1.4). Differing from VINS-Mono [5] that compute the camera's rotation and translation by the 2D-2D-PnP method [25], VINS-RGBD [10] employs the 3D-2D-PnP method (called EPnP) [20] to compute the pose change since depth data are available. However, EPnP tends to result in a larger rotation error because: 1) a depth measurement error (quadratically increases with depth) may result in a reprojection error that is larger than the image noise; 2) a

pose change computed from a low number of features with depth data may be inaccurate. To tackle these issues, we propose a new PnP method called Hybrid-PnP as illustrated in Algorithm 2. First, we employ the 2D-2D-PnP method [25]] to estimate the rotation (line A2.2) by using all features (both w/ and w/o depth data). Meanwhile, the inliers of the 2D features are obtained. After selecting the inliers with depth measurement (line A2.6), we estimate the translation of the camera by minimizing the reprojection error of the de-rotated features (line A2.10).

Specifically, given the rotation R_{ji} between two frames i and j, we can estimate the translation $\mathbf{t}_{ji}$ using the following strategy. Assuming M features $\{\mathbf{p}_k^i, \overline{\mathbf{p}}_k^j\}$, for $k = 1...M$, observed in frames $\mathcal{F}_i$ and $\mathcal{F}_j$, where $\mathbf{p}_k^i = [X_k^i, Y_k^i, Z_k^i]^T$ is the coordinates of feature k in $\mathcal{F}_i$, $\overline{\mathbf{p}}_k^j = [\overline{X}_k^j, \overline{Y}_k^j, 1]^T$ is the normalized coordinates of feature k in $\mathcal{F}_j$ where z is one. Assume the depth in $\mathcal{F}_j$ is z_k^j, we have:

$$z_k^j \overline{\mathbf{p}}_k^j = R_{ji}\mathbf{p}_k^i + \mathbf{t}_{ji} \tag{7}$$

Equation (7) has three rows. Substituting z_k^j in the third row into the first and second rows results in:

$$\begin{aligned}(R_1 - \overline{X}_k^j R_3)\mathbf{p}_k^i + t_1 - \overline{X}_k^j t_3 = 0 \\ (R_2 - \overline{Y}_k^j R_3)\mathbf{p}_k^i + t_2 - \overline{Y}_k^j t_3 = 0\end{aligned} \tag{8}$$

where R_i and t_i $(i = 1, 2, 3)$ are the i-th row of R_{ji} and $\mathbf{t}_{ji}$, respectively. Therefore each feature can provide a 2-dimensional constraint and at least two feature points are needed to estimate $\mathbf{t}_{ji}$. Since we can compute the 3D covariance of a feature point, we use a weighting scheme to combine the constraints resulted from different feature points. For feature point k, the residual vector of $\mathbf{t}_{ji}$ is given by:

$$\mathbf{r}_k = W_k(A_k\mathbf{t}_{ji} - \mathbf{b}_k) \tag{9}$$

where

$$A_k = \begin{bmatrix}1, 0, -\overline{X}_k^j \\ 0, 1, -\overline{Y}_k^j\end{bmatrix}, \mathbf{b}_k = \begin{bmatrix}(\overline{X}_k^j R_3 - R_1)\mathbf{p}_k^i \\ (\overline{Y}_k^j R_3 - R_2)\mathbf{p}_k^i\end{bmatrix},$$

$$W_k = \begin{bmatrix}\frac{1}{\sqrt{J_x \Sigma_{\mathbf{p}_k^i} J_x^T}}, 0 \\ 0, \frac{1}{\sqrt{J_y \Sigma_{\mathbf{p}_k^i} J_y^T}}\end{bmatrix}, J_x = R_1 - \overline{X}_k^j R_3, J_y = R_2 - \overline{Y}_k^j R_3$$

