2014 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS 2014)
September 14-18, 2014, Chicago, IL, USA

Real-time Depth Enhanced Monocular Odometry

Ji Zhang, Michael Kaess, and Sanjiv Singh

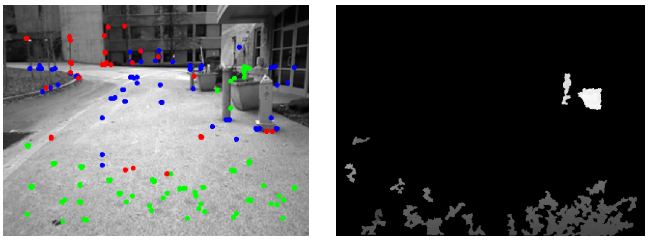
Abstract— **Visual odometry can be augmented by depth information such as provided by RGB-D cameras, or from lidars associated with cameras. However, such depth information can be limited by the sensors, leaving large areas in the visual images where depth is unavailable. Here, we propose a method to utilize the depth, even if sparsely available, in recovery of camera motion. In addition, the method utilizes depth by triangulation from the previously estimated motion, and salient visual features for which depth is unavailable. The core of our method is a bundle adjustment that refines the motion estimates in parallel by processing a sequence of images, in a batch optimization. We have evaluated our method in three sensor setups, one using an RGB-D camera, and two using combinations of a camera and a 3D lidar. Our method is rated #2 on the KITTI odometry benchmark irrespective of sensing modality, and is rated #1 among visual odometry methods.**

I. Introduction

Visual odometry is the process of egomotion estimation given a sequence of camera imagery. Typically, monocular imagery is insufficient to compute the egomotion because motion along the camera optical axis can cause little motion of visual features and therefore the estimation problem can be degenerate. With a single camera [1]–[4], if assuming unconstrained motion, rotation can be recovered but translation is up to scale. This situation can be mitigated by using extra information such as knowledge of a non-holonomic motion constraint [5], or measurements from an IMU integrated with the visual odometry [6]. However, the results are dependent on the quality of the extra information involved.

It is possible to obtain scale by using multiple cameras simultaneously [7], [8]. However, this comes at its own cost– reduced effective field of view and a limitation on the range that can be accurately recovered from the multiple cameras. If a small baseline is used, depth is uncertain for features far away from the camera. But, if the cameras are separated significantly, inter-camera calibration becomes difficult and accuracy can be hard to ensure. When used in scenes where a large difference exists between near and far objects, depth can only be obtained in certain parts of the images.

This paper proposes a method that can effectively utilize depth information along with the imagery, even the depth is only sparsely provided. The method maintains and registers a depth map using the estimated motion of the camera. Visual features are associated with depth from the depth map, or by triangulation using the previously estimated motion. Salient visual features for which depth is unavailable are also used, which themselves provide constraints in solving the problem. Further, the method contains a bundle adjustment which refines the motion estimates in parallel by processing a sequence of images, in a batch optimization.



(a) (b)

Fig. 1. (a) Features tracked at an image frame. The green dots represent features whose depth comes from the depth map, the blue dots represent features whose depth is determined by triangulation using the previously estimated motion of the camera, and the red dots represent features without depth. The proposed method uses all three types of features in determining motion. (b) A depth image from an RGB-D camera corresponding to (a), where depth information is only available in the vicinity of the camera. The gray and white pixels represent close and far objects with respect to the camera, and the black pixels are areas that depth is unavailable.

The proposed method is not limited to RGB-D cameras. It can be adapted to various types of cameras as long as depth information can be acquired and associated. We have collected experimental data using an RGB-D camera and a custom-built sensor consisting a camera and 3D lidar (a 2-axis laser scanner). We have also evaluated the method using the well-known KITTI benchmark datasets [9], [10], which contain carefully registered data from a number of sensors. The method reported here uses images from a single camera in a stereo pair and laser scans from a high rate lidar. The results are ranked by the benchmark server.

The rest of this paper is organized as follows. In section II, we discuss related work. In section III, we state the problem. The sensor hardware and software system are described in Section IV. The frame to frame motion estimation and bundle adjustment are discussed in Sections V and VI. Experimental results are in Section VII and conclusion in Section VIII.

II. Related Work

Vision based methods are now common for motion estimation [11], [12]. To solve 6DOF camera motion, stereo vision systems are often used [13], [14]. The methods track and match visual features between image frames. Depth information of the features are retrieved by triangular geometry established from the two image views with the camera baseline as a reference. Among this area, Paz et al. [15] estimate the motion of stereo hand-hold cameras. The depth is recovered for features close to the cameras, which help

J. Zhang, M. Kaess, and S. Singh are with the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA 15213. Emails: zhangji@cmu.edu, kaess@cmu.edu, and ssingh@cmu.edu.

The paper is based upon work supported by the National Science Foundation under Grant No. IIS-1328930.

solve scale of the translation estimation. Konolige, at al.'s stereo visual odometry recovers the motion from bundle adjustment [8]. The method is integrated with an IMU and capable for long distance off-road navigation.

The introduction of RGB-D cameras has drawn great attention in the research of visual odometry [16]–[19]. Such cameras provide RGB images along with depth information within the camera range limit. Huang et al. [20] use tracked visual features with known depth from an RGB-D camera to compute the motion estimates. The method eliminates features if the corresponding depth is unavailable from the depth images. Henry et al. [21] integrate visual features with the Iterative Closest Point (ICP) method [22]. The motion estimation employs a joint optimization by minimizing combined 2D and 3D feature matching errors. Another popular method is dense tracking [23], [24]. The method minimizes the photometric error using a dense 3D model of the environment from the depth images. Overall, the methods [20], [21], [23], [24] rely on sufficient depth information for image processing. This can limit their applications especially if the methods are used in open environments, where depth information can only be limitedly available.

