sensors

MDPI

Article

RGBD-Inertial Trajectory Estimation and Mapping for Ground Robots

Zeyong Shan [1,2,3,*], Ruijian Li [1] and Sören Schwertfeger [1,*]

1 School of Information Science & Technology, ShanghaiTech University, Shanghai 201210, China; lirj@shanghaitech.edu.cn
2 Chinese Academy of Sciences, Shanghai Institute of Microsyst & Information Technology, Shanghai 200050, China
3 University of Chinese Academy of Sciences, Beijing 100049, China
* Correspondence: shanzy@shanghaitech.edu.cn (Z.S.); soerensch@shanghaitech.edu.cn (S.S.); Tel.: +86-1312-230-0581 (Z.S.); +86-21-2068-5096 (S.S.)

Received: 2 April 2019; Accepted: 7 May 2019; Published: 15 May 2019

check for updates

Abstract: Using camera sensors for ground robot Simultaneous Localization and Mapping (SLAM) has many benefits over laser-based approaches, such as the low cost and higher robustness. RGBD sensors promise the best of both worlds: dense data from cameras with depth information. This paper proposes to fuse RGBD and IMU data for a visual SLAM system, called VINS-RGBD, that is built upon the open source VINS-Mono software. The paper analyses the VINS approach and highlights the observability problems. Then, we extend the VINS-Mono system to make use of the depth data during the initialization process as well as during the VIO (Visual Inertial Odometry) phase. Furthermore, we integrate a mapping system based on subsampled depth data and octree filtering to achieve real-time mapping, including loop closing. We provide the software as well as datasets for evaluation. Our extensive experiments are performed with hand-held, wheeled and tracked robots in different environments. We show that ORB-SLAM2 fails for our application and see that our VINS-RGBD approach is superior to VINS-Mono.

Keywords: visual-inertial systems; SLAM; inertial motion tracking; ground robots; rescue robots; sensor fusion; state estimation; RGBD sensor

1. Introduction

With the sensor and algorithm innovations [1], mobile robots are getting smaller and smarter and are addressing new applications in medicine, agriculture, and security applications [2,3]. The improved sensors allow them to become much more capable for perception, mapping, and navigation.

Unlike indoor navigation, which focuses more on 2D, even and structured environments, rescue robots need more mobility to face various, complicated scenarios such as slopes, stairs, and tunnels [4–6]. Search and rescue robots are usually equipped with LIDAR and high-quality IMU sensors, which are fused with the wheel odometry to estimate the poses and the map [7–9]. However, these leads to expensive robots, which is undesirable, because such mobile rescue robots have a high risk of breaking or being lost during operation. In the past years, we have seen great advances in the area of computer vision, also with respect to vSLAM (visual Simultaneous Localization and Mapping) [10]. Since then, many methods have been proposed to improve the accuracy, robustness, and efficiency of vSLAM, such as using first estimation Jacobian [11] to reduce the inherent nonlinearity in the system. Optimization-based methods also have become very popular [12]. Additionally, monocular cameras were combined with other sensors such as depth cameras [13,14] and IMU [15–17] to achieve more robust performance.

However, these approaches are mostly used in hand-held or UAV (Unmanned Aerial Vehicle) applications, where the motions follow smooth, curved trajectories and frequently change the velocity. Thus, even with strong rotation, when there are few features to track in the image sequences, the IMU can still sense the motion well, which, thus, limits the pose estimation to an acceptable error [15,17].

On the other hand, ground robots have some unique advantages in rescue scenarios [18], compared to aerial robots, for example the long operation time, the rich sensor payload [19] and the ability to traverse in very confined areas. Hence, ground robots still play a critical role in search and rescue scenarios [6]. When rescue robots traverse difficult 3D terrain, the motions can be more complicated [4,20] than those of a UAV. There are very sudden accelerations when falling off a small edge or a stair step, or bumping into a small obstacle. This will result in strong ego-motions, which make it difficult for vSLAM and VIO (Visual Inertial Odometry) to track the features. Fusing the vision data with IMU readings is a direct idea to solve this problem. However, moving straight on a planar environment, the trajectories are mostly in constant acceleration and quite often constant in speed, which leads to a close to zero acceleration. In this case the VINS (Vision-aided INertial System) model suffers extra unobservable directions [21], which leads to inaccurate estimations.

This challenge becomes even tougher for tracked robots or small-sized robots. Besides above problem of unobservability, the IMU readings suffer from large amounts of noise. This is because the small, tracked robots, with a relatively small mass, get easily excited from small obstacles such as stones or sand, over which the robot drives, leading to big and frequent vibrations. An additional challenge rescue robots face is, that there are often high illumination changes and also, that environments can have few features, because everything is covered with dust [22]. So visual trackers for rescue robots face additional challenges.

In recent decades we have seen many innovative and improved sensor concepts, such as RGBD cameras, LIDAR and event cameras [23]. They are perceiving the environments using different approaches and integrate several different sensors on a single board thanks to the advances in MEMS (MicroElectroMechanical System) technology. Kinect and Kinect V2 are two very popular RGBD cameras using structured light and the latter ToF (Time of Flight) technology, respectively. They are both quite large and do not work outdoors, because the sunlight is over shining the infrared light emitted by the devices. LIDARs work outdoors and are quite accurate, but they deliver quite sparse data, are quite expensive and relatively fragile, due to the moving parts (mirror). In the last few years, Intel released their RealSense camera series [24], from R200 to ZR300 to D415/D435 [25] and other models. Some of them share the same market niche as Kinect, while others are facing robotic systems which are compact and relatively cheap. A big advantage of some of those devices, such as ZR300, is, that they come with an integrated IMU. For robotics, a challenging problem is the limited measurement range of those sensors, with a large minimum range and a small maximum. This significantly limits its application and thus many open source VIO or SLAM systems [15,17] only make use of the integrated fisheye camera and IMU sensor. Fortunately, Intel released the D435i [26] after the D415/D435. It integrates the same IMU as the ZR300, but uses the vision sensors of the D435, such that we can take advantage of D435's excellent range (0.2–10 m+), combined with the time-synchronized accelerometer and gyroscope measurements. Another advantage of the Intel RealSense sensor series is, that they also work outdoors in sunny conditions, due to their stereo-infrared camera approach [24,27,28], but also project an infrared pattern to be able to detect features-less surfaces.

The goal of our work is to enable a small rescue robot to do visual SLAM and thus ultimately navigation. We address the difficulties described above and propose to utilize the three different sensor readings of color, depth and IMU provided by the Intel RealSense D435i camera. For that, we base our solution, VINS-RGBD, on the open source VINS implementation [15], which is using RGB and IMU. We extend the framework to also incorporate the depth readings and ultimately build an RGBD inertia SLAM system for ground robots. The main contributions of our work are as follows:

- Formulation and implementation of a depth-integrated initialization process for the VINS-RGBD system.

- Formulation and implementation of a depth-integrated Visual Inertial Odometry (VIO), overcoming the degenerated cases of a vision and IMU only VIO system.
- Design and implementation of a backend mapping function to build dense point clouds with noise suppression, which is suitable for further map post processing and path planning.
- A color-depth-inertial dataset with handheld, wheeled robot, and tracked robot motion, with tracking system data for ground truth poses.
- Thorough evaluation of the proposed VINS-RGBD system using the three datasets mentioned above.

We provide the dataset [29], which also includes a video of the experiments, and VINS-RGBD [30] as open source for the benefit of the robotics and visual SLAM communities.

The rest of the paper is structured as follows: Section 2 describes important related work and the applicability towards VIO with ground robots. Section 3 gives an overview of the system which we base our work, VINS-Mono, and the observability problem it meets. Section 4 introduces our VINS-RGBD approach while Section 5 gives corresponding experiments and results. In Section 6 we draw conclusions and discuss future work.

2. Related Work

Compared to LIDAR Simultaneous Localization and Mapping (SLAM), which has under development for decades, visual SLAM (vSLAM) has recently gained more and more attention, especially after the influential work of Davison et al. [10], which showed that SLAM with a single camera can run in real-time on a PC. Extended Kalman Filter (EKF) or its variants are often used in LIDAR SLAM, and also in that first MonoSLAM [10]. Optimization based methods are becoming more popular, since they can find the sparsity of the Hessian matrix in the optimization function and can usually provide more accurate results. PTAM [31] was the first optimization based method and also the first one to propose to do tracking and mapping in parallel. Based on that tracking front end and optimization back end, approaches such as ORB-SLAM [12] were developed, which additionally use bag-of-words [32] to do loop detection to reduce the accumulative error. After projecting the environment to image planes, the depth information is lost and only the perspective geometry relationship is saved. Therefore, monocular SLAM is up-to-scale. However, combined with another camera with a fixed, known baseline, we can obtain the depth by calculating the disparity of sufficiently distinguishable feature points. Next to this stereo camera approach also structured light and Time of Flight (ToF) techniques, combined with a color camera, can also provide depth and color data, thus RGBD data. Visual SLAM with RGBD cameras [13,33] can be applied to robot autonomy and navigation. As both stereo and RGBD SLAM are using a single camera as the perception source, the performance is limited by camera characteristics such as motion blur under fast rotation, distortion caused by the rolling shutter, and also the short measurement range of the RGBD camera.

For ground robots quite often the trajectory estimation, so the localization of the robot, is assumed to be on the flat ground. We can thus estimate the trajectory under the 2D plane constraint, which accelerates the estimation speed and also improves the robustness with the motion model constraint. Scaramuzza [34] models the 2D motion to approximate planar circular motion, thus a 1-point minimum relative pose estimate solver can be applied. With a general planer motion model, an iterative 2-point algorithm [35] using the Newton method, a 3-point algorithm which is using a linear equation and non-iterative 2-point algorithms [36] were proposed.

Unlike Unmanned Ground Vehicles (UGVs), Unmanned Aerial Vehicles (UAVs), especially quad-copters, have to estimate their motion in 3D with 6 degrees of freedom. To avoid obstacles also the unknown scale factor has to be solved. So there is considerable work on vSLAM for UAVs where the cameras are combined with inertial measurement units (IMU) to achieve robust and scale deterministic pose estimation. Due to the complex motions of the UAV, the IMU is sufficiently activated and thus unobservable states are avoided. The visual and inertial fusion also called VINS (Visual INertial System). It is usually divided into the categories of loosely and tightly coupled systems, where the former [37,38] treats the two sensors as individual pose estimation sources and fuses them by using EKF. Tight coupled visual inertial fusion has both EKF based algorithms such as ROVIO [17], MSCKF [39] and optimization based algorithms such as VINS-Mono [15] and OKVIS [40]. Tightly coupled methods joint all the IMU and camera states together and usually get more accurate results. Most of the VINS are developed to deploy on UAVs or handheld applications. They only fuse a single camera with IMU, as this already gives good results in their application. However, in optimization based algorithms, it is easy to add other residual terms, hence, making it possible to fuse IMU with stereo or even more other sensors [41]. This can also solve the problem mentioned in [21], which is that the scale is unobservable under special motions. However, they only give the results of 2D performance, and also requite a complex configuration of the stereo camera with the IMU sensor, especially w.r.t time synchronization. Besides the unobservability problem, initial values of VINS are crucial for the subsequent VIO (Visual Inertial Odometry) process, and lots of work [42–44] has been put into achieving better performance, mostly for handheld or UAV applications.

Combining RGBD sensors with IMUs for visual SLAM is another way to solve the scale unobservability problem [21], which is also suitable for ground robots navigation. RGBD sensors calculate the depth independently and efficiently and also output dense depth images rapidly. An IMU RGBD extrinsic calibration is proposed in [45], where they analyze the observability and design an OC (Observability Constrained) EKF to improve consistency. A loosely coupled method, which fuses inertial and RGBD cameras for mobile devices is introduced in [46], where an array of Kalman filters is used. Another loosely coupled method is proposed in [47], with application in direct methods of frame-to-frame motion estimation. There, the performance of the IMU aid is evaluated in both semi-dense and fully dense tracking processes. Ref. [48] is the first tightly coupled optimization-based method in RGBD inertial fusion, which focuses on 3D reconstruction with running in real-time on a GPU, while [49] is another extended Kalman filter based indoor scene reconstruction RGBD-inertial method. By utilizing inertial fusion and kernel cross-correlators to match point clouds in the frequency domain, Wang et al. [50] propose a non-iterative odometry, where a dense indoor fused map is given. Ling et al. [51] simply combine Kinect depth images with ORB features to achieve indoor 2D robot pose estimation. However, in this paper only limited qualitative results are reported.

Three-dimesnional reconstruction differs from mapping for robotics mainly in the fact that robots are primarily interested whether a location is occupied or free, while 3D reconstruction cares more about the texture of the estimated shapes. In robotics the original point cloud map is then usually treated as an input to different kinds of post-processes, that are employed to make use of the map. Elevation maps [52] are 2.5D representations which store one height-value per cell in the x-y plane. Multi-Level surface maps [53] are another more powerful map representation that also supports loop closing. OctoMap [54] is a real 3D map representation based on octrees, which can be dynamically expanded with regard to global and local planning scenes. Yang et al. [55] propose an update strategy to OctoMap after loop closing, which builds a globally consistent map.

A benchmark comparison of the VINS algorithms is discussed in [56]. Although it is aiming for flying robots, the results show that VINS-Mono [15] achieves the most accurate performance while having a low computational resource requirement. Therefore we build our the RGBD inertia system for 3D pose estimation of ground robots on the open source VINS Implementation and call it VINS-RGBD. We also extend the VINS system to include a mapping module that is making use of the dense depth information from the RGBD sensor. This 3D map enables further applications such as path planning and navigation. Hence, we propose to generate noise-eliminated point clouds in an octree.

3. VINS-Mono Analysis

VINS-Mono [15] is a tightly coupled, nonlinear optimization based method consisting of a feature tracking module, a visual-inertial odometry which contains the necessary estimator initialization process for the IMU parameters and the metric scale initialization, and a sliding window optimization method which is taking the initialization value as the start value. Moreover, a pose graph module is taking care of relocalization (i.e., loop detection) and the subsequent global optimization. It is a state-of-the-art visual-inertial system, which has a good performance on UAVs and hand-held applications. The system structure is shown in Figure 1. The red solid line blocks represent the modules we modified to incorporate depth information, and the red dashed line blocks are the modules we newly added. Black blocks are original modules of VINS-Mono. We first introduce the original blocks as the base of our work.