$\Sigma_{\mathbf{p}_k^i}$ is the covariance of the 3D point $\mathbf{p}_k^i$ in $\mathcal{F}_i$ and it is computed by Eq. (6). The optimal translation that minimizes the residual vectors of all tracked features can be obtained by solving the following optimization problem:

$$\mathbf{t}_{ji}^* = \arg\min_{\mathbf{t}_{ji}} \sum_{k=1}^{M} ||\mathbf{r}_k||^2 \tag{10}$$

Eq. (10) can be solved by the Levenberg–Marquardt (LM) algorithm (line A2.10). Finally, the pose estimation $[R_{ij}, \mathbf{t}_{ji}]$ is further improved by using the Gaussian Newton (GN) scheme (line A2.11). Because the LM-resulted solution is quite accurate, GN only takes no more than three iterations. By incorporating the uncertainty of the camera's depth data, a more accurate translation is obtained.

Algorithm 1: Initialization

input : camera frames F=$\{\mathcal{F}_i, i = 1...M\}$, IMU preintegration P_u, 2D feature set $S_p = \{pt_i, i = 1...L\}$ and depth set $S_d = \{d_i, i = 1...K\}$ corresponding to points in S_p with depth measurement

output: poses, velocities and biases of IMU at F and gravity direction

```
Γ ← ∅, P_f ←{T_0}
for l ← 1 to M − 1 do
    if largeDisparityBetween(F_l, F_M, S_p) then
        [T, success] = Hybrid-PnP(F_l, F_M, S_p, S_d)
        if success is true then
            Γ = triangulation(T, F_l, F_M, S_p, S_d)
            P_f ← P_f ∪ T
            break
        end
    end
end
[Γ, P_f, success] = PnPAllFrames(Γ, P_f, F, S_p, S_d)
if success is false then
    return [_]
end
[P_f] = bundleAdjustment(Γ, P_f, S_p)
[B_f, V_f, g] = visualIniertialAlignment(P_f, P_u)
return [P_f, B_f, V_f, g]
```

Algorithm 2: Hybrid-PnP

input : camera frames $\mathcal{F}_i$ and $\mathcal{F}_j$, S_p and S_d

output: pose change estimation T_{ji} to transform a point from $\mathcal{F}_i$ to $\mathcal{F}_j$

```
Θ= findMatchedFeatures(F_i, F_j, S_p)
[R_ji, Θ] = 2D-2D-PnP(Θ)
if Θ == ∅ then
    return [_, false]
end
Θ = findInliersWithDepth(Θ, S_d)
if Θ == ∅ then
    return [_, false]
end
t_ji = translationEstimate(R_ji, Θ)
[R_ji, t_ji] = GaussianNewton(R_ji, t_ji, Θ)
return [[R_ji, t_ji], true]
```

B. State Estimator

After initialization, a sliding window-based nonlinear optimization process is employed for state estimation. The full state vector in the sliding window is defined as $\boldsymbol{\chi} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n, \lambda_1, \lambda_2, ..., \lambda_m\}$, where $\mathbf{x}_i = \{\mathbf{t}_{b_i}^w, \mathbf{v}_{b_i}^w, \mathbf{q}_{b_i}^w, \mathbf{b}_a, \mathbf{b}_g\}$ $(i \in [1, n])$ is the IMU's motion state (translation, velocity, rotation, accelerometer bias and gyroscope bias) at the time when the i^{th} keyframe is captured. n is the size of the sliding window ($n = 10$ in this work) and m is the total number of features in the sliding window. λ_k

$(k = 1...m)$ is the estimate for the inverse distance of the kth feature from its first observation. Similar to [5] and [10], we estimate the state vector $\boldsymbol{\chi}$ by minimizing the sum of prior and the Mahalanobis norm of all measurement residuals as:

$$\boldsymbol{\chi}^* = \arg\min_{\boldsymbol{\chi}} \Big\{ ||^p\mathbf{r}||^2 + \sum_j ||^u\mathbf{r}_j||^2 + \sum_{(k,j)} \rho(||^p\mathbf{r}_{kj}||^2) + \sum_{(k,j)} \rho(||^{pd}\mathbf{r}_{kj}||^2) + \sum_k ||^d\mathbf{r}_k||^2 \Big\} \quad (11)$$

where $\rho()$ is the Huber loss function [26]. ${}^p\mathbf{r}$, ${}^u\mathbf{r}_j$, and ${}^p\mathbf{r}_{kj}$ represent the residuals for the prior information from marginalization, IMU preintegration (between keyframes $j-1$ and j) and feature reprojection, respectively. Their definition are referred to [5]. The other residuals, ${}^{pd}\mathbf{r}_{kj}$ and ${}^d\mathbf{r}_k$ are defined as follows by using the inverse depth.

