Auton Robot (2017) 41:31–43
DOI 10.1007/s10514-015-9525-1

A real-time method for depth enhanced visual odometry

Ji Zhang[1] · **Michael Kaess**[1] · **Sanjiv Singh**[1]

Received: 1 March 2015 / Accepted: 17 November 2015 / Published online: 12 December 2015

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 structure from motion using the previously estimated motion, and salient visual features for which depth is unavailable. Therefore, the method is able to extend RGB-D visual odometry to large scale, open environments where depth often cannot be sufficiently acquired. The core of our method is a bundle adjustment step 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 #4 on the KITTI odometry benchmark irrespective of sensing modality—compared to stereo visual odometry methods which retrieve depth by triangulation. The resulting average position error is 1.14 % of the distance traveled.

Keywords Visual odometry · RGB-D · Range sensing

✉ Ji Zhang
zhangji@cmu.edu

Michael Kaess
kaess@cmu.edu

Sanjiv Singh
ssingh@cmu.edu

[1] Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA

1 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 (Klein and Murray 2007; Newcombe et al. 2011; Engel et al. 2013; Forster et al. 2014), 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 (Scaramuzza 2011), or measurements from an IMU integrated with the visual odometry (Weiss et al. 2013). However, the results are dependent on the quality of the extra information involved.

It is possible to obtain scale by using multiple cameras simultaneously (Corke et al. 2004; Konolige et al. 2011). 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. Effectively, only near features are used in recovery of the motion.

This paper proposes a method, namely DEMO, that can effectively utilize depth information along with the imagery. 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 (an example is shown in Fig. 1). Our method handles exactly this scenario—enhance motion estimation through usage of any depth information available.

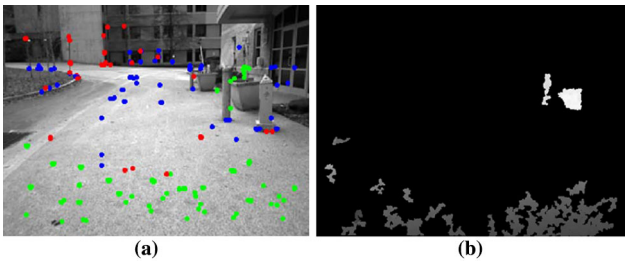


Fig. 1 **a** Features tracked at an image frame. The *green dots* represent features whose depth comes from the depth map, the *blue dots* represent features whose depth is determined by triangulation using the previously estimated motion of the camera, and the *red dots* represent features without depth. 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 (Color figure online)

Hence, we extend RGB-D visual odometry to large scale, open environments where depth often cannot be sufficiently acquired. 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 structure from motion using the previously estimated motion. With depth associated, we derive objective functions that minimize the distances from the features to their lateral projections onto the rays representing the features tracked at a consecutive frame. This makes the method extra sensitive to features at large angles off the camera periapical axis. Salient visual features for which depth is unavailable are also used, which provide different constraints in solving the problem. Further, the method contains a bundle adjustment step which refines the motion estimates in parallel by processing a sequence of images, in a batch optimization.

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 (Geiger et al. 2012, 2013), which contain carefully registered data from a number of sensors and ground truth from a high-accuracy GPS/INS. The method reported here uses images from a single camera in a stereo pair and laser scans from a high rate lidar. By submitting results to the benchmark server for which the ground truth is not provided, the accuracy and ranking are automatically calculated. The method is compared to various methods, including stereo/monocular visual odometry methods and RGB-D visual odometry methods that uses a camera and a lidar. The contributions of the paper are as follows,

- We propose a visual odometry framework that can utilize depth information from various sources to assist motion estimation;
- The proposed method is able to maximally utilize image information with and without depth coverage in motion estimation;
- The proposed method is tested thoroughly with a large number of datasets in various types of environments and in large scale;
- We make an honest attempt to present our work to a level of detail allowing readers to re-implement the method.