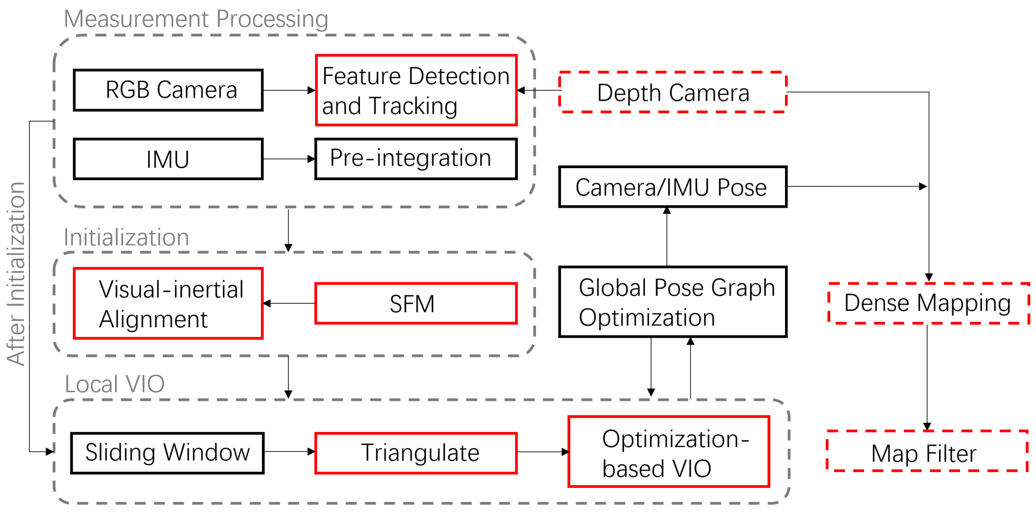


Figure 1. Core system structure. The solid line blocks indicate the original core pipeline of VINS-Mono. Red solid line blocks represent the modules of VINS-Mono that we modified. Red dashed line blocks represent the modules we newly added. VINS-Mono starts with the input of a color image sequence and IMU flow. IMU values are pre-integrated between every color frame, while visual features are detected and tracked. Then, the data is passed to SFM (Structure From Motion) to get an up-to-scale pose estimation. Once the visual and inertial alignment and initialization are done, the system starts to do a sliding window based local VIO process and is using the global pose graph optimization module to achieve the loop-closure task. The camera pose is then in a global consistent configuration.

3.1. Vision Front End

VINS-Mono supports both rolling and global shutter cameras. It is using the KLT sparse optical flow algorithm [57] to track Shi–Tomasi features [58]. The features are made to be uniformly distributed over the images to achieve a robust tracking performance. Keyframes are selected by checking the average parallax between the current frame and the latest keyframes, along with the number of tracked features. If the parallax is beyond a certain threshold or the number of tracked features is below a certain threshold, a new keyframe is inserted.

3.2. IMU Pre-Integration

An IMU typically includes three orthogonal axis accelerometers ($\mathbf{a}_t$) and a 3-axis gyroscope ($\boldsymbol{\omega}_t$), which measure acceleration and angular velocity with respect to an inertial frame. Affected by noise, IMU measurements commonly contains additive white noise $\boldsymbol{n}$ and time-varying bias $\boldsymbol{b}$:

$$\begin{aligned} \tilde{\mathbf{a}}_t &= \mathbf{a}_t + \mathbf{b}_{a_t} + \mathbf{R}_w^t \mathbf{g}^w + \mathbf{n}_a \\ \tilde{\boldsymbol{\omega}}_t &= \boldsymbol{\omega}_t + \mathbf{b}_{w_t} + \mathbf{n}_w \end{aligned} \tag{1}$$

As shown in Equation (1), accelerometers and gyroscopes suffer from noise that is usually modeled as Gaussian white noise,

$$\mathbf{n}_a \sim \mathcal{N}\left(\mathbf{0}, \sigma_a^2\right), \mathbf{n}_\omega \sim \mathcal{N}\left(\mathbf{0}, \sigma_\omega^2\right) \tag{2}$$

The bias is usually modeled as a random walk process [15,17,59].

$$\dot{\mathbf{b}}_{a_t} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2_{b_a}\right), \dot{\mathbf{b}}_{\omega_t} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2_{b_\omega}\right) \tag{3}$$

The accelerometer estimation also suffers the need to compensate the gravity vector $\boldsymbol{g}^w$ in world coordinate system. Earth's rotation is usually neglected, because it is very small compared to the noise mentioned above.

MEMS-IMUs typically have an accelerometer and gyroscope output rate of 200–400 Hz, while cameras usually have a frame rate around 30 Hz. Since they are in different orders of magnitude, i.e., there are about 10 IMU measurements between two camera frames, IMU measurements are usually pre-integrated between two visual frames. The pre-integrated values are treated as constraints between the two camera frames, which is used later during the local window optimization together with the camera reprojection residuals.

We are using b_k and b_{k+1} to indicate two consecutive camera frames, where k is the frame index. As mentioned above, there are a couple of IMU measurements between time t_k (at frame b_k) and time t_{k+1} (at frame b_{k+1}). The pre-integration is then defined as follows:

$$\begin{aligned}
\boldsymbol{\alpha}^{b_k}_{b_{k+1}} &= \iint_{t\in[t_k,t_{k+1}]} \mathbf{R}^{b_k}_t \cdot \mathbf{a}_t \, dt^2 \\
\boldsymbol{\beta}^{b_k}_{b_{k+1}} &= \int_{t\in[t_k,t_{k+1}]} \mathbf{R}^{b_k}_t \cdot \mathbf{a}_t \, dt \\
\gamma^{b_k}_{b_{k+1}} &= \int_{t\in[t_k,t_{k+1}]} \frac{1}{2}\boldsymbol{\Omega}\left(\boldsymbol{\omega}_t\right) \cdot \gamma^{b_k}_t dt
\end{aligned} \tag{4}$$

The three items in Equation (4) $\boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma$ are corresponding to translation, velocity and rotation, respectively. $\mathbf{a}_t$ and $\boldsymbol{\omega}_t$ are ideal values without noise and bias. Having $\boldsymbol{\omega}^\wedge$ as the skew symmetric matrix of a vector, $\boldsymbol{\Omega}\left(\cdot\right)$ is defined as follows:

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} -\boldsymbol{\omega}^\wedge & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix} \tag{5}$$

To save computational resources, the bias $\mathbf{b}_{a_t}$ and $\mathbf{b}_{\omega_t}$ are assumed to be constant during the pre-integration calculation between two consecutive camera frames. Then, the Jacobian matrixes with respect to every pre-integration items are calculated to correct the effect of bias changes. And if the estimated bias changes a lot, the pre-integration process is re-propagated under the new bias value.

3.3. Visual-Inertial Initialization and VIO

VINS-Mono tightly couples camera and IMU states in estimation. Also, as we have seen above, the IMU has a complex parameterization of the noise and bias. Thus, an initial process is necessary for parameter initialization of values such as gyroscope bias, gravity vector direction, and the metric scale for the up-to-scale mono visual SFM part. Once the initialization is done, the system can do the sliding window based local VIO process.

3.3.1. Visual-Inertial Initialization

The SfM problem can be solved up-to-scale using only visual information. If the extrinsic $\left(\mathbf{p}^b_c, \mathbf{q}^b_c\right)$, where $\left(\cdot\right)^b_c$ indicates translation (position $\mathbf{p}$) or rotation (quaternion $\mathbf{q}$) of the body/IMU frame with respect to the camera frame, is provided offline [60,61], then the IMU pose $\left(\mathbf{p}_{b_k}^{w}, \mathbf{q}_{b_k}^{w}\right)$ is also determined up-to-scale by a scalar s and camera pose $\left(\mathbf{p}_{c_k}^{w}, \mathbf{q}_{c_k}^{w}\right)$,

$$\begin{aligned} \mathbf{q}_{b_k}^{w} &= \mathbf{q}_{c_k}^{w} \otimes \left(\mathbf{q}_{c_k}^{b_k}\right)^{-1} \\ s\mathbf{p}_{b_k}^{w} &= s\mathbf{p}_{c_k}^{w} - \mathbf{R}_{b_k}^{w}\mathbf{p}_{c_k}^{b_k} \end{aligned} \tag{6}$$

Here, w is set to equal the first camera frame, and $\otimes$ represents quaternions multiplication.

Combining these states with the IMU pre-integration term γ, we can calibrate the gyroscope bias. We have all IMU orientations $\mathbf{q}_{b_k}^{w}$ and the pre-integration term γ in the window. Using $\mathcal{B}$ to represent all frame indexes in the window, the gyroscope bias is obtained by minimizing a least-square function:

$$\begin{aligned} \min_{\delta b_w} \sum_{k \in \mathcal{B}} & \left\| {\mathbf{q}_{b_{k+1}}^{w}}^{-1} \otimes \mathbf{q}_{b_k}^{w} \otimes \gamma_{b_{k+1}}^{b_k} \right\|^2 \\ \gamma_{b_{k+1}}^{b_k} &\approx \hat{\gamma}_{b_{k+1}}^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\mathbf{J}_{b_\omega}^{\gamma}\delta\mathbf{b}_\omega \end{bmatrix} \end{aligned} \tag{7}$$

Here $\mathbf{J}_{b_\omega}^{\gamma}$ represents the Jacobian matrix of the pre-integration term γ with respect to b_ω.

Combining the states with the IMU pre-integration terms α and β, the velocity, gravity vector, and scale can also be calculated by formulating a linear measurement model and solving this least square problem. We present this part in Section 4.2.

3.3.2. Visual-Inertial Odometry

Once the initialization is done, the sliding window based local VIO process is active to continuously estimate the state. With extrinsic transform between the camera and IMU $\left(\mathbf{p}_c^b, \mathbf{q}_c^b\right)$ calibrated offline, the state variables in the sliding window are defined as follows:

$$\begin{aligned} \mathcal{X} &= [\mathbf{x}_0, \mathbf{x}_1, \ldots \mathbf{x}_n, \rho_0, \rho_1, \ldots \rho_m] \\ \mathbf{x}_k &= \left[\mathbf{p}_{b_k}^{w}, \mathbf{v}_{b_k}^{w}, \mathbf{q}_{b_k}^{w}, \mathbf{b}_a, \mathbf{b}_\omega\right], k \in [0, n] \end{aligned} \tag{8}$$

n is the number of frames in the sliding window, $\mathbf{x}_k$ is IMU state synchronized with the kth camera frame, which contains velocity, orientation and position, acceleration and gyroscope bias with respect to world frame. $[\rho_1, \ldots \rho_m]$ represents the inverse depth from triangulation of all landmarks [62] in the frame of their first observation.

The IMU residual $\mathbf{r}_\mathcal{B}$ between two consecutive frame observations $\tilde{\mathbf{z}}_{b_{k+1}}^{b_k}$ and states $\mathcal{X}$ is then defined(denoted as symbol $\doteq$) as follows:

$$\mathbf{r}_\mathcal{B}\left(\tilde{\mathbf{z}}_{b_{k+1}}^{b_k}, \mathcal{X}\right) \doteq \begin{bmatrix} \delta\boldsymbol{\alpha}_{b_{k+1}}^{b_k} \\ \delta\boldsymbol{\beta}_{b_{k+1}}^{b_k} \\ \delta\boldsymbol{\theta}_{b_{k+1}}^{b_k} \\ \delta\mathbf{b}_a \\ \delta\mathbf{b}_\omega \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{w}^{b_k}\left(\frac{1}{2}\mathbf{g}^{w}\Delta t_k^2 - \mathbf{v}_{b_k}^{w}\Delta t_k + \mathbf{p}_{b_{k+1}}^{w} - \mathbf{p}_{b_k}^{w}\right) - \tilde{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} \\ \mathbf{R}_{w}^{b_k}\left(\mathbf{g}^{w}\Delta t_k + \mathbf{v}_{b_{k+1}}^{w} - \mathbf{v}_{b_k}^{w}\right) - \tilde{\boldsymbol{\beta}}_{b_{k+1}}^{b_k} \\ 2\left[{\mathbf{q}_{b_k}^{w}}^{-1} \otimes \mathbf{q}_{b_{k+1}}^{w} \otimes \left(\tilde{\gamma}_{b_{k+1}}^{b_k}\right)^{-1}\right]_{xyz} \\ \mathbf{b}_{a\,b_{k+1}} - \mathbf{b}_{a\,b_k} \\ \mathbf{b}_{\omega\,b_{k+1}} - \mathbf{b}_{\omega\,b_k} \end{bmatrix} \tag{9}$$

Here, $\delta\boldsymbol{\alpha}_{b_{k+1}}^{b_k}$, $\delta\boldsymbol{\beta}_{b_{k+1}}^{b_k}$ corresponds to translation and velocity IMU pre-integration residual terms, respectively. $\delta\boldsymbol{\theta}_{b_{k+1}}^{b_k}$ is in axis-angle representation and corresponds to the rotation pre-integration residual term and is also the rotation residual. $[\cdot]_{xyz}$ is the function that converts quaternions to the axis-angle vector representation.

The camera visual residual $\mathbf{r}_{\mathcal{C}}$ is defined as the sum of the reprojection errors of every landmark visible in the window. Knowing the extrinsic calibration $\left(\mathbf{p}_c^b, \mathbf{q}_c^b\right)$, the camera poses can be transformed to the IMU poses. Combined with all landmark depths in the state vector, it is easy to formulate the reprojection residual. During the sliding window based VIO process, new states are continuous added while old states are eliminated. Directly discarding the old states will cause a loss of accuracy, thus a marginalization strategy, implemented using the Schur complement [63], is used. The VIO optimization function containing the residuals of marginalization $\mathbf{r}_p$, IMU $\mathbf{r}_{\mathcal{B}}$ and camera $\mathbf{r}_{\mathcal{C}}$ is written as:

$$\min_{\mathcal{X}} \left\{ \left\|\mathbf{r}_p - \mathbf{H}_p \mathcal{X}\right\|^2 + \sum_{k \in \mathcal{B}} \left\| \mathbf{r}_{\mathcal{B}}\left(\tilde{\mathbf{z}}_{b_{k+1}}^{b_k}, \mathcal{X}\right)\right\| + \sum_{(l,k)\in\mathcal{C}} \rho\left(\left\|\mathbf{r}_{\mathcal{C}}\left(\tilde{\mathbf{z}}_l^{c_k}, \mathcal{X}\right)\right\|\right)\right\} \tag{10}$$

Here, $\mathbf{r}_p, \mathbf{H}_p$ are the constructed prior information coming from the marginalization. The pair (l, k) indicates the lth feature observed in the kth camera frame in the window. $\rho(\cdot)$ is the Huber norm loss function to reduce outlier influence.