1) computation of ${}^{pd}\boldsymbol{r}_{kj}$: By using a normalized coordinate system, the k^{th} feature point that was first observed at keyframe i is denoted by $\overline{\mathbf{p}}^i_k = [\overline{X}^i_k, \overline{Y}^i_k, 1]^T$. Its estimated inverse depth is $\hat{\lambda}_k$. Note that we drop subscript k for simplicity from now on. If the feature point is tracked onto keyframe j, the mean $(\mu_{\hat{\lambda}})$ and variance $(\sigma^2_{\hat{\lambda}})$ of its inverse depth measurement at keyframe j can be computed by Eqn. (5). The feature point can be expressed in C_j (i.e., the camera coordinate system for keyframe j) as $\mathbf{p}_j = [X_j, Y_j, Z_j]^T = R_{ji}\frac{\overline{\mathbf{p}}^i}{\hat{\lambda}} + \mathbf{t}_{ji}$, where R_{ji} and $\mathbf{t}_{ji}$ are the rotation matrix and translation from C_j to C_i. By translating $\mathbf{p}_j$ into its normalized form, we can compute the residual vector by: ${}^{pd}\mathbf{r}_{kj} = \sqrt{\Sigma^{-1}_{c_j}} \left[\frac{X_j}{Z_j} - \overline{X}_j, \frac{Y_j}{Z_j} - \overline{Y}_j, \frac{1}{Z_j} - \mu_{\hat{\lambda}} \right]^T$ where Σ_{c_j} is the measurement covariance that is defined by $\Sigma_{c_j} = diag(\frac{\sigma^2_\tau}{f^2_x}, \frac{\sigma^2_\tau}{f^2_y}, \sigma^2_{\hat{\lambda}})$. σ_τ is the image noise and it is set to 1.5 pixels in this work.

2) computation of ${}^d\boldsymbol{r}_k$: If the inverse depth measurement for the k^{th} feature (observed at the i^{th} keyframe) is available from the camera's depth data, its mean $(\mu_{\hat{\lambda}})$ and variance $(\sigma^2_{\hat{\lambda}})$ can be computed by Eqn. (5). Therefore, the ${}^d\mathbf{r}_k$ is computed by ${}^d\mathbf{r}_k = \frac{\hat{\lambda} - \mu_{\hat{\lambda}}}{\sigma_{\hat{\lambda}}}$.

The above residuals ${}^{pd}\mathbf{r}_{kj}$ and ${}^d\mathbf{r}_k$ encode the measurements of the inverse depth based on GMM. The inverse depth measurement is tightly incorporated into the nonlinear optimization process to make more accurate state estimation. The nonlinear optimization problem (11) is solved by using the Ceres solver [27].

VI. Experiment

A. PnP Comparison

We compared the proposed Hybrid-PnP with the state-of-the-art EPnP [20] by simulation. We produced 1000 synthetic 2D features in a 640×480 image by using a virtual camera with a focal length $f_x = f_y = 525$ and a principal point at $(320, 240)$. These features' depth measurementwas were generated by using a uniform distribution in the range $[2, 5]$ meters and their 3D positions were computed accordingly. We then changed the camera viewpoint by applying a rotation R_{true} and a translation t_{true} to the camera's coordinate system. By reprojecting the 3D feature points onto the new