The rest of this paper is organized as follows. In Sect. 2, we discuss related work. In Sect. 3, we state the problem. The sensor hardware and software system are described in Sect. 4. The frame to frame motion estimation and bundle adjustment are discussed in Sects. 5 and 6. Experimental results are in Sect. 7 and conclusion in Sect. 8.

2 Related work

Vision based methods are now common for motion estimation (Nister et al. 2006; Maimone et al. 2007). To solve 6DOF camera motion, stereo vision systems are often used (Howard 2008; Geiger et al. 2011). 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, Konolige, at al.'s stereo visual odometry recovers the motion from bundle adjustment (Konolige et al. 2011). The method is integrated with an IMU and capable for long distance off-road navigation. Further, Paz et al. (2008) estimate the motion of stereo hand-hold cameras. The depth is recovered for features close to the cameras, which help solve scale of the translation estimation. Features far away from the camera are used to solve orientation only. Our method follows the same idea to utilize both near and far features. However, it solves for motion by an optimization based approach. Instead of classifying the features into two categories, we consider three scenarios where the depth is provided from the sensor, by structure from motion, and unavailable.

The introduction of RGB-D cameras has drawn great attention in the research of visual odometry (Newcombe et al. 2011; Engelhard et al. 2011; Whelan et al. 2013; Dryanovski et al. 2013). Such cameras provide RGB images along with depth information within the camera range limit. When depth information is available, motion estimation can be formulated into a 2D–3D matching problem Sattler et al. (2011). Huang et al. (2011) 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. (2012) integrate visual features with the iterative closest

point (ICP) method (Rusinkiewicz and Levoy 2001). The motion estimation employs a joint optimization by minimizing combined 2D and 3D feature matching errors. Another popular method is dense tracking (Kerl et al. 2013; Sturm et al. 2013). The method minimizes the photometric error using a dense 3D model of the environment from the depth images. Overall, the methods Huang et al. (2011), Henry et al. (2012), Kerl et al. (2013), Sturm et al. (2013) 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 in a near range of the camera. 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 (Hu et al. 2012).

In contrast to methods Huang et al. (2011), Henry et al. (2012), Kerl et al. (2013), Sturm et al. (2013), Hu et al. (2012), we propose a single method to handle sparse depth information. To the best of our knowledge, this is the first optimization based visual odometry method combining both features with and without depth in solving for motion. Further, since the method maintains and registers a depth map, it can use depth information from different types of sensors. The method is not limited to RGB-D cameras where depth is associated to visual images for each pixel, but also supports depth from lidars. As far as we know, no existing method in the literature has associated a lidar to a monocular camera for visual odometry estimation. This paper is an extended version of our conference paper Zhang et al. (2014). We address the method in more details and include more results in experiments.

3 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 (Hartley and Zisserman 2004), 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, $i \in \mathcal{I}$.

4 System overview

4.1 Sensor hardware

The proposed method is validated on, but not limited to, 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. Figure 2a is an Xtion Pro Live RGB-D camera. The camera is capable of providing RGB and depth images at 30 Hz, with 640 × 480 resolution and 58° horizontal field of view. Figure 2b shows a custom-built camera and 3D lidar. The camera can provide RGB images up to 60 Hz, 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/s scanning rate. The laser scanner is actuated by a motor for rotational motion to realize 3D scanning.

4.2 Software system overview

Figure 3 shows a diagram of the software system. First, visual features are tracked by the feature tracking block. Depth

Fig. 2 Two sensors involved in the evaluation. **a** An Xtion Pro Live RGB-D camera. The camera is capable of providing 30 Hz RGB and depth images, with 640 × 480 resolution and 58° HFV. **b** A custom-built camera and 3D lidar. The camera provides up to 60 Hz 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/s scanning rate

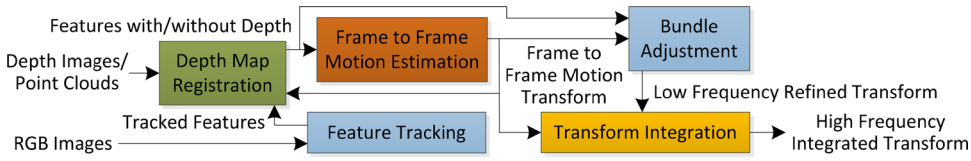


Fig. 3 Block diagram of the visual odometry software system

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 0.25–1.0 Hz). 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. Sections 5 and 6 present each block in detail.