3.4. Loop Detection and Pose Graph Optimization

As the sliding window based VIO is running, the error is unavoidably accumulated, due to all kinds of noise, such as IMU and camera measurement noise, calculation accuracy error, the residual error at the end of the optimization as well as errors induced by the marginalization process. Like most SLAM systems, VINS-Mono also incorporates a loop closure model, which is using a Bag of Words approach (DBoW2) [32] for loop detection. Once the loop is detected, the loop constraint is added together with other relative transformation constraints between sequential keyframes. Then, the pose graph optimization process is active to distribute the error on all poses. In normal applications such as handheld and UAVs, the scale is observable thanks to the accelerometer measurements. The roll and pitch angles are also observable from the IMU. The global pose graph optimization hence, only needs to perform in 4-DOF of x, y, z, and yaw.

3.5. Observability Analysis of VINS-Mono

For dynamic systems, observability is a fundamental property, and the nullspace of the observability matrix is corresponding to system's unobservable directions. More specifically, the rank of the nullspace is equal to the number of unobservable directions, while each unobservable direction has a corresponding physical meaning [64]. Besides, the observability is a system inherent property and hence, independent from the implementation. Since VINS-Mono is still using the normal VINS system modeling equations [64], it suffers from four unobservable directions, which are the 3-DOF global translation and the 1-DOF rotation around the gravity vector, which is the yaw angle. It is degenerated under special motions, which are frequently encountered with ground robots. For instance, when robots are moving at constant local acceleration,

$$\mathbf{a}_t^b \doteq \mathbf{R}_w^b \mathbf{a}_t^w \equiv \mathbf{a}^b, \quad \forall t \geq t_0 \tag{11}$$

such as a uniform linear motion of turning at a constant velocity, the system has another unobservable direction, which corresponds to scale. Note that the acceleration of the IMU is inherently measured in body frame, however to avoid ambiguity, we explicitly label the frame in Equation (11). To simplify the problem, we consider one state and change the corresponding visible landmarks expression in Equation (8) to contain x, y, z, denoted as f:

$$\mathcal{Y} = \left[\mathbf{p}_{b_k}^w, \mathbf{v}_{b_k}^w, \mathbf{q}_{b_k}^w, \mathbf{b}_a, \mathbf{b}_\omega, f_0, \dots f_l\right] \tag{12}$$

Thus, the corresponding up-to-scale state vector $\mathcal{Y}'$ can be written as:

$$\begin{aligned}
\mathbf{p}_{b_k}^{w\,\prime} &= s\mathbf{p}_{b_k}^{w} \;=\; \mathbf{p}_{b_k}^{w} + (s-1)\mathbf{p}_{b_k}^{w} \\
\mathbf{v}_{b_k}^{w\,\prime} &= s\mathbf{v}_{b_k}^{w} \;=\; \mathbf{v}_{b_k}^{w} + (s-1)\mathbf{v}_{b_k}^{w} \\
\mathbf{q}_{b_k}^{w\,\prime} &= \mathbf{q}_{b_k}^{w} \\
\mathbf{b}_a' &= \mathbf{b}_a - (s-1)\mathbf{b}_a \\
\mathbf{b}_\omega' &= \mathbf{b}_\omega \\
f_i' &= sf_i \;= f_i + (s-1)f_i,\; i = 0,\ldots l
\end{aligned} \tag{13}$$

where we use superscript $\prime$ to represent up-to-scale. The error state [64] of $\mathcal{Y}'$ is therefore:

$$\begin{bmatrix} \left[\delta\mathbf{q}_{b_k}^{w\,\prime}\right]_{xyz} \\ \delta\mathbf{b}_\omega' \\ \delta\mathbf{v}_{b_k}^{w\,\prime} \\ \delta\mathbf{b}_a' \\ \delta\mathbf{p}_{b_k}^{w\,\prime} \\ \delta f_0' \\ \vdots \\ \delta f_l' \end{bmatrix} = \begin{bmatrix} \left[\delta\mathbf{q}_{b_k}^{w}\right]_{xyz} \\ \delta\mathbf{b}_\omega \\ \delta\mathbf{v}_{b_k}^{w} + (s-1)\mathbf{v}_{b_k}^{w} \\ \delta\mathbf{b}_a - (s-1)\mathbf{a}_t \\ \delta\mathbf{p}_{b_k}^{w} + (s-1)\mathbf{p}_{b_k}^{w} \\ \delta f_0 + (s-1)f_0 \\ \vdots \\ \delta f_l + (s-1)f_l \end{bmatrix} = \begin{bmatrix} \left[\delta\mathbf{q}_{b_k}^{w}\right]_{xyz} \\ \delta\mathbf{b}_\omega \\ \delta\mathbf{v}_{b_k}^{w} \\ \delta\mathbf{b}_a \\ \delta\mathbf{p}_{b_k}^{w} \\ \delta f_0 \\ \vdots \\ \delta f_l \end{bmatrix} + (s-1)\begin{bmatrix} \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times1} \\ \mathbf{v}_{b_k}^{w} \\ -\mathbf{a}_t \\ \mathbf{p}_{b_k}^{w} \\ f_0 \\ \vdots \\ f_l \end{bmatrix} \tag{14}$$

After rewriting the equation, we get:

$$\delta\mathcal{Y}' = \delta\mathcal{Y} + (s-1)\mathbf{N}_s \tag{15}$$

It is shown in [65] that $\mathbf{N}_s$ lies in the nullspace of the observability matrix, under the condition that Equation (8) is satisfied. It is clear that the error state can be changed by arbitrary s with the same system measurements. Hence, the system is suffering scale unobservability under this special motion. The physical interpretation is also presented in [65]: The accelerometer bias can not be distinguished from true acceleration and since monocular VINS is relying on acceleration to achieve scale information, it is unobservable under this condition.

To intuitively show the constant local acceleration motion, we plot measured acceleration values under three different scenarios in Figure 2. The columns from left to right represent handheld, wheeled robot under circular motions and tracked robot under uniform linear motion and rows from top to bottom represent the x, y, z axes separately. The raw acceleration values are plotted in red. Just for visualization in Figure 2 we also show the result of a lowpass filter in blue for the wheeled and tracked robot data, which contain lots of noise. It is clear that in the handheld applications, the IMU is sufficiently excited, especially in x and y direction, while wheeled and tracked robots under circular and uniform linear motion have almost constant acceleration. Besides, they suffer considerably from noise, especially for the tracked robots. Although IMU-preintegration can overcome most noise since if it is zero mean, the monocular system still becomes unstable when the noise is quite large, which is the usual condition of tracked robots.

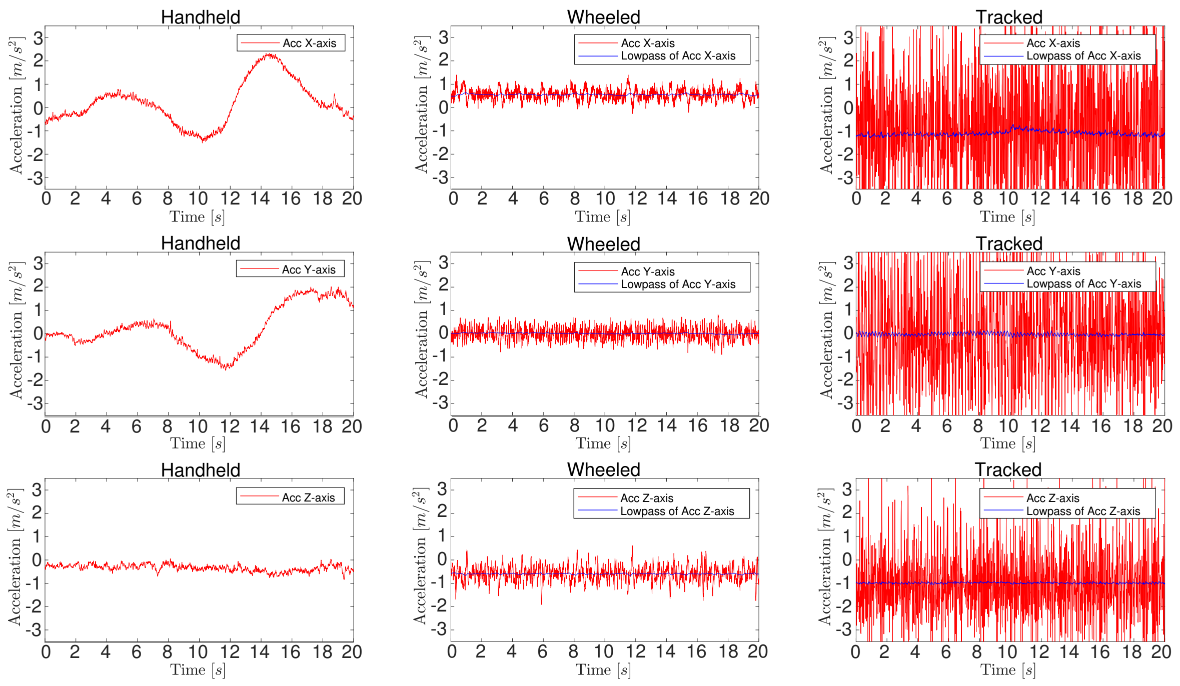


Figure 2. Acceleration values of handheld, wheeled robots and tracked robots shown separately. The left column is three-axis acceleration under normal handheld motion, the middle column is the acceleration of a wheeled robot which is driving a circular motion. The right column is the acceleration of a tracked robot which is under a uniform, linear motion. The red curves indicate measured raw acceleration values of the IMU sensors and the blue curves in the two right columns indicate the acceleration values after a lowpass filter.

On the other hand, when robots are performing no rotation motion

$$\mathbf{R}_w^{b_t} \equiv \mathbf{R}_w^b, \quad \forall t \geq t_0 \tag{16}$$

they suffer other unobservable directions, which makes all three global orientations unobservable. Wu et al. [21] developed mVINS (VINS within a Manifold), which incorporates VINS with manifold motions to solve the problem. Unfortunately, for rescue robots, the assumption is invalid due to the complexity of the terrain. Thanks to the loop closure model and our map filter, though it is left with an unobservable global orientation, we can still estimate a good trajectory and build nice maps.

4. VINS-RGBD

To address the scale unobservability problem described above and to improve the overall accuracy and robustness, we propose to incorporate the depth measurements of the RGBD sensor into the VINS-Mono system. We will be using the Realsense D435i RGBD sensor, which has an integrated IMU. Our algorithm contains a feature tracker node which is tracking features on the RGB image and gets the distance information from depth image. By utilize depth images, the system initialization gains another source of scale information, thus can be more robust and accurate. The VIO process is also working with the depth source to address the scale unobservable problem under constant local acceleration motion, but it also increases the accuracy for normal motions. Last but not least, we develop a mapping function which builds dense point clouds and eliminates noise at the same time.

4.1. Depth Estimation

The feature extracting and tracking method used as well as keyframe decision criteria are introduced as presented in Section 3.1. The tracked 2D features in the color images are combined with the pixel values in the same location of the depth images, which are aligned to the color frame.

Then, the 3D points together with their coordinates in the image frame are passed to the following initialization or VIO process. Since the D435i sensor is using stereo infrared cameras with additional structure light, and also due to the intrinsic characteristic of depth sensor, holes and some patterns (zero measurement) of structure light remain, although the depth images have been processed by the hole-filling filter RealSense provides. On the one hand, if a 3D feature is observed multiple times in different frames, we can reject zero and other incorrect depth measurements with our depth verification algorithm. Points which cannot achieve relatively accurate depth are further passed to the original VINS-Mono triangularization process, where their depth can be estimated, as long as they are observed in two different color frames.

4.2. System Initialization

After the tracking module, 3d points are extracted from each image with respect to the image frame which first observed them. VINS-Mono is using a window based optimization. We first need to set the value of the state vector, i.e., Equation (8). In VINS-Mono, since only the color source is available, the initial poses of the camera, hence, the IMU in window, are solved using the five-point algorithm [66]. The leads to the up-to-scale problem. Although the up-to-scale poses are then aligned to the IMU to achieve absolute scale, since IMU are always suffering noise, especially in ground robot applications and on low-cost IMU, the results of alignment are not always good and stable. With the depth source available in our VINS-RGBD approach, it is first feasible to perform the absolute scale pose initialization of cameras using PnP (Perspective-n-Point) algorithms [67–69]. The PNP problem is estimating the camera poses by using 2D features in the images and their corresponding 3D points in space. The difference of five-point algorithm and PNP algorithms is illustrated in Figure 3.

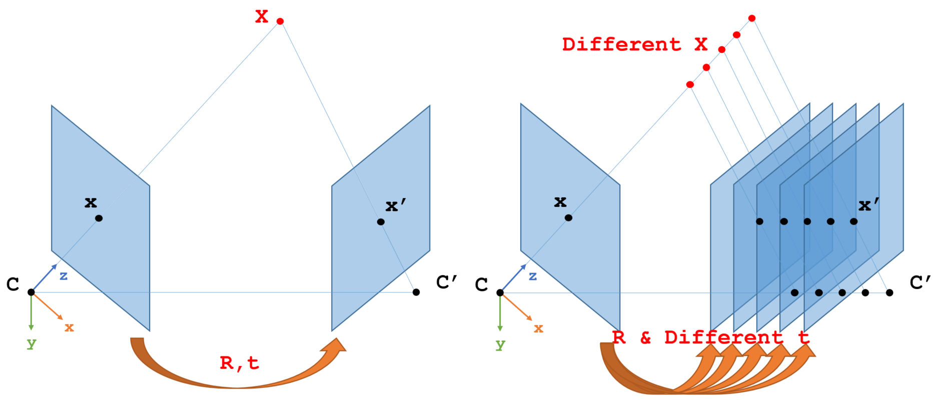


Figure 3. Illustration of 3D point projections on 2D images. C and C' are camera centers at different times with a rotation and translation R, t between them. X is a 3D point. X, R and t are unknown in monocular VINS. In the left image RGBD is used and the depth of the pixels is used to get X and PNP [67–69] is used to get R, t with constant scale factor. The right image illustrates the unknown scale factor when using just RGB. By having at least 5 point pairs like x and x', five-point algorithm [66] can be applied to get solutions.