Few RGB-D visual odometry methods are able to handle insufficiently provided depth information. In the work of Hu et al., a heuristic switch is used to choose between an RGB-D and a monocular visual odometry method [25]. In contrast to these methods [20], [21], [23]–[25], we propose a single method to handle sparse depth information, by combining both features with and without depth. The method is compared to [20], [23] experimentally, and robust estimation results are shown. Further, since the method maintains and registers a depth map, it can use depth information from different types of sensors. The method is currently tested with depth from RGB-D cameras and lidars, but is supposed to work with depth from stereo cameras also.

III. Notations and Task Description

The visual odometry problem addressed in this paper is to estimate the motion of a camera using monocular images with assistance of depth information. We assume that the camera is well modeled as a pinhole camera [26], the camera intrinsic parameters are known from pre-calibration, and the lens distortion is removed. As a convention in this paper, we use right superscript k, $k \in Z^+$ to indicate image frames. Define camera coordinate system, $\{C\}$, as follows,

- $\{C\}$ is a 3D coordinate system with its origin at the camera optical center. The x-axis points to the left, the y-axis points upward, and the z-axis points forward coinciding with the camera principal axis.

We want to utilize features with and without depth. Let $\mathcal{I}$ be a set of feature points. For a feature i, $i \in \mathcal{I}$, that is associated with depth, its coordinates in $\{C^k\}$ are denoted as $\boldsymbol{X}_i^k$, where $\boldsymbol{X}_i^k = [x_i^k,\ y_i^k,\ z_i^k]^T$. For a feature with unknown depth, we normalize the coordinates such that its z-coordinate is one. Let $\bar{\boldsymbol{X}}_i^k$ be the normalized term of the feature, $\bar{\boldsymbol{X}}_i^k = [\bar{x}_i^k,\ \bar{y}_i^k,\ 1]^T$. Intuitively, we can imagine a plane that is parallel to the $x-y$ plane of $\{C^k\}$ and at a unit length in front of the coordinate origin, and $\bar{\boldsymbol{X}}_i^k$ is the point projected onto the plane. With notations defined, our visual odometry problem can be described as

Problem: Given a sequence of image frames k, $k \in Z^+$, and features, $\mathcal{I}$, compute the motion of the camera between each two consecutive frames, k and $k-1$, using $\boldsymbol{X}_i^k$, if the depth is available, and $\bar{\boldsymbol{X}}_i^k$, if the depth is unknown, $i \in \mathcal{I}$.

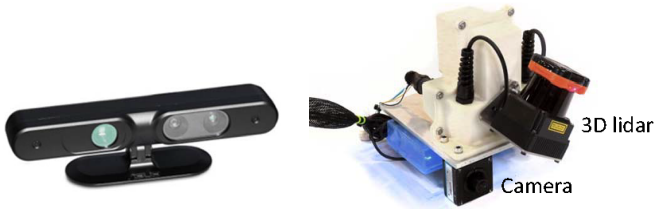
IV. System Overview

A. Sensor Hardware

The proposed method is validated on three different sensor systems. The first two sensors shown in Fig. 2 are used to acquire author-collected data, while the third one uses configuration of the KITTI benchmark datasets. Through the paper, we will use data from the first two sensors to illustrate the method. Fig. 2(a) is an Xtion Pro Live RGB-D camera. The camera is capable of providing RGB and depth images at 30Hz, with 640×480 resolution and $58°$ horizontal field of view. Fig. 2(b) shows a custom-built camera and 3D lidar. The camera can provide RGB images up to 60Hz, with 744×480 resolution and $83°$ horizontal field of view. The 3D lidar is based on a Hokuyo UTM-30LX laser scanner, which has $180°$ field of view with $0.25°$ resolution and 40 lines/sec scan rate. The laser scanner is actuated by a motor for rotational motion to realize 3D scan.

B. Software System Overview

Fig. 3 shows a diagram of the software system. First, visual features are tracked by the feature tracking block. Depth images from RGB-D cameras or point clouds from lidars are registered by the depth map registration block, using the estimated motion. The block also associates depth for the visual features. The frame to frame motion estimation block takes the features as the input, and its output is refined by the bundle adjustment block using sequences of images. The bundle adjustment runs at a low frequency (around 0.25-1.0Hz). The transform integration block combines the high frequency frame to frame motion with the low frequency refined motion, and generates integrated motion transforms at the same frequency as the frame to frame motion transforms. Section V and VI present each block in detail.



(a) (b)

Fig. 2. Two sensors involved in the evaluation. (a) An Xtion Pro Live RGB-D camera. The camera is capable of providing 30Hz RGB and depth images, with 640×480 resolution and $58°$ HFV. (b) A custom-built camera and 3D lidar. The camera provides up to 60Hz RGB images with 744×480 resolution and $83°$ HFV. The 3D lidar consists of a Hokuyo UTM-30LX laser scanner rotated by a motor to realize 3D scan. The laser scanner has $180°$ FOV with $0.25°$ resolution and 40 lines/sec scan rate.

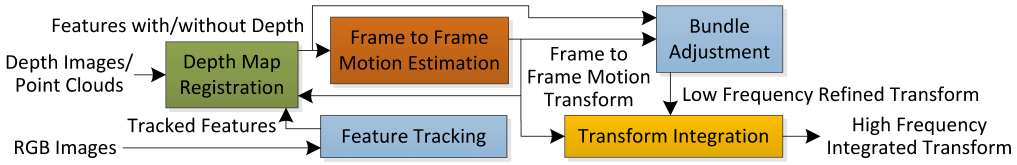


Fig. 3. Block diagram of the visual odometry software system.