5 Frame to frame motion estimation

5.1 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 associate depth for one of the two frames–frame $k-1$. This is because the depth maps are registered in the camera coordinates by the estimated motion. By the time of frame k, the depth map at frame $k-1$ is available and the depth of $\boldsymbol{X}_i^{k-1}$ can be associated. However, the depth map is unavailable to frame k as the last frame to frame motion has not been computed. This is especially true for the sensor configuration in Fig. 2b. Since laser points are perceived continuously over time, estimated motion is needed to register the points on the depth map. Further, our observation is that depth maps change little between consecutive frames and using that of one frame is sufficient. 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

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

where $\mathbf{R}$ and $\boldsymbol{T}$ form an $\mathbf{SE}(3)$ transformation Murray et al. (1994). Equation (2) contains three rows. Combining the 1st and 2nd rows with the 3rd row, respectively, we eliminate z_i^k as the unknown depth. 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 hth rows 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 $\boldsymbol{X}_i^{k-1}$,

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

Equation (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 the normalized term $\theta/||\theta||$ represents direction of the rotation axis and $||\theta||$ is the rotation angle between frames k and $k-1$. The rotation matrix $\mathbf{R}$ can be expressed by exponential map through Rodrigues formula (Murray et al. 1994),

$$\mathbf{R} = e^{\hat{\theta}} = \begin{cases} \mathbf{I} + \frac{\hat{\theta}}{||\theta||}\sin||\theta|| + \frac{\hat{\theta}^2}{||\theta||^2}(1 - \cos||\theta||) \\ \qquad \text{if } \theta \text{ is not a small angle,} \\ \mathbf{I} + \hat{\theta} + \frac{1}{2}\hat{\theta}^2 \qquad \text{otherwise,} \end{cases} \tag{7}$$

where $\hat{\theta}$ is the skew symmetric matrix of θ. Here, note that when θ is a small angle, we take the second-order Taylor expansion of the Rodrigues formula in (7) to avoid the problem of dividing by zero.

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,

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

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 $\mathbf{J} = \partial f/\partial[\boldsymbol{T};\ \theta]$. Equation (8) can be solved by the Levenberg–Marquardt (LM) method (Hartley and Zisserman 2004),

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

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

The proposed method is a sparse feature based method. Features are tracked regardless of depth information being available. The tracked features are then formulated into different equations according to the depth information. In comparison, direct methods such as Kerl et al. (2013) and Sturm et al. (2013) employ overall image warping in the depth available areas, eliminating the need to process individual features. However, as discussed in Forster et al. (2014), sparse feature based methods can produce better accuracy. The method Forster et al. (2014), therefore, performs direct image matching to generate initial guess followed by sparse motion estimation to warrant accuracy.