Equation (6) is then rewritten without scale s,

$$\begin{aligned} \mathbf{q}^w_{b_k} &= \mathbf{q}^w_{c_k} \otimes \left(\mathbf{q}^{b_k}_{c_k}\right)^{-1} \\ \mathbf{p}^w_{b_k} &= \mathbf{p}^w_{c_k} - \mathbf{R}^w_{b_k}\mathbf{p}^{b_k}_{c_k} \end{aligned} \tag{17}$$

As shown in Equation (7), by getting the IMU pre-integration term γ, the gyroscope bias is calibrated initially. Since the scale is known, later parameters initialization only contains velocities and gravity vector, which can be calculated by minimizing terms $\delta\boldsymbol{\alpha}^{b_k}_{b_{k+1}}, \delta\boldsymbol{\beta}^{b_k}_{b_{k+1}}$ of Equation (9) in window with respect to variables $\mathbf{v}^w_{b_k}, \mathbf{g}^w$. Thus, the state vector can be defined as:

$$\mathcal{X}_{k_{init}} = \left[\mathbf{v}^w_{b_k}, \mathbf{v}^w_{b_{k+1}}, \mathbf{g}^w\right], k \in [0, n-1] \tag{18}$$

The linear measurement model is:

$$\tilde{\mathbf{z}}_k = \begin{bmatrix} \tilde{\boldsymbol{\alpha}}^{b_k}_{b_{k+1}} - \mathbf{R}^{b_k}_w \left(\mathbf{p}^w_{b_{k+1}} - \mathbf{p}^w_{b_k}\right) \\ \tilde{\boldsymbol{\beta}}^{b_k}_{b_{k+1}} \end{bmatrix} = \begin{bmatrix} -\mathbf{R}^{b_k}_w \Delta t_k & \mathbf{0} & \frac{1}{2}\mathbf{R}^{b_k}_w \Delta {t_k}^2 \\ -\mathbf{R}^{b_k}_w & \mathbf{R}^{b_k}_w & \mathbf{R}^{b_k}_w \Delta t_k \end{bmatrix} \doteq \mathbf{H}_k {\mathcal{X}_{k_{init}}}^T + \mathbf{n}_k \tag{19}$$

The linear least square problem is then defined by adding all the residual terms:

$$\min_{\mathcal{X}_{all_{init}}} \sum_{k \in [0, n-1]} \left\| \tilde{\mathbf{z}}_k - \mathbf{H}_k {\mathcal{X}_{k_{init}}}^T \right\|^2 \tag{20}$$

After solving the least square problem, another gravity refinement approach is applied as in original VINS-Mono, since the magnitude of the gravity is always known. By taking advantage of this information, the gravity vector can be modelled as a unit vector plus two DOF perturbation in its tangent space. Note that the acceleration bias does not include in the initialization process, since the gravity vector is a few orders of magnitude larger than the bias, which is making it hard to recover. This affects the scale recovery to a certain degree in VINS-Mono, however, has less influence on our VINS-RGBD.

4.3. Depth-Integrated VIO

The depth-integrated VIO process takes over the image streams once the initialization is done. The process start from the last state $\mathcal{X}$ in Equation (8). Then, the window slides once to marginalize the oldest or latest frame based on a parallax calculation, where keyframes are also decided. The next new $\mathbf{x}_k$ is propagated by IMU integration and then added to the window. Based on all states in the window, in VINS-Mono points which do not have depth are triangulated to achieve depth. In our implementation, the depth is attached in each observation frame of a feature. Thus, a consistent depth verification Algorithm 1 can be applied to filter noisy depth measurements. For the points which are beyond the depth sensor's range, the original triangularization method is used. Note that the sensor's range does not need to be equal to device real range. It can be manually set by the user. This way the user can compensate for the increased error for faraway points in RGBD sensors.

Once we have all features in the window with their verified depth, the camera visual residual, i.e., the reprojection error, can be calculated. With IMU residual and marginalization prior in Equation (10), the Ceres non-linear optimization solver [70] is used to solve the problem. However, for low-cost IMUs such as Realsense D435i is using, noise and bias are in bad condition, which violates the modelling in Equation (1). Hence, we set the features whose depth come from depth verification algorithms as constant while leaving the features whose depth come from triangularization still as optimization variables, as Figure 4 shows.

Algorithm 1: Depth verification algorithm.

```
Data: $\mathcal{T}$: transformation between observation frames,
      $\mathcal{P}$: set of all features within observation frames
      $\mathcal{F}$: set of observation frames containing 2D points and depth
      $\mathcal{C}$: verified depths set
Result: verified consistent depth in reference frames
initialization;
for feature $f$ in $\mathcal{P}$ do
    if $f$ has no depth then
        $\mathcal{C} \leftarrow \emptyset$;
        for $frame \in \mathcal{F}$ with $f \in frame$ do
            for $frame' \in \mathcal{F} \setminus frame$ with $f \in frame'$ do
                project 3D point $p$ of $f$ in $frame$ to 2D image point of $f$ in $frame'$;
                calculate projection residual;
                if $residual < threshold$ then
                    transform $p$ to reference frame;
                    get depth $d$ of transformed point;
                    $\mathcal{C} \leftarrow d$;
                end
            end
        end
        calculate mean $d_{mean}$ in $\mathcal{C}$;
        if $d_{mean}$ between device range then
            set depth for $f$ as $d_{mean}$
        end
    end
end
```

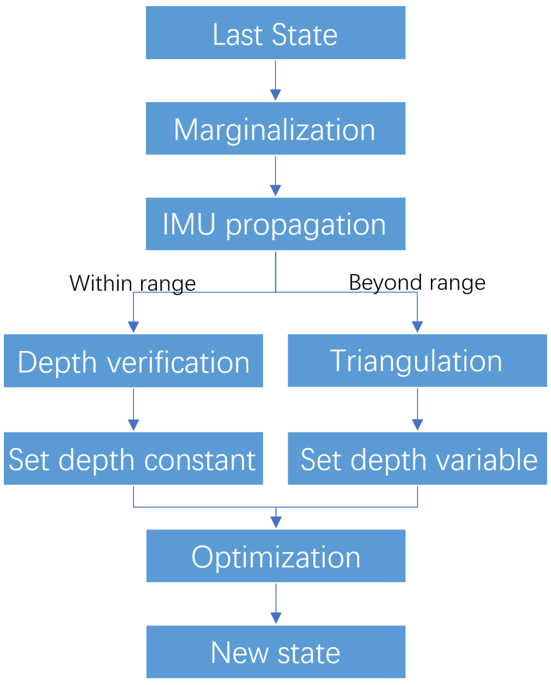


Figure 4. Depth integrated VIO process.

4.4. Loop Closing

After detecting a loop using the Bag of Words approach, VINS-Mono is estimating the transform between the two frames using a PnP approach on the 3D points estimated by triangulation. Thus, our VINS-RGBD is automatically also improving the loop closing transform estimation, since the used depths are now provided by the more reliable RGBD sensor.

4.5. Mapping and Noise Elimination

Different from Kinect Fusion [14], which uses GPU accelerated ICP [71] for finding matched points between two depth images, to make the system compact and suitable for small robot integration, we simply project 3D points from each depth image, which are synchronized with keyframes with their corresponding poses estimated by VINS, to build the map. With this straightforward method, however, it does not match corresponding points among frames, resulting in a lot of redundant points.

In order to better manage the point clouds, an octree [72] structure is maintained. Octree is a tree-like data structure used to describe three-dimensional space, each node of which represents a volume element of a cube. A parent cube can be at least cut into eight equal child cubes. Such the number of child nodes of any node in the tree cannot be anything but eight or zero. The volume elements represented by the eight child nodes are added together to be equal to the volume of the parent node. The resolution defines the size of the smallest voxel. A basic structure of octree is as shown in Figure 5.

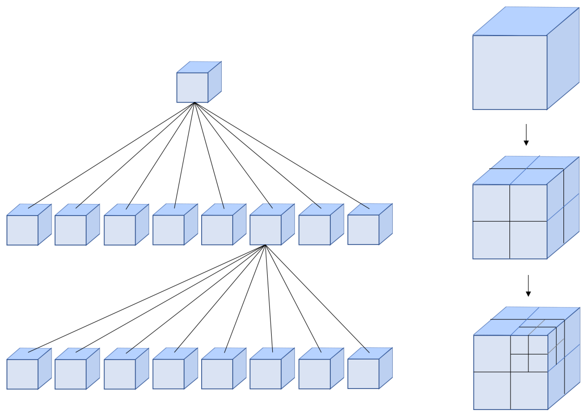
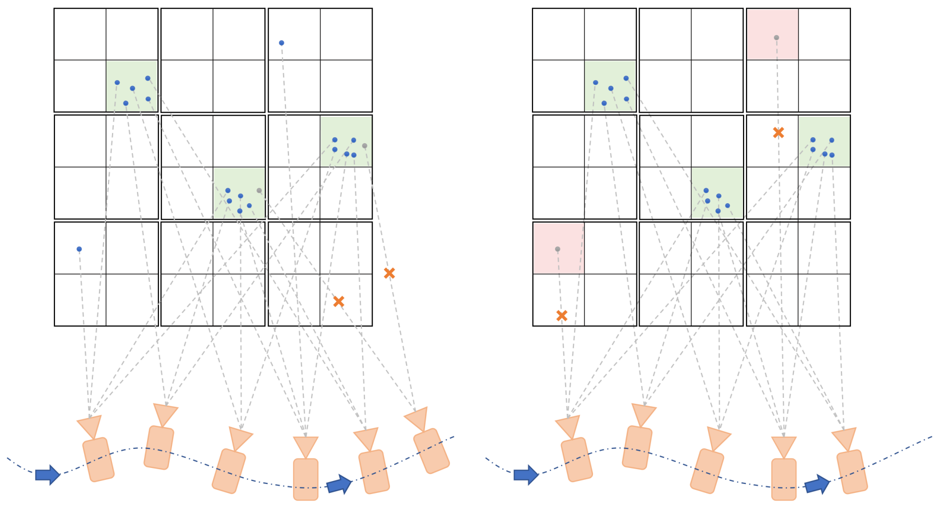


Figure 5. Octree Structure.

To achieve real-time low-cost mapping and navigation, we first downsample every depth image with step size 10 to get around 3000 points, while the original depth image contains more than 300,000 points at a resolution of 640 $\times$ 480. Note that these points are inherently different from the feature points that the tracking thread provides. They are able to provide dense information, which is useful in navigation tasks, especially in the ground robot application, since they are not only doing obstacle avoidance but also interaction with obstacles in environments such as going over stairs.

After downsampling, we set the upper bound of each leaf node in the octree to only contain five map points. Any extra points coming later are rejected and also removed from the point vector attached on each keyframe, as Figure 6a shows. This may loss information, however, the resolution of the octree can always set high enough to satisfy the application demand. Also note that we are not aiming to do a 3D reconstruction, but to build a map applicable to ground robots, especially for rescue robot navigation.



(**a**) Eliminate points in VIO process. (**b**) Eliminate points after loop closure.

Figure 6. Points elimination. (**a**) illustrates the elimination during the VIO process. Cameras in figures represent keyframes. The points (blue) are continuously added to the octree leaf nodes until reaching the upper bound. Later coming points (gray) are rejected and also removed from point vector of corresponding keyframe to save space. After loop closure, as illustrated in (**b**), if a leaf node contains fewer points than a given threshold, the points are discarded, both in the octree and point vector of the keyframe.

On the other hand, when a loop is detected and closed, we need to update all the points after loop closure, where the time consumption is linearly related to the number of points. Setting the upper bound of the capacity of each leaf node significantly reduces the number of newly added points, ensuring the performance of real-time mapping. However, the loop closure algorithms minimize the error and thus may change all poses in the loop circle. There might still exist a few poses which have a significant error. The 3D points of those frames ideally should not be projected to the map. To achieve this we set a threshold on the number of points in a leaf: Points in leaves containing a fewer number than the threshold points are discarded. The threshold is depending on the velocity, frame rate and keyframe selection strategy. For instance, if the robots move fast, it may occur that most of the numbers of points in octree leaf nodes are less than the threshold and be discarded incorrectly. Besides, the depth image provided by the depth camera has noise which considered as random. Our elimination strategy can also discard them, thus leading to a more consistent map. The elimination process is shown in Figure 6b.

5. Experiments and Results

To our knowledge, there is no public dataset which contains color, depth and IMU data together. The EuRoC MAV visual-inertial datasets [73] contains stereo images rather than dense depth images directly. Besides, they are also aiming to MAV (Micro Aerial Vehicle) applications. The RGBD SLAM Datasets [74] provided by TUM contains RGBD images; however, they only have accelerometer data. To this end, the experiments are all evaluated in real-world environments compared to the original VINS-Mono by using a RealSense D435i camera. The datasets are also attached with our open source code. To comprehensively evaluate our system, we perform both single and integrated experiments. The first single experiment evaluates the scale drift problem during local constant acceleration motion. The second single experiment validates our depth-integrated VIO process illustrated in Figure 4. The integrated experiments with handheld, wheeled and tracked robots are then performed to show the versatility of our system. Last but not least, the mapping results are shown in different scenarios. The experiments are all running on an Intel NUC equipped with i7-8559U CPU at 2.7–4.5 GHz and 16 GB memory; no GPU is used. The ground truth poses in experiments are provided by our OptiTrack [75] tracking system and we evaluate the trajectories using the methods provided by [76].

5.1. Scale Drift Experiment

The scale drift experiment is performed on a Jackal [77] UGV, driving at a constant radius circle. When the monocular VINS system is under constant radius circular motion (radius about 2 m, speed about 0.4 m/s), the scale direction is unobservable, as we showed in Section 3.5. However, combined with absolute depth information, another scale resource is integrated into the system. Figure 7 illustrates our VINS-RGBD compared to VINS-Mono with ground truth. For comparison we also tested both systems with deactivated loop closing (no loop), showing the performance of only the pure VIO. Our system is well aligned with the ground truth in both loop and no loop conditions. Since RealSense D435i is using a low-cost IMU, the monocular VINS drifts a lot as shown in Figure 8. Actually, VINS-Mono initialized with a wrong scale, which can be seen in Figure 7b.

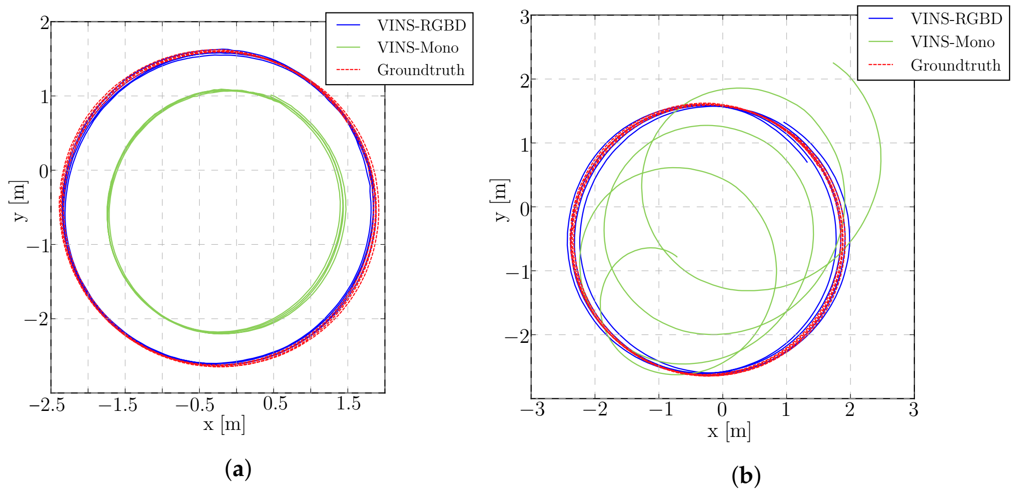


Figure 7. Scale drift evaluation with loop closure (**a**) and with deactivated loop closing (**b**). The trajectories are produced by a wheeled Jackal UGV.