V. Frame to Frame Motion Estimation

A. Mathematical Derivation

We start with mathematical derivation for the frame to frame motion estimation. The corresponding algorithm is discussed in the next section. Recall that $\boldsymbol{X}_i^{k-1}$ and $\boldsymbol{X}_i^k$ are the coordinates of a tracked feature in $\{C^{k-1}\}$ and $\{C^k\}$. Define $\mathbf{R}$ and $\boldsymbol{T}$ as the 3×3 rotation matrix and 3×1 translation vector of the camera between the two frames, we model the camera motion as rigid body transformation,

$$\boldsymbol{X}_i^k = \mathbf{R}\boldsymbol{X}_i^{k-1} + \boldsymbol{T}. \tag{1}$$

Next, we will work on features with known depth. Acquiring depth for the features will be discussed in the later part of this paper. Here, note that we only use depth information from one of the two frames. We choose frame $k-1$ for simplicity of implementation. This is because the depth maps are registered in the camera coordinates by the previously estimated motion. By the time of frame k, the depth map at frame $k-1$ is available, and the depth of $\boldsymbol{X}_i^{k-1}$ is associated. Recall that $\bar{\boldsymbol{X}}_i^k$ is the normalized term of $\boldsymbol{X}_i^k$, where the z-coordinate is one, we rewrite (1) as

$$z_i^k \bar{\boldsymbol{X}}_i^k = \mathbf{R}\boldsymbol{X}_i^{k-1} + \boldsymbol{T}. \tag{2}$$

Eq. (2) contains three rows. Combining the 1st and 2nd rows with the 3rd row, respectively, we can eliminate z_i^k. This gives us two equations as follows,

$$(\boldsymbol{R}_1 - \bar{x}_i^k \boldsymbol{R}_3)\boldsymbol{X}_i^{k-1} + T_1 - \bar{x}_i^k T_3 = 0, \tag{3}$$

$$(\boldsymbol{R}_2 - \bar{y}_i^k \boldsymbol{R}_3)\boldsymbol{X}_i^{k-1} + T_2 - \bar{y}_i^k T_3 = 0, \tag{4}$$

where $\boldsymbol{R}_h$ and T_h, $h \in \{1, 2, 3\}$ are the h-th row of $\mathbf{R}$ and $\boldsymbol{T}$, respectively.

For a feature with unknown depth, we rewrite (1) as the following. Here, $\bar{\boldsymbol{X}}_i^{k-1}$ is the normalized term of $\boldsymbol{X}_i^{k-1}$,

$$z_i^k \bar{\boldsymbol{X}}_i^k = z_i^{k-1} \mathbf{R} \bar{\boldsymbol{X}}_i^{k-1} + \boldsymbol{T}. \tag{5}$$

Eq. (5) also contains three rows. Combining all rows to eliminate both z_i^k and z_i^{k-1}, we obtain,

$$[-\bar{y}_i^k T_3 + T_2,\ \bar{x}_i^k T_3 - T_1,\ -\bar{x}_i^k T_2 + \bar{y}_i^k T_1]\mathbf{R}\bar{\boldsymbol{X}}_i^{k-1} = 0. \tag{6}$$

So far, we have modeled the frame to frame motion for features with and without depth separately. Now, we will solve the motion using both types of features. Define θ as a 3×1 vector, $\theta = [\theta_x,\ \theta_y,\ \theta_z]^T$, where θ_x, θ_y, and θ_z are the rotation angles of the camera around the x-, y-, and z- axes, between frames k and $k-1$. The rotation matrix $\mathbf{R}$ can be expressed by the Rodrigues formula [27],

$$\mathbf{R} = e^{\hat{\theta}} = \mathbf{I} + \frac{\hat{\theta}}{||\theta||}\sin||\theta|| + \frac{\hat{\theta}^2}{||\theta||^2}(1 - \cos||\theta||), \tag{7}$$

where $\hat{\theta}$ is the skew symmetric matrix of θ.

Substituting (7) into (3)-(4), we can derive two equations for a feature with depth, and substituting (7) into (6), we can derive one equation for a feature with unknown depth. Each equation is a function of θ and $\boldsymbol{T}$. Suppose we have a total of m and n features with known and unknown depth. Stacking the equations, we obtain a nonlinear function,

$$\boldsymbol{f}([\boldsymbol{T};\ \theta]) = \epsilon, \tag{8}$$

where $\boldsymbol{f}$ has $2m+n$ rows, ϵ is a $(2m+n)\times 1$ vector containing the residuals, and $[\boldsymbol{T};\ \theta]$ is the vertical joining of the vectors. Compute the Jacobian matrix of $\boldsymbol{f}$ with respect to $[\boldsymbol{T};\ \theta]$, denoted as $\mathbf{J}$, where $\mathbf{J} = \partial f / \partial [\boldsymbol{T};\ \theta]$. (8) can be solved by the Levenberg-Marquardt (LM) method [26],

$$[\boldsymbol{T};\ \theta]^T \leftarrow [\boldsymbol{T};\ \theta]^T - (\mathbf{J}^T\mathbf{J} + \lambda \mathrm{diag}(\mathbf{J}^{\mathrm{T}}\mathbf{J}))^{-1}\mathbf{J}^{\mathrm{T}}\epsilon. \tag{9}$$

Here, λ is a scale factor determined by the LM method.

B. Motion Estimation Algorithm

The frame to frame motion estimation algorithm is presented in Algorithm 1. As we have discussed that the proposed method only uses depth information associated at frame $k-1$, all features taken as the input at frame k are

Algorithm 1: Frame to Frame Motion Estimation

```
input : X̄_i^k, X_i^{k-1} or X̄_i^{k-1}, i ∈ I
output : θ, T
begin
    θ, T ← 0;
    for a number of iterations do
        for each i ∈ I do
            if i is depth associated then
                Derive (3)-(4) using X̄_i^k and X_i^{k-1}, substitute
                (7) into (3)-(4) to obtain two equations, stack
                the equations into (8);
            end
            else
                Derive (6) using X̄_i^k and X̄_i^{k-1}, substitute (7)
                into (6) to obtain one equation, stack the
                equation into (8);
            end
            Compute a bisquare weight for each feature
            based on the residuals in (3)-(4) or (6);
            Update θ, T for one iteration based on (9);
        end
        if the nonlinear optimization converges then
            Break;
        end
    end
    Return θ, T;
end
```