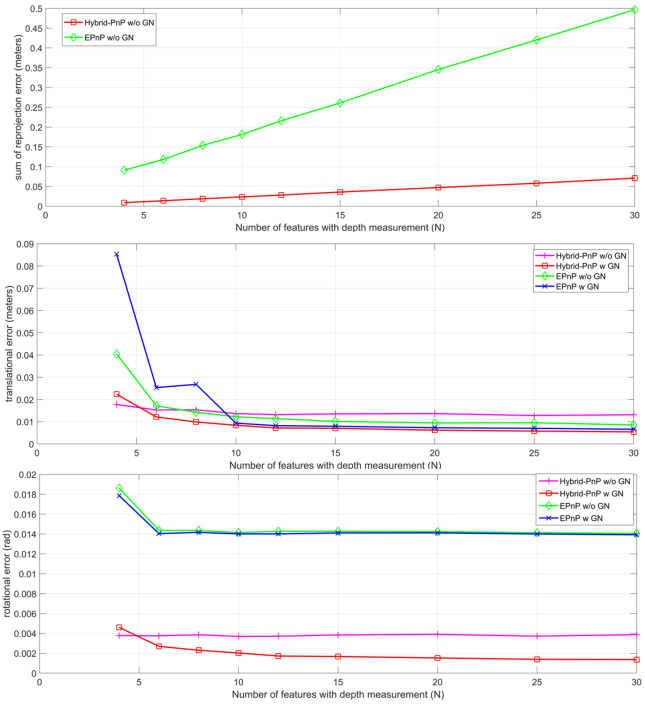

Fig. 2: PnP Comparison. Top: Reprojection Error; Middle: Translation Error; Bottom: Rotation Error.

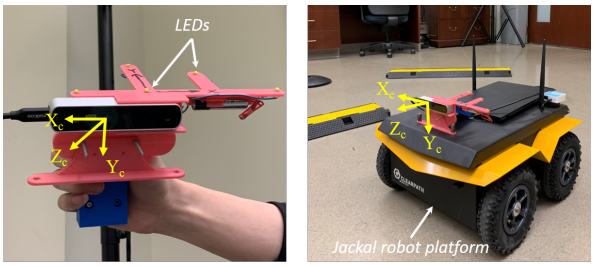

Fig. 3: Handheld and Mobile Robot for data collection

camera image plane, we obtain their observations in the new camera coordinate system. A zero mean Gaussian noise $(\sigma_u = 1, \sigma_v = 1)$ was then added to the coordinates of the features. Also, a Gaussian noise that quadratically increases with depth [23] was added to the depth measurements. Given the feature matches between the two camera frames, we employed Hybrid-PnP and EPnP methods to compute the pose change estimate, denoted as $[R_{est}, t_{est}]$. We then computed the relative angular and translation error for the estimation. The angular error e_θ and the translation error e_t can be computed by $R_e = R^{-1}_{true} R_{est}$ and $e_t = ||t_{true} - t_{est}||$, respectively. t_{true} is set to $[0.2, 0.05, 0.3]^T$, while the rotation R_{true} is generated by rotating the camera by 7, 5, and 10 degrees around the x, y, and z axis, respectively. We carried out nine tests and in each test we independently ran the hybrid-PnP and EPnP methods w/ and w/o GN for 500 times to estimate the pose changes. In each test, we assumed that at least 30 inliers can be found and only a portion of these inliers have depth measurements. In the 2D-2D-PnP function (line A2.2) of the Hybrid-PnP, we used 30 features for rotation estimation and we used N ($N <= 30$) features for translation estimation. We also used these N features in

TABLE I: Results on the Lab Datasets: RMSE in meters of the estimated trajectory of each VIO method. In each row, the best result is bolded. TL - Trajectory Length, X – Failed to initialize or to converge in the state estimator

Dataset	VINS-Mono	Vins-Fusion	VINS-RGBD	DUI-VIO	TL (m)
H1	0.142	0.178	0.137	**0.109**	15
H2	0.175	0.324	0.431	**0.151**	17
H3	0.236	0.224	**0.116**	0.123	23
H4	0.619	0.827	0.638	**0.602**	30
H5	0.373	0.873	0.378	**0.372**	80
H6	0.706	0.676	0.725	**0.549**	84
R1	X	X	X	**0.258**	15
R2	X	X	1.174	**0.179**	28
R3	0.402	X	0.308	**0.191**	25
R4	0.971	X	X	**0.134**	43