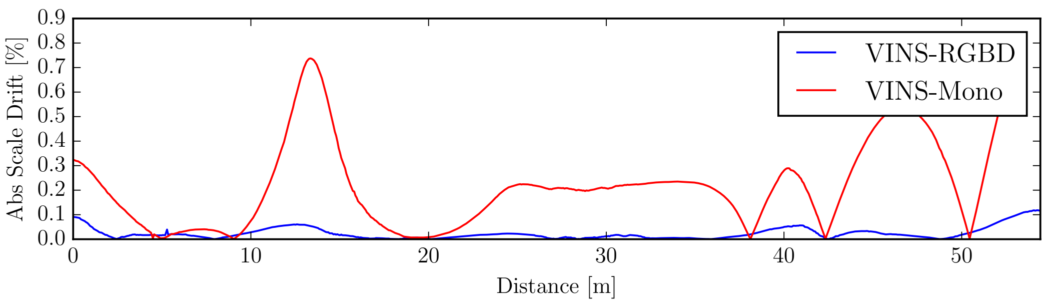
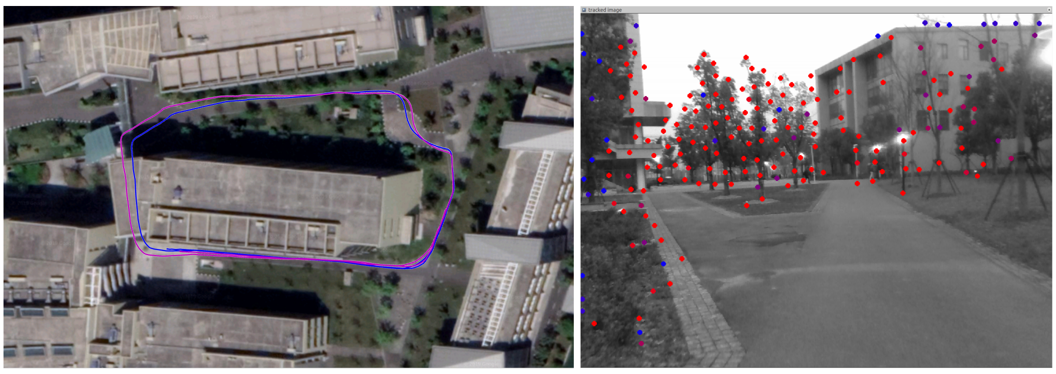


Figure 8. Absolute scale drift percentage of the deactivated loop closing experiment.

5.2. Open Outdoor Experiment

RGBD cameras usually use structured light or ToF (Time of Flight) technology to achieve dense depth images, which also limits the measurement range to several meters. Also, since they are active infrared sensors, they usually get over shined by the sunlight. For those reasons almost all handheld and UAV applications use color only VINS. RealSense D400 series is using a stereo approach with structure light as backup for uniform surfaces and has a maximum range up to 10 m. The error also increases exponentially with distance. In this paper we proposed to use a verified depth within a certain range and use the triangularization method to estimate points beyond depth measurement range, as shown in Figure 4. We evaluate this part in a handheld open outdoor scenario which shown in Figure 9b. The trajectories estimated by VINS-Mono and our system aligned to Google Maps are shown in Figure 9a. We can see the performance of both algorithms is similar, which validates that our system does not degenerate in outdoor and out-of-range scenarios.



(**a**) (**b**)

Figure 9. Open outdoor experiment. (**b**) shows a representative moment when our system is running, where the points are far away and most of them are beyond depth range of the device. The color in more red indicates the longer of tracking times. (**a**) shows the trajectories aligned to Google Maps. The blue one is estimated by our system, and the purple one is from VINS-Mono.

5.3. Integrated Experiments

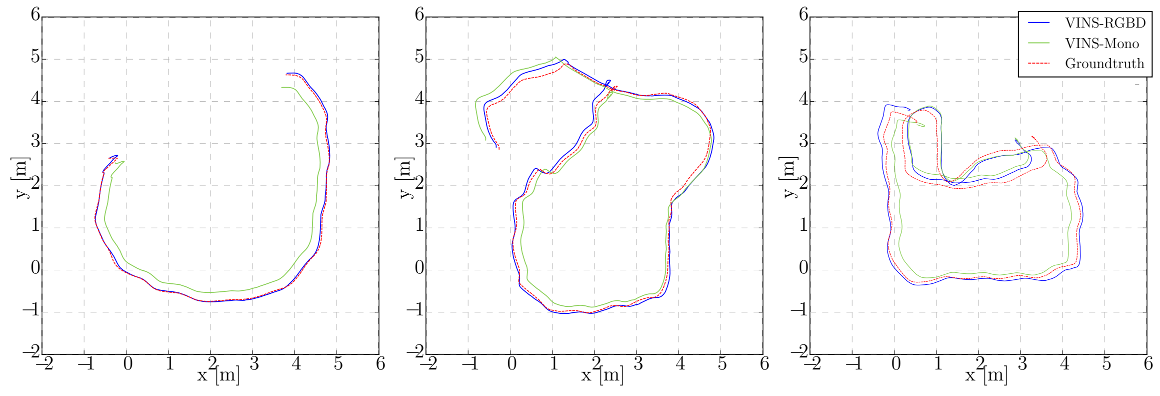
The integrated experiments cover handheld, wheeled robot and tracked robot tests. We evaluate the trajectories of our proposed VINS-RGBD and VINS-Mono with the ground truth poses from the tracking system. The RMSE (Root-Mean-Square Error) is used to evaluate the absolute error after aligning every trajectory to its corresponding ground truth path. The absolute translation and orientation are also calculated to compare. Besides, the relative pose error is shown to give a more comprehensively evaluation. To highlight the comparisons between the two methods, we disable the loop detection and optimization in the following tests, however, loop performance is also evaluated in the form of RMSE in Table 1.

Table 1. RMSE of three different experiments in meters. "-" denotes no loop, "x" denotes test failed.

Experiment	VINS-RGBD	VINS-RGBD with loop	VINS-Mono	VINS-Mono with loop
Handheld Simple	**0.07**	-	0.24	-
Handheld Normal	**0.13**	-	0.20	-
Handheld With more Rotation	**0.18**	0.20	0.23	0.23
Wheeled Slow	0.20	**0.16**	0.39	0.27
Wheeled Normal	0.17	**0.09**	0.18	**0.09**
Wheeled Fast	0.23	**0.20**	0.63	0.31
Tracked 1 Ground and Up-down Slopes	0.11	**0.10**	x	x
Tracked 2 Cross and Up-down Slopes	**0.17**	0.23	x	x
Tracked 3 Cross and Up-down Slopes and Ground	**0.21**	0.24	0.77	0.75

5.3.1. Handheld and Wheeled Experiment

To show the versatility of our system, we perform handheld experiments with RealSense D435i under different motions. The trajectories are shown in Figure 10 and the absolute and relative error of the test "With more rotation" is shown in Figure 11. It is shown that with depth information integrated, the handheld application, which is like UAVs, also has better performance compared to the original VINS-Mono. In Figure 10a,c, the scale evaluation of VINS-Mono method is in bad condition, however, our method not only recovers scale more accurate but also has better performance in relative pose estimation, which is shown in relative error part of Figure 11.



(**a**) Simple (**b**) Normal (**c**) With more rotation

Figure 10. Handheld trajectory estimation.

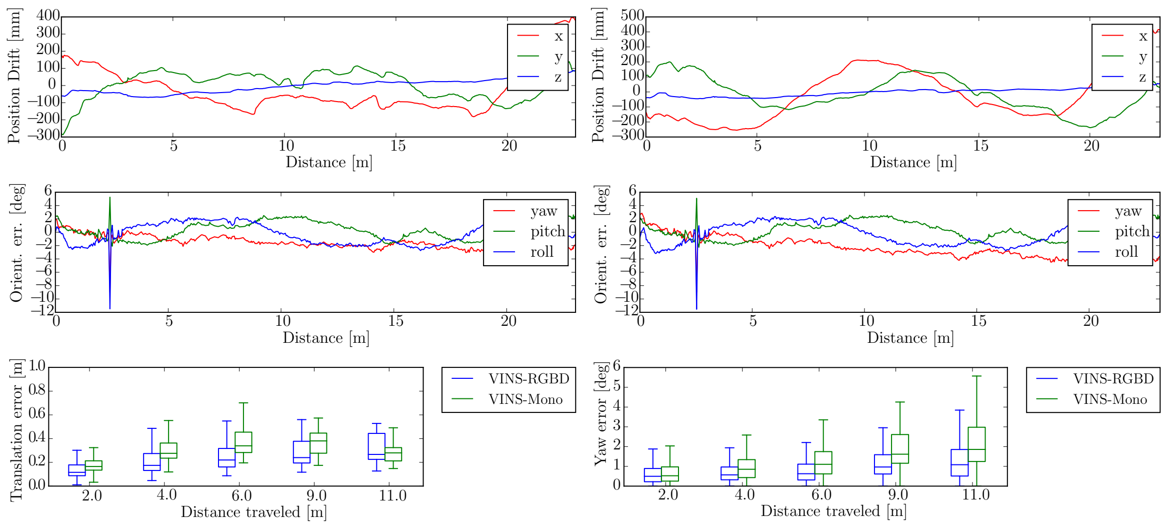
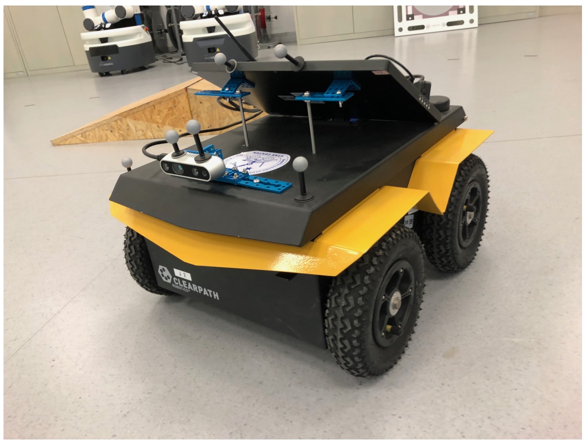


Figure 11. Handheld Absolute and Relative Error in "With more rotation" Experiment. The first two rows show the absolute position and orientation error, where the left results are from our VINS-RGBD and right are from VINS-Mono. The last row shows the relative translation and yaw error of our VINS-RGBD and VINS-Mono, respectively.

The wheeled robot hardware platform is shown in Figure 12a, which is a commercial wheeled ground vehicle with differential drive. The wheeled acceleration reading of IMU in Figure 2 is from this platform, which is shown that small sized wheeled robots usually suffer larger acceleration noise than handheld applications. This is due to the mechanism of such robots, because large vehicles such as passenger vehicles usually have larger wheels which can absorb more shock and independent shock mitigation systems that small-sized robots do not have. In the rescue field, small sized wheeled robots are also used sometimes in easy terrain. Figures 13 and 14 show the trajectories and error comparison of our wheeled robot. The trajectories are from the robot driving on the ground with two up-down slopes with different velocities (average speeds: slow: 31 min 148 s; normal: 32 min 74 s; fast: 48 min 99 s (with intermittent high speed phases)). Our VINS-RGBD aligned to ground truth better, especially in slow and fast experiments. In the normal velocity test, both methods have good performance, which may be caused by the sufficiently activated accelerometer under this motion. However, with faster motion, which is shown in Figure 13c, since more motion blur from the color camera is introduced, the performance degenerates. In the relative error diagrams of Figure 14, the relative translation error performance is much better in our method while relative yaw does not have much difference, which is as was expected since, adding depth information into VINS has a large effect on translation evaluation but little on orientation.


(**a**) Wheeled Robot. A ramp is visible in the background.

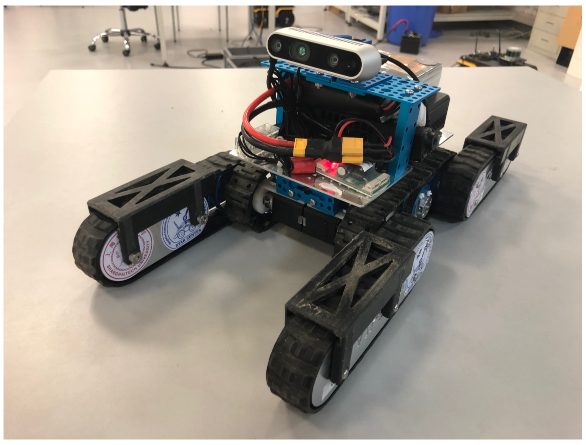
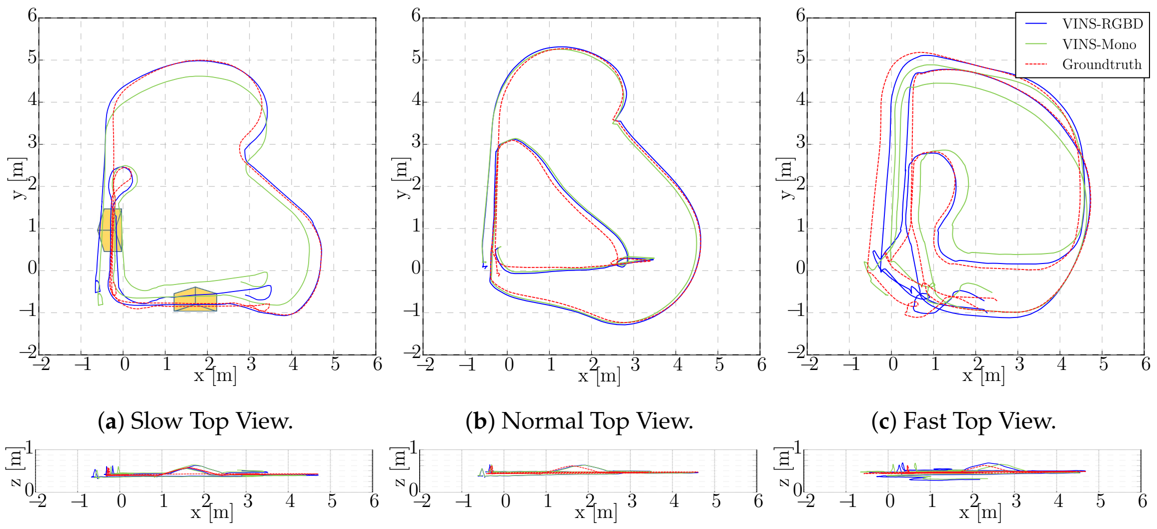

(**b**) Tracked Robot.