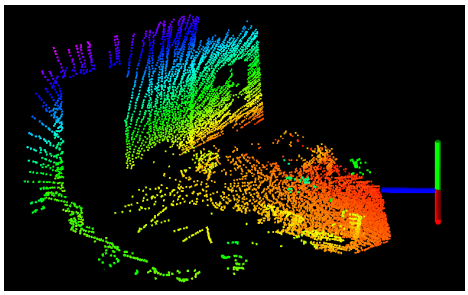
without depth, as $\bar{\boldsymbol{X}}_i^k$. Features at frame $k-1$ are separated into $\boldsymbol{X}_i^{k-1}$ and $\bar{\boldsymbol{X}}_i^{k-1}$ for those with and without depth. On line 8, a feature with depth contributes two equations to the nonlinear function (8), and on line 11, a feature with unknown depth contributes one equation.

The terms θ and $\boldsymbol{T}$ are initialized to zero on line 4. The algorithm is adapted to a robust fitting framework [28]. On line 13, the algorithm assigns a bisquare weight for each feature, based on their residuals in (3)-(4) and (6), respectively. Features that have larger residuals are assigned with smaller weights, and features with residuals larger than a threshold are considered as outliers and assigned with zero weights. On line 14, θ and $\boldsymbol{T}$ are updated for one iteration. The nonlinear optimization terminates if convergence is found, or the maximum iteration number is met. Finally, the algorithm returns the motion estimation θ and $\boldsymbol{T}$.

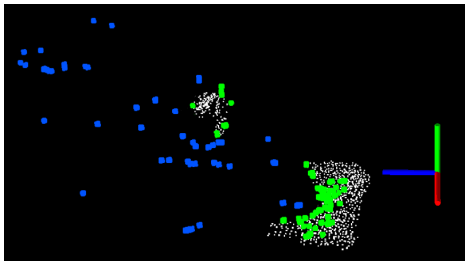
C. Feature Depth Association

In this section, we discuss how to associate depth to the visual features. A depth map is registered by the estimated motion of the camera. The depth map is projected to the last image frame whose transform to the previous frame is established. Here, we use frame $k-1$ to keep the same convention with the previous sections.

New points are added to the depth map upon receiving from depth images or point clouds. Only points in front of the camera are kept, and points that are received a certain time ago are forgotten. Then, the depth map is downsized to maintain a constant point density. We want to keep an even angular interval among the points viewed from the origin of $\{C^{k-1}\}$, or the optical center of the camera at frame $k-1$. Here, we choose angular interval over Euclidean distance interval with the consideration that an image pixel represents an angular interval projected into the 3D environment. We use the same format for the depth map.



(a)



(b)

Fig. 4. (a) An example of the depth map with point clouds perceived by the lidar in Fig. 2(b). The points are color coded by elevation. The camera points to a wall during the data collection. (b) Features projected into the 3D environment corresponding to Fig. 1. The depth map (white colored dots) is from depth images collected by the RGB-D camera in Fig. 2(a), only available within a limited range. The green colored dots are features whose depth is provided by the depth map, and the blue colored dots are from triangulation using the previously estimated motion.

The map points are converted into a spherical coordinate system coinciding with $\{C^{k-1}\}$. A point is represented by its radial distance, azimuthal angle, and polar angle. When downsizing, only the two angular coordinates are considered, and the points are evenly distributed with respect to the angular coordinates. This results in a denser point distribution that is closer to the camera, and vice versa. An example of a registered depth map is shown in Fig. 4(a), color coded by elevation. The point clouds are collected by the lidar in Fig. 2(b), while the camera points to a wall.

To associate depth to the visual features, we store the depth map in a 2D KD-tree [29] based on the two angular coordinates. For each feature i, $i \in \mathcal{I}$, we find three points from the KD-tree that are the closest to the feature. The three points form a local planar patch in the 3D environment, and the 3D coordinates of the feature are found by projecting onto the planar patch. Denote $\hat{\boldsymbol{X}}_j^{k-1}$, $j \in \{1,2,3\}$ as the Euclidean coordinates of the three points in $\{C^{k-1}\}$, and recall that $\boldsymbol{X}_i^{k-1}$ is the coordinates of feature i in $\{C^{k-1}\}$. The depth is computed by solving a function,

$$(\boldsymbol{X}_i^{k-1} - \hat{\boldsymbol{X}}_1^{k-1})((\hat{\boldsymbol{X}}_1^{k-1} - \hat{\boldsymbol{X}}_2^{k-1}) \times (\hat{\boldsymbol{X}}_1^{k-1} - \hat{\boldsymbol{X}}_3^{k-1})) = 0. \quad (10)$$

Further, if the depth is unavailable from the depth map for some features but they are tracked for longer than a certain distance in the Euclidean space, we triangulate the features using the first and the last frames in the image sequences where the features are tracked. Fig. 4(b) gives an example of depth associated features, corresponding to Fig. 1. The white colored dots are points on the depth map, only available within a limited range. The green colored dots represent features whose depth are provided by the depth map, and the blue colored dots are from triangulation.

VI. Bundle Adjustment

The camera frame to frame motion estimated in the previous section is refined by a bundle adjustment, which takes a sequence of images and performs a batch optimization. As a trade-off between accuracy and processing time, we choose one image out of every five images as the bundle adjustment input. The image sequence contains a number of eight images (taken from 40 original images). This allows the batch optimization to finish before the next image sequence is accumulated and ready for processing. The bundle adjustment uses the open source library iSAM [30]. We choose iSAM over other libraries because it supports user-defined camera models, and can conveniently handle both features with and without available depth.