Algorithm 1: Frame to Frame Motion Estimation

1 **input** : $\bar{\boldsymbol{X}}_i^k$, $\boldsymbol{X}_i^{k-1}$ or $\bar{\boldsymbol{X}}_i^{k-1}$, $i \in \mathscr{I}$
2 **output** : θ, $\boldsymbol{T}$
3 **begin**
4 $\theta, \boldsymbol{T} \leftarrow$ initialization based on a constant velocity model or zero at the first frame;
5 **for** a number of iterations **do**
6 **for** each $i \in \mathscr{I}$ **do**
7 **if** i is depth associated **then**
8 Derive (3)-(4) using $\bar{\boldsymbol{X}}_i^k$ and $\boldsymbol{X}_i^{k-1}$ as functions of θ and $\boldsymbol{T}$,
$(3) : (\boldsymbol{R}_1 - \bar{x}_i^k\boldsymbol{R}_3)\boldsymbol{X}_i^{k-1} + T_1 - \bar{x}_i^k T_3 = 0$,
$(4) : (\boldsymbol{R}_2 - \bar{y}_i^k\boldsymbol{R}_3)\boldsymbol{X}_i^{k-1} + T_2 - \bar{y}_i^k T_3 = 0$;
9 Stack (3)-(4) into $(8) : \boldsymbol{f}([\boldsymbol{T};\ \theta]) = \varepsilon$;
10 **end**
11 **else**
12 Derive (6) using $\bar{\boldsymbol{X}}_i^k$ and $\bar{\boldsymbol{X}}_i^{k-1}$ as a function of θ and $\boldsymbol{T}$,
$(6) : [-\bar{y}_i^k T_3 + T_2,\ \bar{x}_i^k T_3 - T_1,\ -\bar{x}_i^k T_2 + \bar{y}_i^k T_1]$
$\mathbf{R}\bar{\boldsymbol{X}}_i^{k-1} = 0$;
13 Stack (6) into $(8) : \boldsymbol{f}([\boldsymbol{T};\ \theta]) = \varepsilon$;
14 **end**
15 **end**
16 Compute a bisquare weight for each feature based on the residuals in $(8) : \boldsymbol{f}([\boldsymbol{T};\ \theta]) = \varepsilon$;
17 Update θ, $\boldsymbol{T}$ for one iteration based on,
$(9) : [\boldsymbol{T};\ \theta]^T \leftarrow [\boldsymbol{T};\ \theta]^T - (\mathbf{J}^T\mathbf{J} + \lambda\mathrm{diag}(\mathbf{J}^{\mathrm{T}}\mathbf{J}))^{-1}\mathbf{J}^{\mathrm{T}}\varepsilon$
18 **if** the inlier ratio is smaller than a threshold **then**
19 Report the frame is skipped and return;
20 **end**
21 **if** the nonlinear optimization converges **then**
22 Break;
23 **end**
24 **end**
25 Return θ, $\boldsymbol{T}$;
26 **end**

5.2 Motion estimation algorithm

The frame to frame motion estimation algorithm is presented in Algorithm 1. We do not implement key framing as we seek for the maximum number of features tracked between consecutive frames. This helps provide more constraints in the bundle adjustment step which will be discussed in Sect. 6. However, if not enough inliers can be found, for example due to moving objects in the scene, the frame is skipped. As we have mentioned that the proposed method only uses depth information associated at frame $k-1$, all features taken as the input at frame k are 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 12, a feature with unknown depth contributes one equation.

We use a constant velocity model for initialization of θ and $\boldsymbol{T}$ on line 4. Optionally, the terms are initialized to zero at the first frame. The algorithm is adapted to a robust fitting framework (Andersen 2008). On line 16, the algorithm assigns a bisquare weight for each feature based on their residuals in (8). 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 17, θ and $\boldsymbol{T}$ are updated for one iteration. The nonlinear optimization terminates if convergence is found, or the maximum iteration number is met. The nonlinear optimization also terminates if the ratio of the inlier features is lower than a threshold, which lead to neglecting of the current frame. Finally, the algorithm returns the motion estimation θ and $\boldsymbol{T}$.

5.3 Feature depth association

In this section, we discuss how to associate depth to the visual features. As illustrated in Fig. 4, 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.