Figure 12. Robot Hardware Platform.



(**a**) Slow Top View. (**b**) Normal Top View. (**c**) Fast Top View.

(**d**) Slow Side View. (**e**) Normal Side View. (**f**) Fast Side View.

Figure 13. Wheeled Robot trajectory estimation. Driving on flat ground with two ramps (approximate position indicated in (**a**). (**b**,**c**) are similar).

5.3.2. Tracked Experiment

The self-developed tracked robot hardware platform is shown in Figure 12b, which has a typical operation size of 0.4 $\times$ 0.3 $\times$ 0.4 m, and four independent flippers for obstacle crossing. Since the size is smaller than regular rescue robots, there is lots of shaking caused by tracks during the movement, which causes pose estimation to be more difficult, as Figure 2 shows. Besides, the movement in 3D scenes usually contains sudden falls from obstacles, and fast rotations, which causes strong ego-motion and makes the camera only SLAM such as ORB-SLAM2 fail, as shown in Figure 15. We perform the tracked robot experiments in three different tests: Ground and Up-down Slopes; Cross and Up-down Slopes; Cross and Up-down Slopes and Ground. The schematic trajectories are shown in Figure 16, which is the resulting map from our method under the handheld approach, with light blue, green and red colors, respectively. The three trajectories evaluations are shown in Figures 17 and 18. As can be seen in Figure 17, the trajectories are flexuose even on the ground truth. The performance degenerated due to this factor, where VINS-Mono does not work in the first two tests and also hardly works in the last test. Our method has a larger error than previous handheld and wheeled tests. However, it is still at an acceptable level. The RMSE of all the nine experiments is listed in Table 1, which shows that with depth-integration, our method has a better performance. Figure 19 shows representative selected frames in handheld, wheeled and tracked robots, respectively.

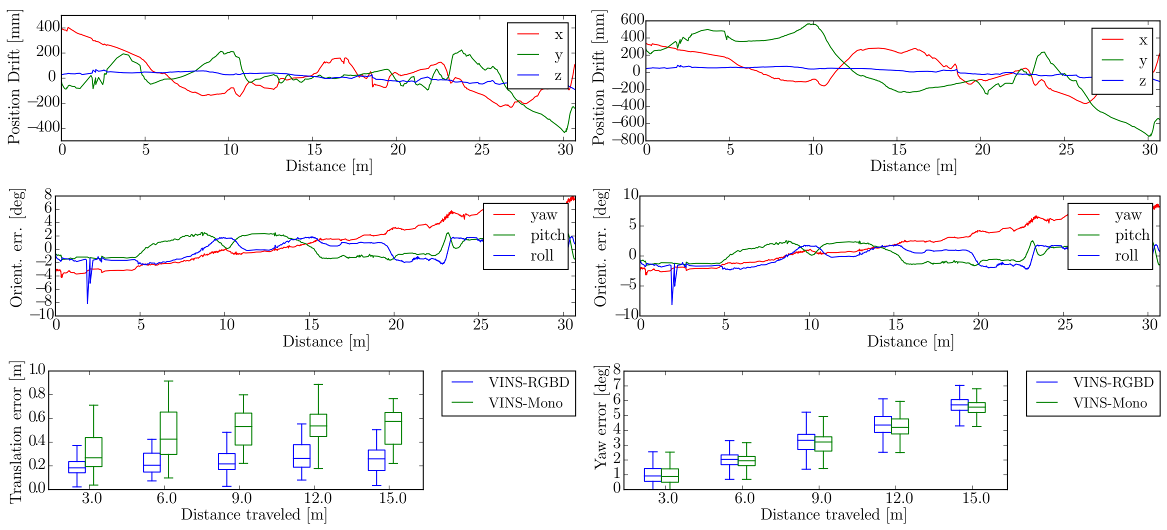


Figure 14. Wheeled Robot Absolute and Relative Error in "Slow" Experiment. The first two rows show the absolute position and orientation error, where the left results are from our VINS-RGBD and right are from VINS-Mono. The last row shows the relative translation and yaw error of our VINS-RGBD and VINS-Mono respectively.

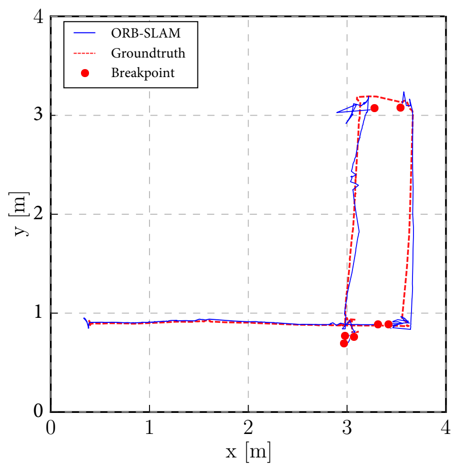


Figure 15. ORB-SLAM in Tracked 2 (Cross and Up-down Slopes) test. It lost track three times. We reset the ORB-SLAM once it lost track and get four individual trajectories. The red points indicate the beginning and end of four trajectories. Note that since we align trajectories separately, the aligned result could look good, which should not be treated as the real performance of this method.

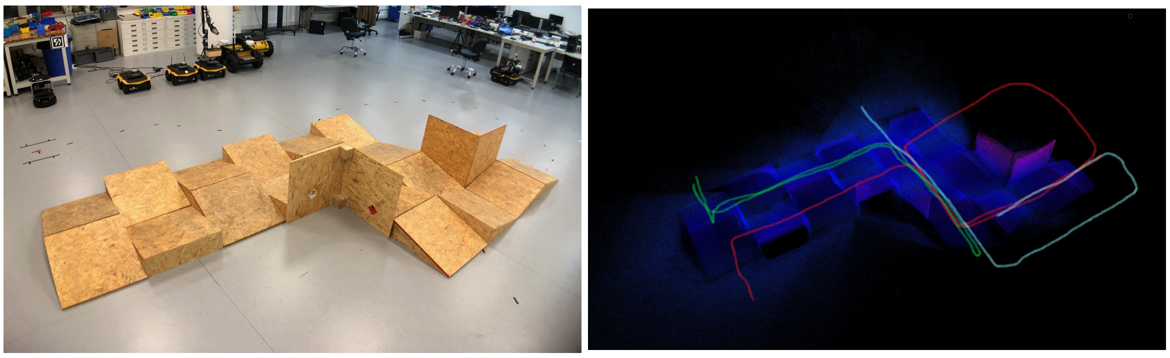


Figure 16. Tracked robot experiment site. (**a**) Real Trajectory Testing Environment; (**b**) Hand-drawn trajectories of the tracked robot experiments. Tracked 1: light-blue; Tracked 2: green; Tracked 3: red.

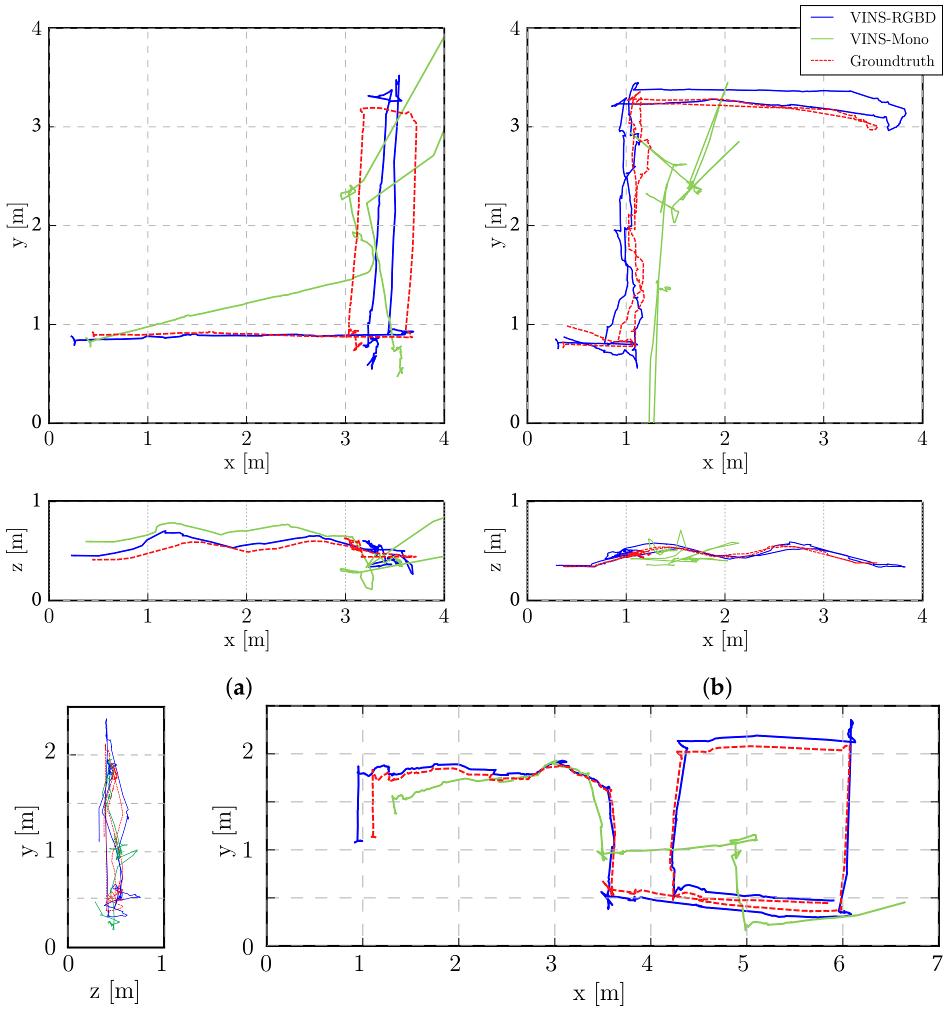


Figure 17. Tracked robot trajectory estimations. Start point on the left for all three experiments. (**a**) Tracked 1: Ground and Up-down Slopes. Top View (top), Side View (bottom); (**b**) Tracked 2: Cross and Up-down Slopes. Top View (top), Side View (bottom); (**c**) Tracked 3: Cross and Up-down Slopes and Ground. Top View (right), Side View (left).

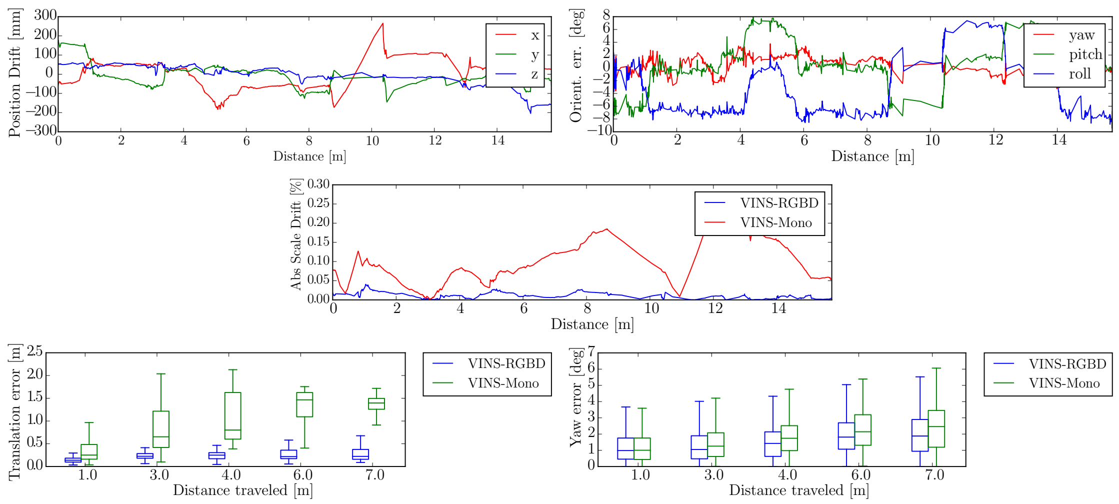
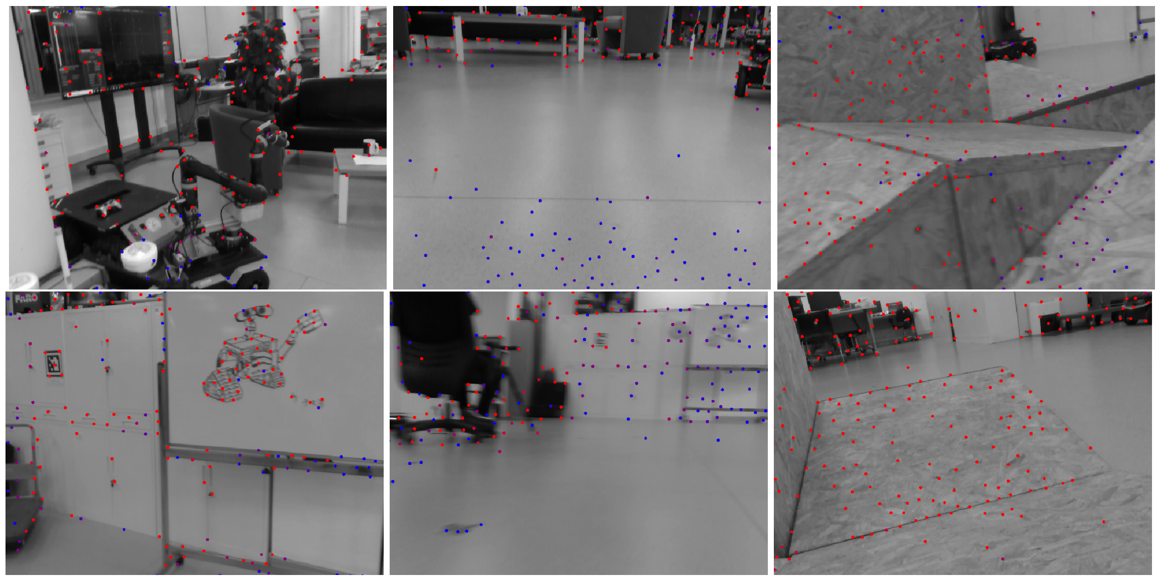


Figure 18. Tracked Robot Absolute and Relative Error in the "Cross and Up-down Slopes and Ground" experiment. The first row shows the absolute position and orientation error of our system. The relative translation and yaw error together with the scale drift percentage comparison of our VINS-RGBD and VINS-Mono are shown in the last two rows.