Here, we define another representation of the features, $\tilde{\boldsymbol{X}}_i^k = [\bar{x}_i^k,\ \bar{y}_i^k,\ z_i^k]^T$, where the x- and y- entries contain the normalized coordinates, and the z-entry contains the depth.

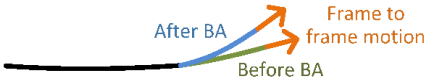


Fig. 5. Illustration of transform integration. The curves represent transforms. The green colored segment is refined by the bundle adjustment and become the blue colored segment, published at a low frequency. The orange colored segments represent frame to frame motion transforms, generated at a high frequency. The transform integration step takes the orange segment from the green segment and connects it to the blue segment. This results in integrated motion transforms published at the high frequency.

For features without depth, z_i^k is set at a default value. Let $\mathcal{J}$ be the set of image frames in the sequence, and let l be the first frame in the set. Upon initialization, all features appear in the sequence are projected into $\{C^l\}$, denoted as $\tilde{\boldsymbol{X}}_i^l$, $i \in \mathcal{I}$. Define $\mathcal{T}_l^j$ as the transform projecting $\tilde{\boldsymbol{X}}_i^l$ from $\{C^l\}$ to $\{C^j\}$, where j is a different frame in the sequence, $j \in \mathcal{J} \setminus \{l\}$. The bundle adjustment minimizes the following function by adjusting the motion transform between each two consecutive frames and the coordinates of $\tilde{\boldsymbol{X}}_i^l$,

$$\min \sum_{i,j} (\mathcal{T}_l^j(\tilde{\boldsymbol{X}}_i^l) - \tilde{\boldsymbol{X}}_i^j)^T \Omega_i^j (\mathcal{T}_l^j(\tilde{\boldsymbol{X}}_i^l) - \tilde{\boldsymbol{X}}_i^j), \quad i \in \mathcal{I},\ j \in \mathcal{J} \setminus \{l\}. \tag{11}$$

Here, $\tilde{\boldsymbol{X}}_i^j$ represents the observation of feature i at frame j, and Ω_i^j is its information matrix. The first two entries on the diagonal of Ω_i^j are given constant values. If the depth is from the depth map, the 3rd entry is at a larger value, and if the depth is from triangulation, the value is smaller and is inversely proportional to the square of the depth. A zero value is used for features with unknown depth.

The bundle adjustment publishes refined motion transforms at a low frequency. With the camera frame rate between 10-40Hz, the bundle adjustment runs at 0.25-1.0Hz. As illustrated in Fig. 5, a transform integration step takes the bundle adjustment output and combines it with the high frequency frame to frame motion estimates. The result is integrated motion transforms published at the high frequency as the frame to frame motion transforms.

VII. Experiments

The visual odometry is tested with author-collected data and the KITTI benchmark datasets. It tracks Harris corners [26] by the Kanade Lucas Tomasi (KLT) method [31]. The program is implemented in C++ language, on robot operating system (ROS) [32] in Linux. The algorithms run on a laptop computer with 2.5GHz cores and 6GB memory, using around three cores for computation. The feature tracking and bundle adjustment (Section VI) take one core each, and the frame to frame motion estimation (Section V) and depth map registration together consume another core. Our software code and datasets are publicly available[1], in two different versions based on the two sensors in Fig. 2.

A. Tests with Author-collected Datasets

We first conduct tests with author-collected datasets using the two sensors in Fig. 2. The data is collected from four types of environments shown in Fig. 6: a conference room, a large lobby, a clustered road, and a flat lawn. The difficulty increases over the tests as the environments are opener and depth information changes from dense to sparse. We present two images from each dataset, on the 2nd and 4th rows in Fig. 6. The red colored areas indicate coverage of depth maps, from the RGB-D camera (right figure) and the lidar (left figure). Here, note that the depth map registers depth images from the RGB-D camera or point clouds from the lidar at multiple frames, and usually contains more information than that from a single frame. With the RGB-D camera, the average amount of imaged area covered by the depth map reduces from 94% to 23% over the tests. The lidar has a longer detection range and can provide more depth information in open environments. The depth coverage changes from 89% to 47% of the images.

The camera frame rate is set at 30Hz for both sensors. To evenly distribute the features within the images, we separate an image into 3×5 identical subregions. Each subregion provides maximally 30 features, giving maximally 450 features in total. The method is compared to two popular RGB-D visual odometry methods. Fovis estimates the motion of the camera by tracking image features, and depth is associated to the features from the depth images [20]. DVO is a dense tracking method that minimizes the photometric error within the overall images [23]. Both methods use data from the RGB-D camera. Our method is separated into two versions, using the two sensors in Fig. 2, respectively. The resulting trajectories are presented on the 1st and 3rd rows in Fig. 6, and the accuracy is compared in Table I, using errors in 3D coordinates. Here, the camera starts and stops at the same position, and the gap between the two ends of a trajectory compared to the length of the trajectory is considered the relative position error.

From these results, we conclude that all four methods function similarly when depth information is sufficient (in the room environment), while the relative error of DVO is slightly lower than the other methods. However, as the depth information becomes sparser, the performance of Fovis and DVO reduces significantly. During the last two tests, Fovis frequently pauses without giving odometry output due to insufficient number of inlier features. DVO continuously generates output but drifts heavily. This is because both methods use only imaged areas where depth is available, leaving large amount of areas in the visual images being unused. On the other hand, the two versions of our method

TABLE I

Results using author-collected data. The error is measured at the end of a trajectory as a % of the distance traveled

Environment	Distance	Relative position error			
		Fovis	DVO	Our VO (RGB-D)	Our VO (Lidar)
Room	16m	2.72%	1.87%	2.14%	2.06%
Lobby	56m	5.56%	8.36%	1.84%	1.79%
Road	87m	13.04%	13.60%	1.53%	0.79%
Lawn	86m	9.97%	32.07%	3.72%	1.73%