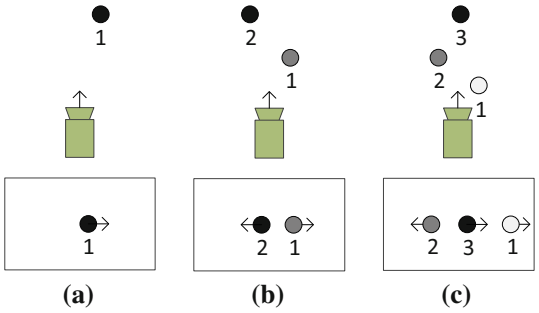


Fig. 4 Illustration of depth map registration. **a–c** represent three image frames. In each figure, we show the camera frame on the *top* and the corresponding image on the *bottom*. Three points are registered labeled with 1–3. The illumination of a point indicates its distance to the camera. In this example, the camera moves forward. The registered points change illumination during the camera motion, leave the camera field of view in the end and are removed from the depth map

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. 5, color coded by elevation. The point clouds are collected by the lidar in Fig. 2b, while the camera points to a wall.

To associate depth to the visual features, we store the depth map in a 2D KD-tree (de Berg et al. 2008) based on the two angular coordinates. As illustrated in Fig. 6, this equals to projecting points on the depth map onto a sphere with a unit distance to the camera center. For each feature i, $i \in \mathcal{I}$, as illustrated by the orange point, 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 as follows,

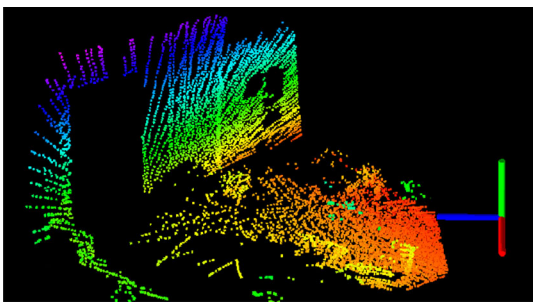


Fig. 5 An example of the depth map with point clouds perceived by the lidar in Fig. 2b. The *points* are color coded by elevation. The camera points to a wall during data collection

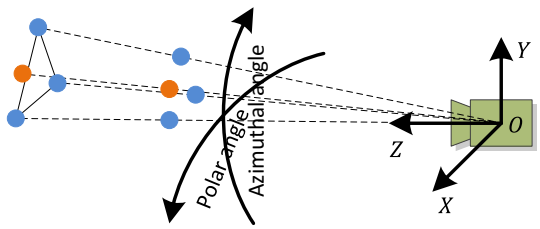


Fig. 6 Illustration of depth association. We represent points on the depth map by spherical coordinate representation. The points are stored in a 2D KD-tree based on the azimuthal angle and polar angle as illustrated in the figure. This equals to projecting the points onto a sphere with a unit distance to the camera center. For each feature (indicated by the *orange point*), we find the three closest points from the KD-tree. The depth of the feature is then interpolated from the three points assuming a local planar patch formed by the three points in the 3D environment (Color figure online)

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

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 image sequences where the features are tracked. This uses a similar procedure as Vogiatzis and Hernandez (2011), Forster et al. (2014), where the depth of a feature is updated based on a Bayesian probabilistic model each time the feature is tracked for one more frame. Figure 7 gives an example of depth associated features, corresponding to Fig. 1. The white dots are points on the depth map, only available within a limited range. The green dots represent features whose depth are provided by the depth map, and the blue dots are from structure from motion.

6 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.

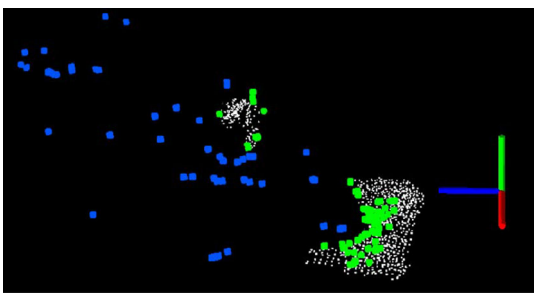


Fig. 7 Features projected into the 3D environment corresponding to Fig. 1. The depth map (*white dots*) is from depth images collected by the RGB-D camera in Fig. 2a, only available within a limited range. The *green dots* are features whose depth is provided by the depth map, and the *blue dots* are from triangulation using the previously estimated motion (Color figure online)