(**a**) Handheld (**b**) Wheeled (**c**) Tracked

Figure 19. Selected frames from the handheld, wheeled and tracked experiments.

5.4. Map Comparison

In Section 4.5 we proposed to bound the point clouds by eliminating points using octree during the VIO process. The resulting point clouds are further filtered if an octree leaf node contains fewer points than another threshold. This strategy can build maps with different density point clouds, which also accelerates the loop closing speed.

The experiments are performed a 60 cm version of the ASTM Standard Test Method for Evaluating Emergency Response Robot Capabilities. Those are ASTM E2803-11 Confined Area Obstacles: Inclined Planes and ASTM 2827-11 Crossing Pitch/Roll Ramps [78], the first test methods with the rails is still in the standardization process. Those test methods are also extensively used in RoboCup Rescue [79,80], for which we plan to use VINS-RGBD.

Figure 20 shows the results of our VINS-RGBD mapping. In every column, there is one of the test methods. The rows with the maps show the mapping with all subsampled points on the top, the maps with the upper limit of five points in an octree cell in the middle and the maps where additionally points in cells with fewer than three points are filtered in the bottom. Note that the last filter step is triggered manually here to keep it consistent with the above maps. Comparing the third rows to the second rows, we can see that the map structure remains very similar while the point density is reduced. The last rows eliminate points if fewer than a threshold of 3 are in a cell. We can see that the point clouds is further reduced, mainly in areas which are of little interest. To quantitatively evaluate the efficiency of the octree structure based strategy, we perform experiments in three different size scenes: lab (large), arena (middle), and arena (small) in Table 2. The corresponding map results are shown in Figure 21. The experiments evaluate the upper bound number during the VIO process, filtering after loop closing is disabled to keep the results consistent. We can see with the upper bound number decreasing, the number of points is also reduced near linearly. The last two columns list the update time after loop closure with an octree leaf capacity at 5 and the speed up rate compared to origin points. This is calculated by points rate, since the points update time is linearly related to the number of points.

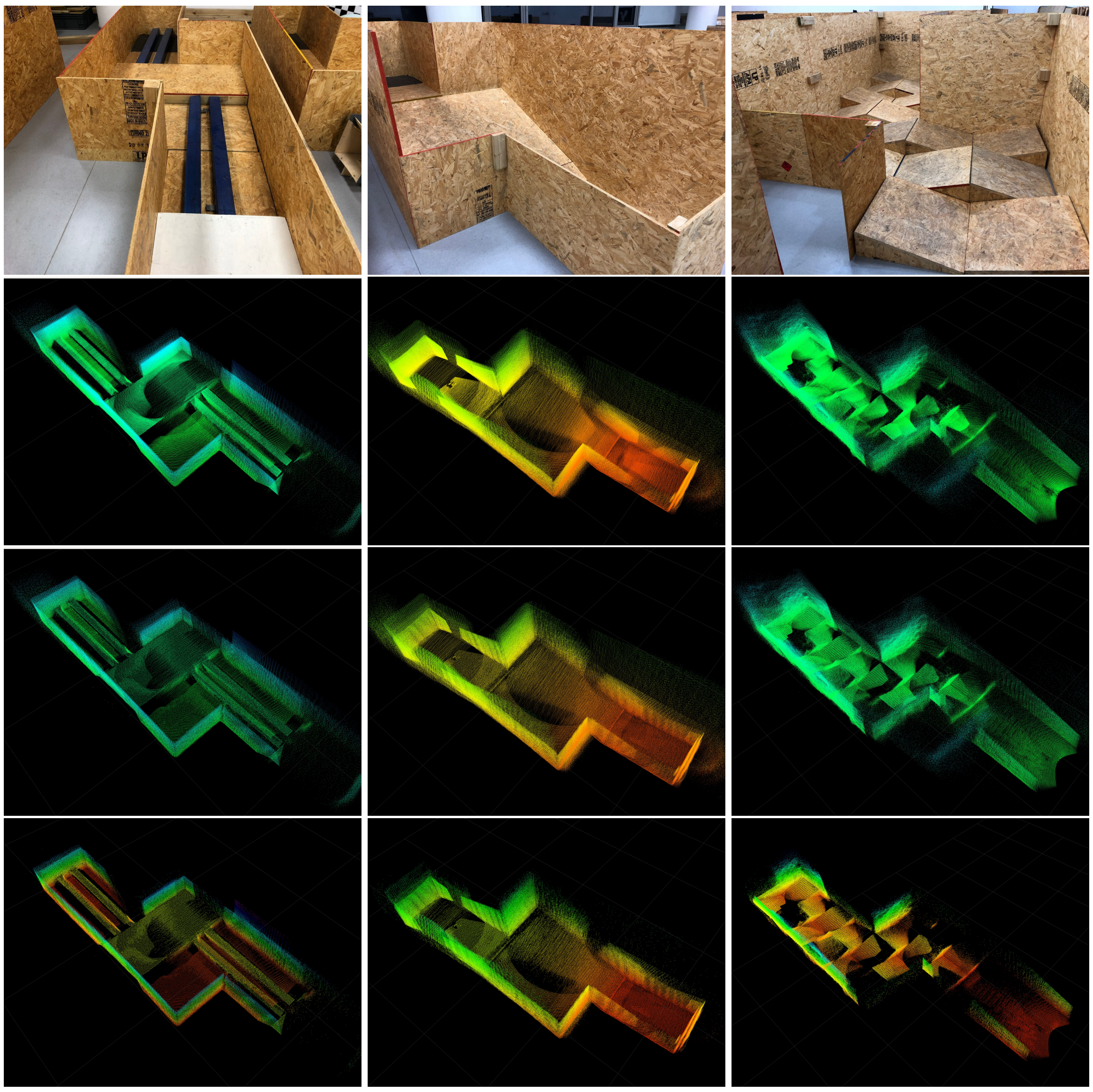


Figure 20. Mapping results in ASTM Standard Test Methods.

Table 2. Efficiency of the octree structure.

Scene	Octree Resolution (m)	Origin (Points)	Octree 5pts	Octree 3pts	Octree 1pt	Loop Cost 5pts (ms)	Speed up (5pts)
lab (large)	0.05	3,981,242	2,026,884	1,562,519	760,411	378	49.1%
arena (middle)	0.02	6,153,656	849,748	619,237	284,029	159	86.2%
arena (small)	0.02	4,259,847	198,161	128,615	48,091	37	95.3%

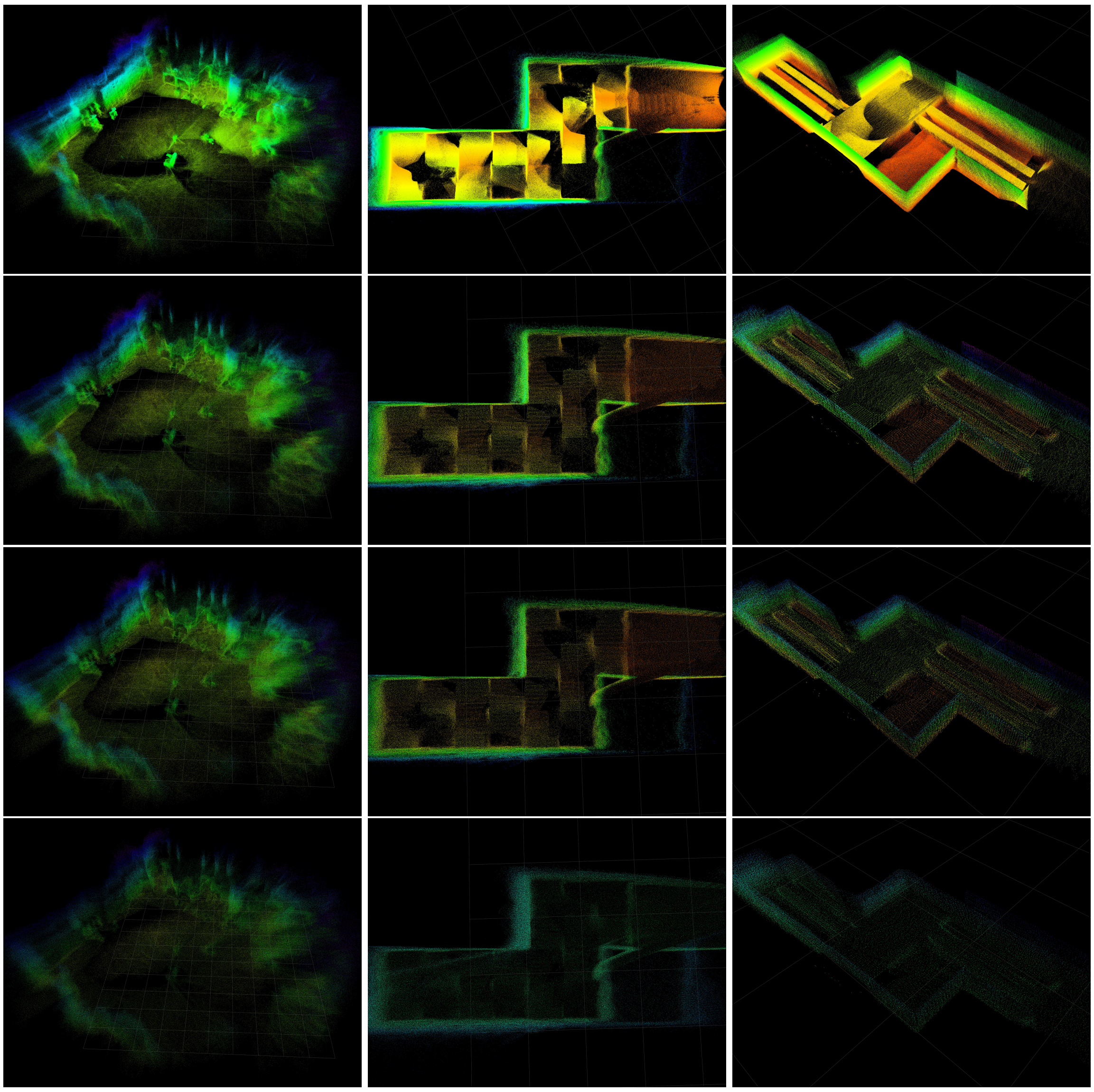


Figure 21. Mapping result corresponding to comparison in Table 2, from left to right: lab (large), arena (middle), arena (small).

6. Conclusions

We proposed a system to fuse color, depth and inertial sensors for trajectory estimation and mapping for small ground rescue robots. Our system called VINS-RGBD is based on the state-of-the-art software VINS-Mono. We are taking advantage of the RealSense D435i, which is small, lightweight and has an integrated IMU. We extend the VINS-Mono system to make use of the depth data during the initialization process as well as during the VIO phase. Exhaustive experiments and comparisons with handheld, wheeled robots and tracked robots are performed to evaluate the performance and versatility of our system. Our software is available as open source and the dataset is also provided. The experiments show the excellent performance of our system compared to the ground truth trajectories, to the VINS-Mono system and also to an ORB-based approach. We show that for ground robots, especially for tracked robots, the noise of the IMU readings is very high due to the vibrations induced by the ground contact. More importantly, under some special motions common for ground robots, the acceleration bias cannot be estimated since it is unobservable. This is a problem for traditional VIO systems, that are relying heavily on the IMU to estimate the unknown scale factor. Using depth information we can prevent the IMU from having a too big negative effect on the results, thus enabling our solution to provide good maps for applications such as path planning or navigation.

In this work, we fuse the RGBD camera data with both gyroscope and accelerometer measurements from the IMU. However, in some extreme situations such as very challenging terrain, the accelerometer is susceptible to high-frequency noise and becomes unreliable. One naive way is to only fuse the RGBD camera images with the gyroscope and don't use the accelerometer, since the depth camera already provides absolute scale information. However, both accelerometer and depth camera have their degeneration conditions. Using both accelerometer and depth cameras also has advantages, e.g., the gravity vector estimated from the accelerometer can correct the pitch and roll angle of the camera. Thus, we keep both and are working on finding better ways to model the accelerometer, especially in extreme conditions. With a pose ground truth, such as recorded in our dataset, it is easy to get accelerometer ground truth and make it possible to use deep learning to model the accelerometer or even IMU more correctly. With better sensors modelling, we expect to achieve more accurate trajectory estimation and mapping results.

To build a whole navigation system, we are also working on flipper planning of our tracked robot under the limited environmental perception, since RGBD cameras typically only provide one view of the environment rather than 360 degrees by LIDAR. The 3D path planning is also a further topic together with ground robot navigation map generation, by utilizing the consistent point cloud information provided by our VINS-RGBD system.