[1] `wiki.ros.org/demo_rgbd` and `wiki.ros.org/demo_lidar`

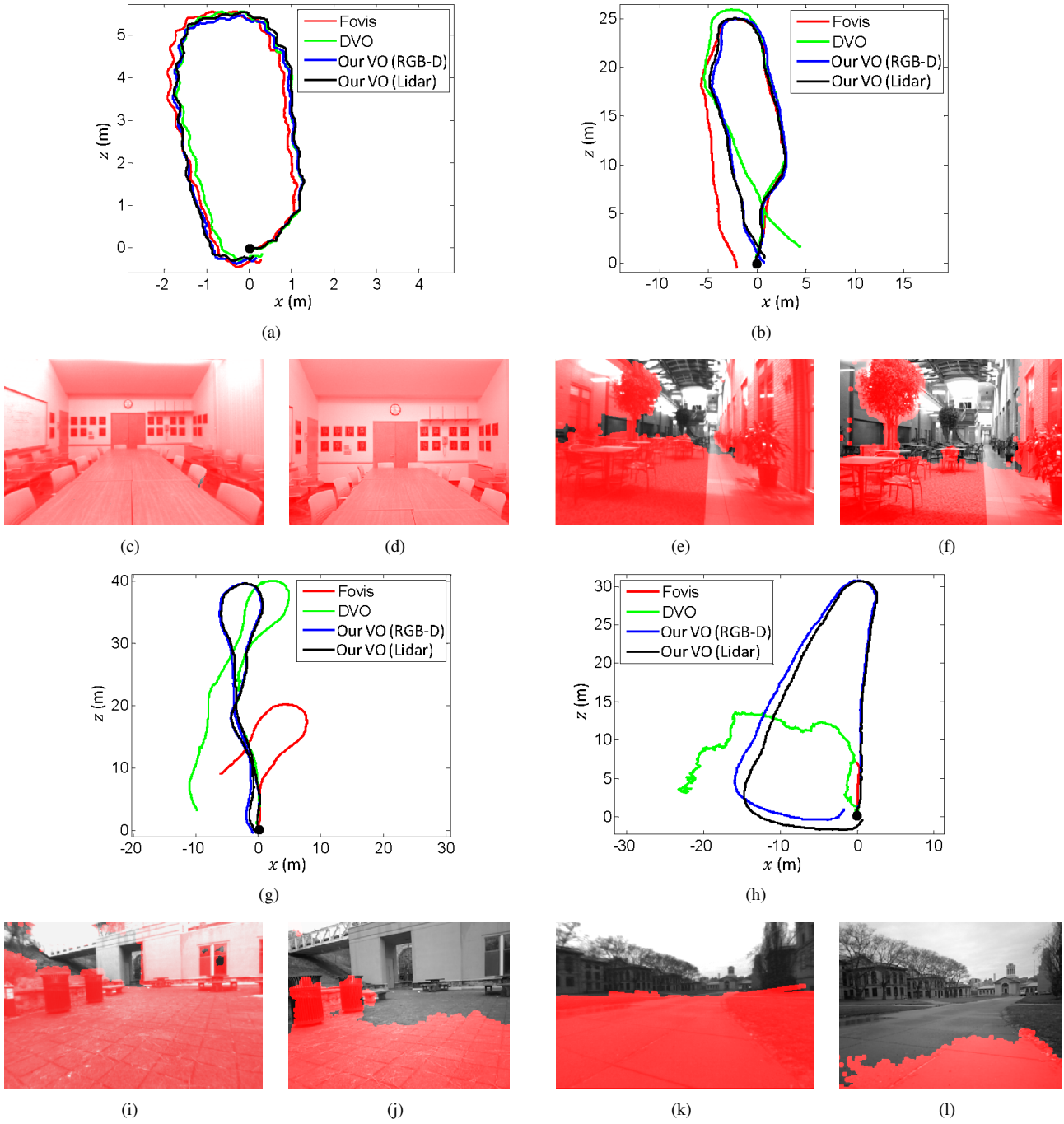


Fig. 6. Comparison of four methods using author-collected datasets: Fovis, DVO, and two versions of our method using depth from an RGB-D camera and a lidar. The environments are selected respectively from a conference room ((a), (c), and (d)), a large lobby ((b), (e), and (f)), a clustered road ((g), (i), and (j)), and a flat lawn ((h), (k), and (l)). We present two images from each dataset. The red colored areas indicate availability of depth maps, from the RGB-D camera (right figure) and the lidar (left figure). The depth information is sparser from each test to the next as the environment becomes opener, resulting in the performance of Fovis and DVO reduces significantly. Our methods relatively keep the accuracy in the tests.

are able to maintain accuracy in the tests, except that the relative error of the RGB-D camera version is relatively large in the lawn environment, because the depth is too sparse during the turning on top of Fig. 6(h).

B. Tests with KITTI Datasets

The proposed method is further tested with the KITTI datasets. The datasets are logged with sensors mounted on the top of a passenger vehicle, in road driving scenarios. The vehicle is equipped with color stereo cameras, monochrome stereo cameras, a 360° Velodyne laser scanner, and a high accuracy GPS/INS for ground truth. Both image and laser data are logged at 10Hz. The image resolution is around 1230×370 pixels, with 81° horizontal field of view. Our method uses the imagery from the left monochrome camera and the laser data, and tracks maximally 2400 features from 3×10 identical subregions in the images.