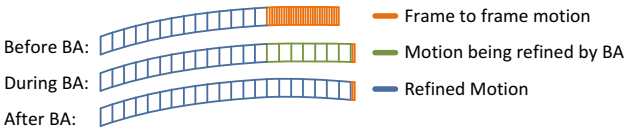


Fig. 8 Illustration of bundle adjustment. The *orange segment* represents frame to frame motion, the *green segment* represents the motion being refined by bundle adjustment, and the *blue segment* is refined motion. Each *vertical bar* indicates an image frame. On the *top row*, the system accumulates frame to frame motion (*orange*) in front of the refined motion (*blue*). When enough image frames are accumulated, on the *second row*, bundle adjustment is executed refining the green segment. After the bundle adjustment, on the *third row*, the *green segment* becomes refined motion. The system keeps accumulating frame to frame motion (*orange*) for the next bundle adjustment to execute (Color figure online)

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 process is illustrated in Fig. 8, using the iSAM (Kaess et al. 2012) open source library. We choose iSAM over other libraries because it supports user-defined camera models, and can conveniently handle both features with and without available depth.

In this step, outlier features are already eliminated in the frame to frame motion estimation. All selected inlier features are used in the bundle adjustment. While bundle adjustment itself is time-consuming, skipping outlier removal helps reduce processing time. In the image sequence, the first fame is fixed. The bundle adjustment refines every frame thereafter and the 3D feature poses along the sequence.

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

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

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 representing the feature tracking errors. If the depth is from the depth map which is perceived by a time-of-fly sensor such as the lidar in Fig. 2b, the 3rd entry is set at a fixed value determined by the measurement noise of the sensor. If the depth map is reconstructed by triangulation such as using the RGB-D camera in Fig. 2a, or if the depth is from structure from motion using the estimated motion, the 3rd entry is set to be inversely proportional to the square of the depth. A zero value is used for features with unknown depth. The information matrices associated with the frame to frame poses are set to small and negligible values, meaning that poses from the frame to frame motion estimation only provide initial guess to the bundle adjustment.

The bundle adjustment publishes refined motion transforms at a low frequency. With the camera frame rate between 10–40 Hz, the bundle adjustment runs at 0.25–1.0 Hz. As illustrated in Fig. 8, a transform integration step takes the bundle adjustment output (blue segment) and combines it with the high frequency frame to frame motion estimates (orange segment). The result is integrated motion transforms at the frame to frame motion frequency.

7 Experiments

The visual odometry is tested with author-collected data and the KITTI benchmark datasets. It tracks Harris corners (Hartley and Zisserman 2004) by the Kanade Lucas Tomasi (KLT) method (Lucas and Kanade 1981). The program is implemented in C++ language, on top of the robot operating system (ROS) (Quigley et al. 2009) in Linux. The algorithms run on a laptop computer with 2.5 GHz cores and 6 GB memory, using around three cores for computation. The feature tracking and bundle adjustment (Sect. 6) take one core each, and the frame to frame motion estimation (Sect. 5) and depth map registration together consume another core.

7.1 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. 9: 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. 9. The red 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