EPnP to compute both rotation and translation. This is to simulate the real cases that some of the matched features do not have depth measurements. In Fig. 2, we first show the sum of the feature reprojections given the estimated transformation by Hybrid-PnP and EPnP. It shows that Hybrid-PnP consistently generate much smaller feature reprojection error than EPnP. Then the mean of $[e_\theta, e_t]$ for each method are plotted. We also compare the errors of Hybrid-PnP and EPnP after the refinement by a GN optimization process [20]. From the result shown in Fig. 2, we can see that Hybrid-PnP w/ or w/o GN produced a much smaller rotation error than the EPnP counterparts. Also, the translation error of Hybrid-PnP with GN is the smallest. Finally, Hybrid-PnP can provide accurate pose change estimation even when the number of features with depth is very small.

B. DUI-VIO Evaluation

In this section, we carried out thirteen experiments to compare the pose estimation performance of the proposed DUI-VIO with the state-of-the-art VIO methods: VINS-Mono [5], VINS-Fusion [7], and VINS-RGBD [10]. Ten of the experiments were conducted in our laboratory and the rest were in the hallways of the East Engineering Building on campus. The pose estimation results were summarized below.

1) Laboratory Experiments: Six of the ten experiments were conducted in our laboratory by handholding the SC (Fig. 3) and walking in a looped trajectory at a normal walking speed $0.6m/s$. For the first three experiments we moved the camera smoothly, while for the other three, we rotated the camera heavily on purpose. The last four experiments were carried out by installing the camera on a wheeled robot (Fig. 3) and driving the robot to move on a looped trajectory at a speed $0.2m/s$. At the beginning of each experiment using the wheeled robot, we first handheld and rotated the SC for about five seconds to excite the visual-inertial system to allow for a good system initialization. Then we installed the SC on the robot and drove the robot for the experiment. By using the ground truth trajectory generated by our OptiTrack motion capture system, we calculated the absolute pose error for each point on the trajectories generated by DUI-VIO, VINS-Mono, VINS-Fusion, and VINS-RGBD. Table I summarizes the results, the smallest error was highlighted in boldface. As can be seen from the table, our proposed algorithm succeeded in every experiment while the other methods failed in most of the experiments with the wheeled robot. In addition, DUI-VIO has the smallest RMSE in nine of the ten experiments and its RMSE is only slightly larger than that of VINS-RGBD in one experiment. This demonstrates that DUI-VIO has a much more accurate pose estimation than the other methods. Fig. 4. compares the trajectories generated by the four methods for some of the experiments.

TABLE II: EPEN (%) statistics for each method, the best result if bolded. TL - Trajectory Length, X – Failed to initialize

Dataset	VINS-Mono	Vins-Fusion	VINS-RGBD	DUI-VIO	TL (m)
H1	1.54	2.5	2.1	**1.42**	185
H2	1.69	1.33	X	**0.85**	174
H3	1.86	3.85	X	**1.03**	180
Mean	1.7	2.56	X	**1.1**	180

2) Real World Experiments: To validate the DUI-VIO in the real world, we carried out three experiments by handholding the SC and walking in a looped trajectory (i.e., returning to the starting point) in the hallways of the East Engineering Building. The Endpoint Position Error Norm (EPEN) in percentage of the path-length is use as the metric for pose estimation accuracy. The results are tabulated in Table II. It can be seen that DUI-VIO achieved a smaller EPEN than the other methods in all of the four experiments. On average, DUI-VIO reduces the EPEN by 35.3%, 57%, and 47.6% compared to VINS-Mono, VINS-Fusion, and VINS-RGBD, respectively. Fig. 5 compares the trajectories estimated by the four methods, from which it can be observed that DUI-VIO resulted in a more accurate trajectory. In two of the experiments, we observed that VINS-RGBD failed in its initialization step. The reason was that it failed to find enough 3D feature points for pose change estimation. In contrast, DUI-VIO could reliably initialized itself and thus achieved more accurate pose estimation results.