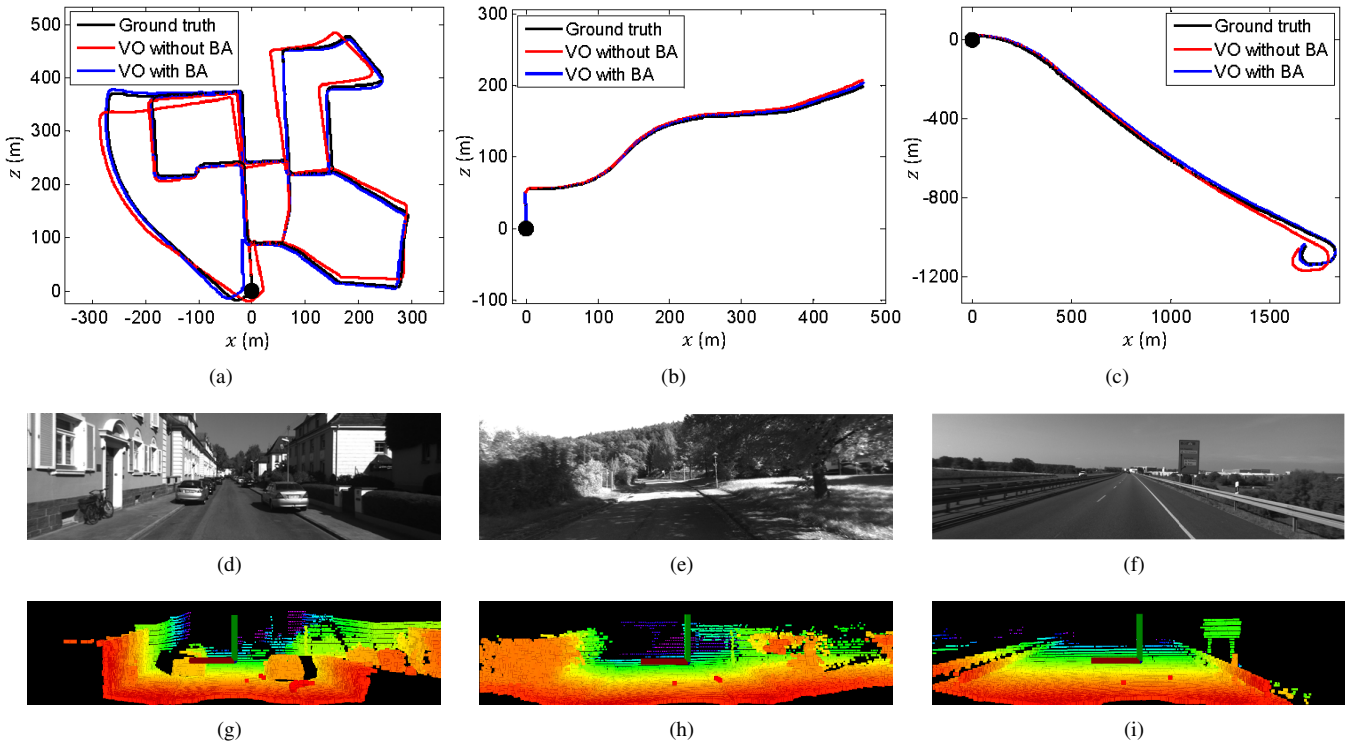


Fig. 7. Sample results of the proposed method using the KITTI datasets. The datasets are chosen from three types of environments: urban, country, and highway from left to right. In (a)-(c), we compare results of the method with and without the bundle adjustment, to the GPS/INS ground truth. The black colored dots are the starting points. An image is shown from each dataset to illustrate the three environments, in (d)-(f), and the corresponding laser point cloud in (g)-(i). The points are colored coded by depth, where red color indicates near objects and blue color indicates far objects.

TABLE II

CONFIGURATIONS AND RESULTS OF THE KITTI DATASETS. THE ERROR IS MEASURED USING SEGMENTS OF A TRAJECTORY AT 100M, 200M, ..., 800M LENGTHES, AS AN AVERAGED % OF THE SEGMENT LENGTHES.

Data no.	Configuration		Mean relative position error
	Distance	Environment	
0	3714m	Urban	1.05%
1	4268m	Highway	1.87%
2	5075m	Urban + Country	0.93%
3	563m	Country	0.99%
4	397m	Country	1.23%
5	2223m	Urban	1.04%
6	1239m	Urban	0.96%
7	695m	Urban	1.16%
8	3225m	Urban + Country	1.24%
9	1717m	Urban + Country	1.17%
10	919m	Urban + Country	1.14%

The datasets contain 11 tests with the GPS/INS ground truth provided. The data covers mainly three types of environments: "urban" with buildings around, "country" on small roads with vegetations in the scene, and "highway" where roads are wide and the surrounding environment is relatively clean. Fig. 7 presents sample results from the three environments. On the top row, the results of the proposed method are compared to the ground truth, with and without using the bundle adjustment introduced in Section VI. On the middle and bottom rows, an image and the corresponding laser point cloud is presented from each of the three datasets, respectively. The points are color coded by depth. The complete test results with the 11 datasets are listed in Table II. The three tests from left to right in Fig. 7 are respectively datasets 0, 3, and 1 in the table. Here, the accuracy is measured by averaging relative position errors using segments of a trajectory at 100m, 200m, ..., 800m lengthes, based on 3D coordinates. On the KITTI benchmark[2], our accuracy is comparable to state of the art stereo visual odometry [33], [34], which retrieve depth from stereo imagery without aid from the laser data.

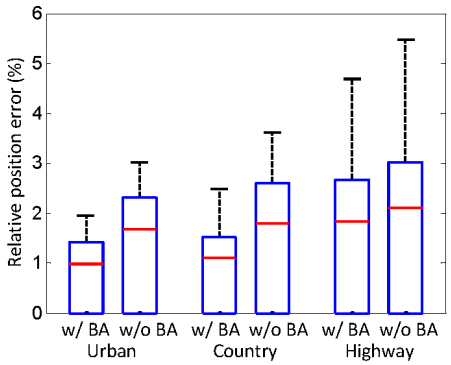


Fig. 8. Comparison of relative position errors in urban, country, and highway environments, tested with the KITTI datasets. In each environment, we compare errors with and without the bundle adjustment. The black, blue, and red colored lines indicate 100%, 75%, and median of the errors.

Further, to inspect the effect of the bundle adjustment, we compare accuracy of the results in the three environments.