Author Contributions: Conceptualization, Z.S.; Investigation, Z.S.; Methodology, Z.S.; Software, Z.S. and R.L.; Supervision, S.S.; Validation, R.L.; Visualization, R.L.; Writing—original draft, Z.S.; Writing—review & editing, S.S.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bloss, R. Sensor innovations helping unmanned vehicles rapidly growing smarter, smaller, more autonomous and more powerful for navigation, mapping, and target sensing. *Sens. Rev.* **2015**, *35*, 6–9. [CrossRef]
2. Bloss, R. Unmanned vehicles while becoming smaller and smarter are addressing new applications in medical, agriculture, in addition to military and security. *Ind. Robot* **2014**, *41*, 82–86. [CrossRef]
3. Birk, A.; Schwertfeger, S.; Pathak, K. A networking framework for teleoperation in safety, security, and rescue robotics. *IEEE Wirel. Commun.* **2009**, *16*, 6–13. [CrossRef]
4. Colas, F.; Mahesh, S.; Pomerleau, F.; Liu, M.; Siegwart, R. 3D path planning and execution for search and rescue ground robots. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 722–727.
5. Pathak, K.; Birk, A.; Schwertfeger, S.; Delchef, I.; Markov, S. Fully autonomous operations of a jacobs rugbot in the robocup rescue robot league 2006. In Proceedings of the IEEE International Workshop on Safety, Security and Rescue Robotics, Rome, Italy, 27–29 September 2007; pp. 1–6.
6. Thrun, S.; Thayer, S.; Whittaker, W.; Baker, C.; Burgard, W.; Ferguson, D.; Hahnel, D.; Montemerlo, D.; Morris, A.; Omohundro, Z.; et al. Autonomous exploration and mapping of abandoned mines. *IEEE Robot. Autom. Mag.* **2004**, *11*, 79–91. [CrossRef]
7. Grisetti, G.; Stachniss, C.; Burgard, W. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Trans. Robot.* **2007**, *23*, 34. [CrossRef]
8. Birk, A.; Vaskevicius, N.; Pathak, K.; Schwertfeger, S.; Poppinga, J.; Buelow, H. 3-D perception and modeling. *IEEE Robot. Autom. Mag.* **2009**, *16*, 53–60. [CrossRef]

9. Kohlbrecher, S.; Von Stryk, O.; Meyer, J.; Klingauf, U. A flexible and scalable slam system with full 3d motion estimation. In Proceedings of the IEEE International Symposium on Safety, Security, and Rescue Robotics, Kyoto, Japan, 1–5 November 2011; pp. 155–160.
10. Davison, A.J. Real-time simultaneous localisation and mapping with a single camera. In Proceedings of the ICCV 2003, Nice, France, 13–16 October 2003; Volume 3, pp. 1403–1410.
11. Huang, G.P.; Mourikis, A.I.; Roumeliotis, S.I. Analysis and improvement of the consistency of extended Kalman filter based SLAM. In Proceedings of the IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, 19–23 May 2008; pp. 473–479.
12. Mur-Artal, R.; Montiel, J.M.M.; Tardos, J.D. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robot.* **2015**, *31*, 1147–1163. [CrossRef]
13. Endres, F.; Hess, J.; Sturm, J.; Cremers, D.; Burgard, W. 3-D mapping with an RGB-D camera. *IEEE Trans. Robot.* **2014**, *30*, 177–187. [CrossRef]
14. Newcombe, R.A.; Izadi, S.; Hilliges, O.; Molyneaux, D.; Kim, D.; Davison, A.J.; Kohi, P.; Shotton, J.; Hodges, S.; Fitzgibbon, A. KinectFusion: Real-time dense surface mapping and tracking. In Proceedings of the IEEE International Symposium on Mixed and Augmented Reality, Basel, Switzerland, 26–29 October 2011; pp. 127–136.
15. Qin, T.; Li, P.; Shen, S. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Trans. Robot.* **2018**, *34*, 1004–1020. [CrossRef]
16. Li, M.; Mourikis, A.I. High-precision, consistent EKF-based visual-inertial odometry. *Int. J. Robot. Res.* **2013**, *32*, 690–711. [CrossRef]
17. Bloesch, M.; Omari, S.; Hutter, M.; Siegwart, R. Robust visual inertial odometry using a direct EKF-based approach. In Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 298–304.
18. Murphy, R.R.; Kravitz, J.; Stover, S.L.; Shoureshi, R. Mobile robots in mine rescue and recovery. *IEEE Robot. Autom. Mag.* **2009**, *16*, 91–103. [CrossRef]
19. Ghamry, K.A.; Kamel, M.A.; Zhang, Y. Cooperative forest monitoring and fire detection using a team of UAVs-UGVs. In Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS), Arlington, VA, USA, 7–10 June 2016; pp. 1206–1211.
20. Menna, M.; Gianni, M.; Ferri, F.; Pirri, F. Real-time autonomous 3D navigation for tracked vehicles in rescue environments. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; pp. 696–702.
21. Wu, K.J.; Guo, C.X.; Georgiou, G.; Roumeliotis, S.I. Vins on wheels. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 5155–5162.
22. Fritsche, P.; Zeise, B.; Hemme, P.; Wagner, B. Fusion of radar, LiDAR and thermal information for hazard detection in low visibility environments. In Proceedings of the IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), Shanghai, China, 11–13 October 2017; pp. 96–101.
23. Lichtsteiner, P.; Posch, C.; Delbruck, T. A 128 $\times$ 128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE J. Solid-State Circuits* **2008**, *43*, 566–576. [CrossRef]
24. Keselman, L.; Woodfill, J.I.; Grunnet-Jepsen, A.; Bhowmik, A. Intel (r) realsense (tm) stereoscopic depth cameras. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, USA, 21–26 July 2017; pp. 1267–1276.
25. Carfagni, M.; Furferi, R.; Governi, L.; Santarelli, C.; Servi, M.; Uccheddu, F.; Volpe, Y. Metrological and Critical Characterization of the Intel D415 Stereo Depth Camera. *Sensors* **2019**, *19*, 489. [CrossRef]
26. Intel RealSense Depth Camera D435i. Available online: https://www.intelrealsense.com/depth-camera-d435i (accessed on 14 May 2019).
27. Yang, K.; Wang, K.; Hu, W.; Bai, J. Expanding the detection of traversable area with RealSense for the visually impaired. *Sensors* **2016**, *16*, 1954. [CrossRef]
28. Vit, A.; Shani, G. Comparing RGB-D Sensors for Close Range Outdoor Agricultural Phenotyping. *Sensors* **2018**, *18*, 4413. [CrossRef]
29. Available online: https://robotics.shanghaitech.edu.cn/datasets/VINS-RGBD (accessed on 14 May 2019).
30. Available online: https://github.com/STAR-Center/VINS-RGBD (accessed on 14 May 2019).

31. Klein, G.; Murray, D. Parallel tracking and mapping for small AR workspaces. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*; IEEE Computer Society: Washington, DC, USA, 2007; pp. 1–10.
32. Gálvez-López, D.; Tardos, J.D. Bags of binary words for fast place recognition in image sequences. *IEEE Trans. Robot.* **2012**, *28*, 1188–1197. [CrossRef]
33. Mur-Artal, R.; Tardós, J.D. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Trans. Robot.* **2017**, *33*, 1255–1262. [CrossRef]
34. Scaramuzza, D. Performance evaluation of 1-point-RANSAC visual odometry. *J. Field Robot.* **2011**, *28*, 792–811. [CrossRef]
35. Ortin, D.; Montiel, J.M.M. Indoor robot motion based on monocular images. *Robotica* **2001**, *19*, 331–342. [CrossRef]
36. Choi, S.; Kim, J.H. Fast and reliable minimal relative pose estimation under planar motion. *Image Vis. Comput.* **2018**, *69*, 103–112. [CrossRef]
37. Weiss, S.; Achtelik, M.W.; Lynen, S.; Chli, M.; Siegwart, R. Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments. In Proceedings of the IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 14–18 May 2012; pp. 957–964.
38. Lynen, S.; Achtelik, M.W.; Weiss, S.; Chli, M.; Siegwart, R. A robust and modular multi-sensor fusion approach applied to mav navigation. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 3923–3929.
39. Mourikis, A.I.; Roumeliotis, S.I. A multi-state constraint Kalman filter for vision-aided inertial navigation. In Proceedings of the IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007; pp. 3565–3572.
40. Leutenegger, S.; Furgale, P.; Rabaud, V.; Chli, M.; Konolige, K.; Siegwart, R. Keyframe-based visual-inertial slam using nonlinear optimization. In Proceedings of the 2013 Robotics: Science and Systems Conference, Berlin, Germany, 24–28 June 2013.
41. Qin, T.; Cao, S.; Pan, J.; Shen, S. A General Optimization-based Framework for Global Pose Estimation with Multiple Sensors. *arXiv* **2019**, arXiv:1901.03642.
42. Mur-Artal, R.; Tardós, J.D. Visual-inertial monocular SLAM with map reuse. *IEEE Robot. Autom. Lett.* **2017**, *2*, 796–803. [CrossRef]
43. Mu, X.; Chen, J.; Zhou, Z.; Leng, Z.; Fan, L. Accurate Initial State Estimation in a Monocular Visual-Inertial SLAM System. *Sensors* **2018**, *18*, 506.
44. Hong, E.; Lim, J. Visual-Inertial Odometry with Robust Initialization and Online Scale Estimation. *Sensors* **2018**, *18*, 4287. [CrossRef]
45. Guo, C.X.; Roumeliotis, S.I. IMU-RGBD camera 3D pose estimation and extrinsic calibration: Observability analysis and consistency improvement. In Proceedings of the IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 2935–2942.
46. Brunetto, N.; Salti, S.; Fioraio, N.; Cavallari, T.; Stefano, L. Fusion of inertial and visual measurements for rgb-d slam on mobile devices. In Proceedings of the IEEE International Conference on Computer Vision Workshops, Santiago, Chile, 11–12 and 17–18 December 2015; pp. 1–9.
47. Falquez, J.M.; Kasper, M.; Sibley, G. Inertial aided dense & semi-dense methods for robust direct visual odometry. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, South Korea, 9–14 October 2016; pp. 3601–3607.
48. Laidlow, T.; Bloesch, M.; Li, W.; Leutenegger, S. Dense RGB-D-inertial SLAM with map deformations. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 6741–6748.
49. Zhu, Z.; Xu, F. Real-time Indoor Scene Reconstruction with RGBD and Inertia Input. *arXiv* **2018**, arXiv:1812.03015.
50. Wang, C.; Hoang, M.C.; Xie, L.; Yuan, J. Non-iterative RGB-D-inertial Odometry. *arXiv* **2017**, arXiv:1710.05502.
51. Ling, Y.; Liu, H.; Zhu, X.; Jiang, J.; Liang, B. RGB-D Inertial Odometry for Indoor Robot via Keyframe-based Nonlinear Optimization. In Proceedings of the IEEE International Conference on Mechatronics and Automation (ICMA), Changchun, China, 5–8 August 2018; pp. 973–979.

52. Herbert, M.; Caillas, C.; Krotkov, E.; Kweon, I.S.; Kanade, T. Terrain mapping for a roving planetary explorer. In Proceedings of the International Conference on Robotics and Automation, Scottsdale, AZ, USA, 14–19 May 1989; pp. 997–1002.
53. Triebel, R.; Pfaff, P.; Burgard, W. Multi-level surface maps for outdoor terrain mapping and loop closing. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, 9–15 October 2006; pp. 2276–2282.
54. Hornung, A.; Wurm, K.M.; Bennewitz, M.; Stachniss, C.; Burgard, W. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Auton. Robots* **2013**, *34*, 189–206. [CrossRef]
55. Yang, S.; Yang, S.; Yi, X.; Yang, W. Real-time globally consistent 3D grid mapping. In Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO), Macau, China, 5–8 December 2017; pp. 929–935.
56. Delmerico, J.; Scaramuzza, D. A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 2502–2509.
57. Lucas, B.D.; Kanade, T. An Iterative Image Registration Technique with an Application to Stereo Vision. 1981. Available online: https://ri.cmu.edu/pub_files/pub3/lucas_bruce_d_1981_2/lucas_bruce_d_1981_2.pdf (accessed on 14 May 2019).
58. Shi, J.; Tomasi, C. *Good Features to Track*; Technical Report; Cornell University: Ithaca, NY, USA, 1993.
59. Forster, C.; Carlone, L.; Dellaert, F.; Scaramuzza, D. On-Manifold Preintegration for Real-Time Visual–Inertial Odometry. *IEEE Trans. Robot.* **2017**, *33*, 1–21. [CrossRef]
60. Rehder, J.; Nikolic, J.; Schneider, T.; Hinzmann, T.; Siegwart, R. Extending kalibr: Calibrating the extrinsics of multiple imus and of individual axes. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 4304–4311.
61. Zhi, X.; Schwertfeger, S. Simultaneous hand-eye calibration and reconstruction. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 1470–1477.
62. Civera, J.; Davison, A.J.; Montiel, J.M. Inverse depth parametrization for monocular SLAM. *IEEE Trans. Robot.* **2008**, *24*, 932–945. [CrossRef]
63. Sibley, G.; Matthies, L.; Sukhatme, G. Sliding window filter with application to planetary landing. *J. Field Robot.* **2010**, *27*, 587–608. [CrossRef]
64. Hesch, J.A.; Kottas, D.G.; Bowman, S.L.; Roumeliotis, S.I. Consistency analysis and improvement of vision-aided inertial navigation. *IEEE Trans. Robot.* **2014**, *30*, 158–176. [CrossRef]
65. Wu, K.J.; Roumeliotis, S.I. *Unobservable Directions of VINS under Special Motions*; Department of Computer Science & Engineering, University of of Minnesota: Minneapolis, MN, USA, 2016.
66. Nistér, D. An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Anal. Mach. Intell.* **2004**, *26*, 756–777. [CrossRef]
67. Gao, X.S.; Hou, X.R.; Tang, J.; Cheng, H.F. Complete solution classification for the perspective-three-point problem. *IEEE Trans. Pattern Anal. Mach. Intell.* **2003**, *25*, 930–943.
68. Lepetit, V.; Moreno-Noguer, F.; Fua, P. Epnp: An accurate o (n) solution to the pnp problem. *Int. J. Comput. Vis.* **2009**, *81*, 155. [CrossRef]
69. Penate-Sanchez, A.; Andrade-Cetto, J.; Moreno-Noguer, F. Exhaustive linearization for robust camera pose and focal length estimation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 2387–2400. [CrossRef]
70. Agarwal, S.; Mierle, K. Ceres Solver. 2012. Available online: http://ceres-solver.org/ (accessed on 14 May 2019).
71. Besl, P.J.; McKay, N.D. Method for registration of 3-D shapes. In *Proceedings of the Sensor Fusion IV: Control Paradigms and Data Structures*; International Society for Optics and Photonics: Bellingham, WA, USA, 1992; Volume 1611, pp. 586–607.
72. Meagher, D. Geometric modeling using octree encoding. *Comput. Graphics Image Process.* **1982**, *19*, 129–147. [CrossRef]
73. Burri, M.; Nikolic, J.; Gohl, P.; Schneider, T.; Rehder, J.; Omari, S.; Achtelik, M.W.; Siegwart, R. The EuRoC micro aerial vehicle datasets. *Int. J. Robot. Res.* **2016**, *35*, 1157–1163. [CrossRef]

74. Sturm, J.; Engelhard, N.; Endres, F.; Burgard, W.; Cremers, D. A benchmark for the evaluation of RGB-D SLAM systems. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, Algarve, 7–12 October 2012; pp. 573–580.
75. OptiTrack. Available online: https://optitrack.com (accessed on 14 May 2019).
76. Zhang, Z.; Scaramuzza, D. A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 7244–7251.
77. Jackal. Available online: https://www.clearpathrobotics.com/jackal-small-unmanned-ground-vehicle (accessed on 14 May 2019).
78. ASTM 2827-11 Crossing Pitch/Roll Ramps. Available online: https://www.astm.org/Standards/E2827.htm (accessed on 14 May 2019).
79. Sheh, R.; Schwertfeger, S.; Visser, A. 16 Years of RoboCup Rescue. *KI-Künstliche Intelligenz* **2016**, *30*, 267–277. [CrossRef]
80. Sheh, R.; Jacoff, A.; Virts, A.M.; Kimura, T.; Pellenz, J.; Schwertfeger, S.; Suthakorn, J. Advancing the state of urban search and rescue robotics through the RoboCupRescue robot league competition. In *Field and Service Robotics*; Springer: Berlin, Germany, 2014.