VII. Conclusion

We proposed a new VIO method—DUI-VIO—for state estimation for an RGB-D camera-based VINS. We employed GMM to estimate the uncertainties for the depth and the inverse depth measurement from an RGB-D camera. The modeled uncertainties are incorporated into the initialization and nonlinear optimization processes of the VIO to improve reliability and accuracy. In the initialization process, we also proposed a Hybrid-PnP to reliably boost the initialization process of the DUI-VIO. We carried out extensive experiments to show that DUI-VIO outperforms the state-of-the-art in terms of reliability and accuracy.

References

[1] M. Ben-Ari and F. Mondada, "Robots and their applications," in *Elements of Robotics*. Springer, 2018, pp. 1–20.

[2] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

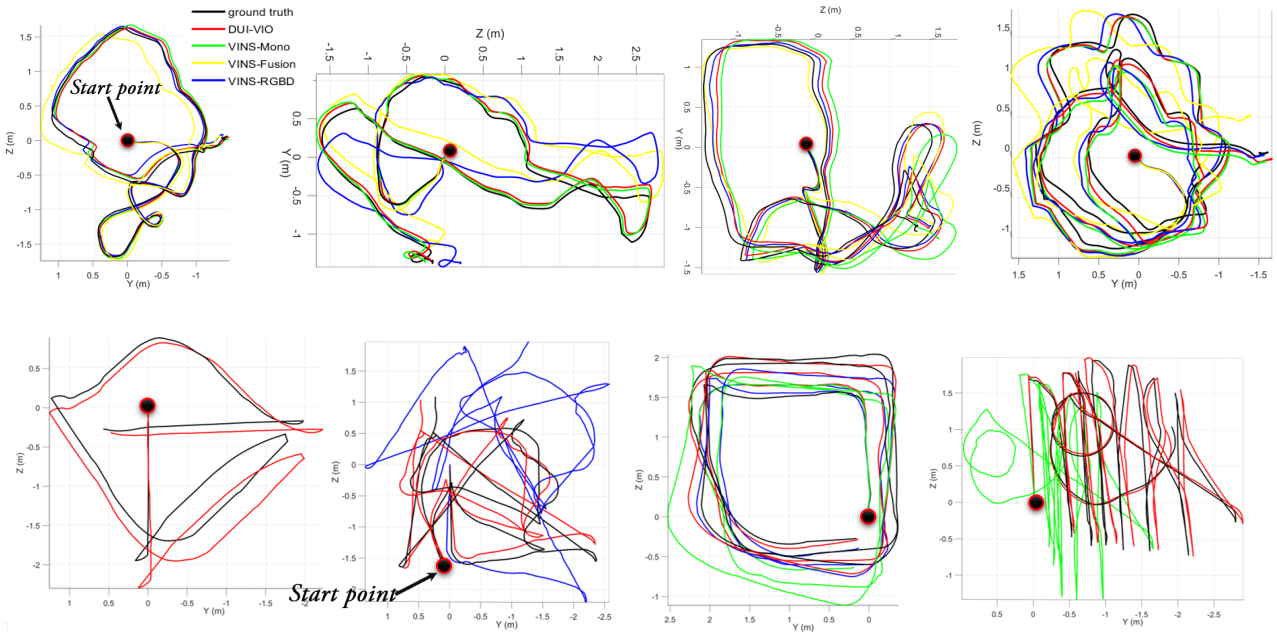

Fig. 4: Top: Trajectory Comparison for Handheld datasets(H1-H4) in the Lab. Start point is labeled by dark solid circle in each dataset; Bottom: Trajectory Comparison for Robot datasets (R1-R4) in the Lab