[2]www.cvlibs.net/datasets/kitti/eval_odometry.php

The 11 datasets are manually separated into segments and labeled with an environment type. For each environment, the visual odometry is tested with and without the bundle adjustment. Fig. 8 shows the distributions of the relative errors. Overall, the bundle adjustment helps reduce the mean errors by 0.3%-0.7%, and seems to be more effective in urban and country scenes than on highways, partially because the feature quality is lower in the highway scenes.

VIII. Conclusion and Future Work

The scenario of insufficiency in depth information is common for RGB-D cameras and lidars which have limited ranges. Without sufficient depth, solving the visual odometry is hard. Our method handles the problem by exploring both visual features whose depth is available and unknown. The depth is associated to the features in two ways, from a depth map and by triangulation using the previously estimated motion. Further, a bundle adjustment is implemented which refines the frame to frame motion estimates. The method is tested with author-collected data using two sensors and the KITTI benchmark datasets. The results are compared to popular visual odometry methods, and the accuracy is comparable to state of the art stereo methods.

Considering future work, the current method uses Harris corners tracked by the KLT method. We experience difficulty of reliably tracking features in some indoor environments, such as a homogenously colored corridor. Improvement of feature detection and tracking is needed. Further, the method is currently tested with depth information from RGB-D cameras and lidars. In the future, we will try to utilize depth provided by stereo cameras, and possibly extend the scope of our method to stereo visual odometry.

References

[1] G. Klein and D. Murray, "Parallel tracking amd mapping for small AR workspaces," in *Proc. of the International Symposium on Mixed and Augmented Reality*, Nara, Japan, Nov. 2007, pp. 1–10.

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

[3] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *IEEE International Conference on Computer Vision (ICCV)*, Sydney, Australia, Dec. 2013.

[4] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2014.

[5] D. Scaramuzza, "1-point-ransac structure from motion for vehicle-mounted cameras by exploiting non-holonomic constraints," *International Journal of Computer Vision*, vol. 95, p. 7485, 2011.

[6] S. Weiss, M. Achtelik, S. Lynen, M. Achtelik, L. Kneip, M. Chli, and R. Siegwart, "Monocular vision for long-term micro aerial vehicle state estimation: A compendium," *Journal of Field Robotics*, vol. 30, no. 5, pp. 803–831, 2013.

[7] P. Corke, D. Strelow, and S. Singh, "Omnidirectional visual odometry for a planetary rover," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, Sept. 2004, pp. 149–171.

[8] K. Konolige, M. Agrawal, and J. Sol, "Large-scale visual odometry for rough terrain," *Robotics Research*, vol. 66, p. 201212, 2011.

[9] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The kitti vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.

[10] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *International Journal of Robotics Research*, no. 32, pp. 1229–1235, 2013.

[11] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry for ground vechicle applications," *Journal of Field Robotics*, vol. 23, no. 1, pp. 3–20, 2006.

[12] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the mars exploration rovers," *Journal of Field Robotics*, vol. 24, no. 2, pp. 169–186, 2007.

[13] A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles," in *IEEE International Conference on Intelligent Robots and Systems*, Nice, France, Sept 2008.

[14] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3D reconstruction in real-time," in *IEEE Intelligent Vehicles Symposium*, Baden-Baden, Germany, June 2011.

[15] L. Paz, P. Pinies, and J. Tardos, "Large-scale 6-DOF SLAM with stereo-in-hand," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 946–957, 2008.

[16] R. Newcombe, A. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *IEEE International Symposium on Mixed and Augmented Reality*, 2011, pp. 127–136.

[17] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard, "Real-time 3D visual SLAM with a hand-held RGB-D camera," in *RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum*, 2011.

[18] T. Whelan, H. Johannsson, M. Kaess, J. Leonard, and J. McDonald, "Robust real-time visual odometry for dense RGB-D mapping," in *IEEE International Conference on Robotics and Automation*, 2013.

[19] I. Dryanovski, R. Valenti, and J. Xiao, "Fast visual odometry and mapping from RGB-D data," in *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 2013.

[20] A. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Int. Symposium on Robotics Research (ISRR)*, 2011.

[21] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using kinect-style depth cameras for dense 3D modeling of indoor environments," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2012.

[22] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, Quebec City, Canada, June 2001.

[23] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, 2013.

[24] J. Sturm, E. Bylow, C. Kerl, F. Kahl, and D. Cremer, "Dense tracking and mapping with a quadrocopter," in *Unmanned Aerial Vehicle in Geomatics (UAV-g)*, Rostock, Germany, 2013.

[25] G. Hu, S. Huang, L. Zhao, A. Alempijevic, and G. Dissanayake, "A robust RGB-D slam algorithm," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura, Portugal, Oct. 2012.

[26] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. New York, Cambridge University Press, 2004.

[27] R. Murray and S. Sastry, *A mathematical introduction to robotic manipulation*. CRC Press, 1994.

[28] R. Andersen, "Modern methods for robust regression." *Sage University Paper Series on Quantitative Applications in the Social Sciences*, 2008.

[29] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computation Geometry: Algorithms and Applications, 3rd Edition*. Springer, 2008.

[30] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *International Journal of Robotics Research*, vol. 31, pp. 217–236, 2012.

[31] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of Imaging Understanding Workshop*, 1981, pp. 121–130.

[32] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: An open-source robot operating system," in *Workshop on Open Source Software (Collocated with ICRA 2009)*, Kobe, Japan, May 2009.

[33] A. Y. H. Badino and T. Kanade, "Visual odometry by multi-frame feature integration," in *Workshop on Computer Vision for Autonomous Driving (Collocated with ICCV 2013)*, Sydney, Australia, 2013.

[34] H. Badino and T. Kanade, "A head-wearable short-baseline stereo system for the simultaneous estimation of structure and motion," in *IAPR Conference on Machine Vision Application*, Nara, Japan, 2011.