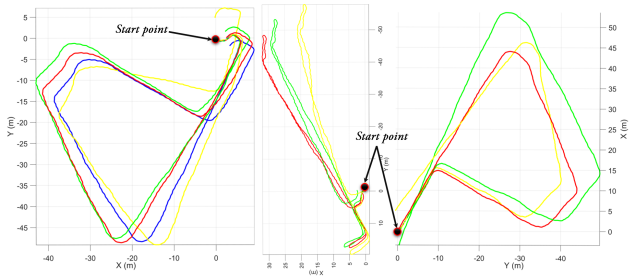

Fig. 5: Trajectory Comparison for the Corridor datasets

[3] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *Proc. of European conference on computer vision*. Springer, 2014, pp. 834–849.

[4] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[5] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[6] K. J. Wu, C. X. Guo, G. Georgiou, and S. I. Roumeliotis, "Vins on wheels," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2017, pp. 5155–5162.

[7] T. Qin, J. Pan, S. Cao, and S. Shen, "A general optimization-based framework for local odometry estimation with multiple sensors," *arXiv preprint arXiv:1901.03638*, 2019.

[8] S. Leutenegger *et al.*, "Keyframe-based visual-inertial slam using nonlinear optimization," in *Proc. of Robotis Science and Sys.*, 2013.

[9] N. Brunetto *et al.*, "Fusion of inertial and visual measurements for rgb-d slam on mobile devices," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 1–9.

[10] Z. Shan, R. Li, and S. Schwertfeger, "Rgbd-inertial trajectory estimation and mapping for ground robots," *Sensors*, vol. 19, no. 10, p. 2251, 2019.

[11] T. Laidlow, M. Bloesch, W. Li, and S. Leutenegger, "Dense rgb-d-inertial slam with map deformations," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2017, pp. 6741–6748.

[12] Y. Ling, H. Liu, X. Zhu, J. Jiang, and B. Liang, "Rgb-d inertial odometry for indoor robot via keyframe-based nonlinear optimization," in *Proc. of IEEE Int. Conf. on Mechatronics and Automation (ICMA)*, 2018, pp. 973–979.

[13] M. Li and A. I. Mourikis, "Vision-aided inertial navigation for resource-constrained systems," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 1057–1063.

[14] H. Zhang and C. Ye, "An indoor navigation aid for the visually impaired," in *Proc. of IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2016, pp. 467–472.

[15] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," in *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2502–2509.

[16] C. X. Guo and S. I. Roumeliotis, "Imu-rgbd camera 3d pose estimation and extrinsic calibration: Observability analysis and consistency improvement," in *Proc. of IEEE International Conference on Robotics and Automation*, 2013, pp. 2935–2942.

[17] H. Zhang, L. Jin, H. Zhang, and C. Ye, "A comparative analysis of visual-inertial slam for assisted wayfinding of the visually impaired," in *IEEE Winter Conf. on Applications of Computer Vision (WACV)*, 2019, pp. 210–217.

[18] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2004, pp. I–I.

[19] J. Shi *et al.*, "Good features to track," in *Proceedings of IEEE conf. on computer vision and pattern recognition*, 1994, pp. 593–600.

[20] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision*, vol. 81, no. 2, p. 155, 2009.

[21] I. Dryanovski, R. G. Valenti, and J. Xiao, "Fast visual odometry and mapping from rgb-d data," in *Proc. of IEEE International Conference on Robotics and Automation*, 2013, pp. 2305–2310.

[22] D. Gutiérrez-Gómez, W. Mayol-Cuevas, and J. J. Guerrero, "Inverse depth for accurate photometric and geometric error minimisation in rgb-d dense visual odometry," in *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 83–89.

[23] A. Concha and J. Civera, "Rgbdtam: A cost-effective and accurate rgb-d tracking and mapping system," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2017, pp. 6756–6763.

[24] B. D. Lucas, T. Kanade *et al.*, "An iterative image registration technique with an application to stereo vision."

[25] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[26] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in statistics*. Springer, 1992, pp. 492–518.

[27] S. Agarwal, K. Mierle *et al.*, "Ceres solver," 2012.