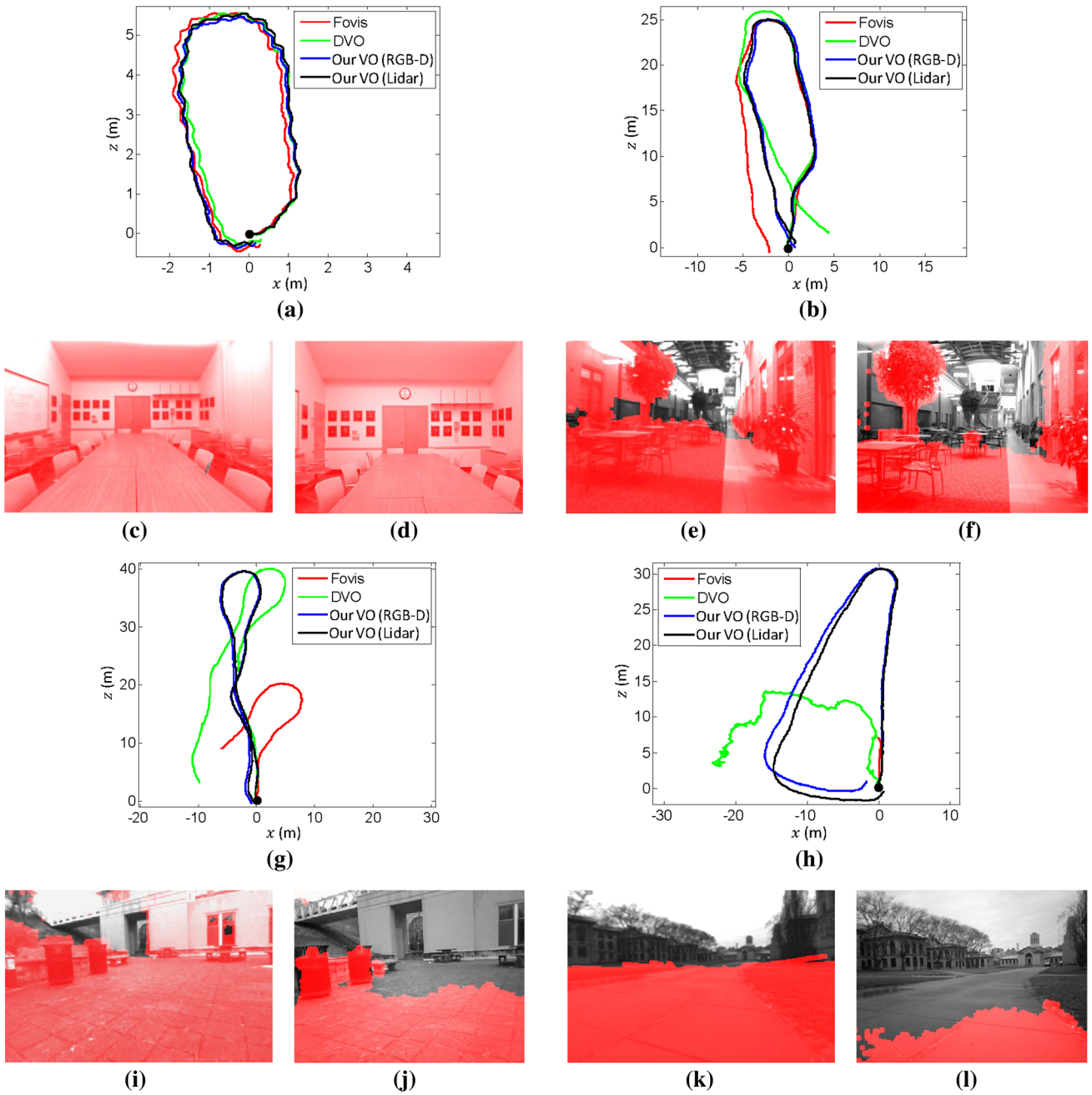


Fig. 9 Comparison of four methods using author-collected datasets: Fovis, DVO, and two versions of our method using depth from an RGB-D camera and a lidar. The environments are selected respectively from a conference room (**a**, **c**, **d**), a large lobby (**b**, **e**, **f**), a clustered road (**g**, **i**, **j**), and a flat lawn (**h**, **k**, **l**). We present two images from each dataset. The *red 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 (Color figure online)

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 30 Hz 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 (Huang et al. 2011). DVO is a dense tracking method that minimizes the photo-

Table 1 Results using author-collected data

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

The error is measured at the end of a trajectory as a % of the distance traveled

metric error within the overall images (Kerl et al. 2013). 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. 9, and the accuracy is compared in Table 1, 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 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. 9h.

7.2 Tests with KITTI benchmark 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. As shown in Fig. 10, 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.

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

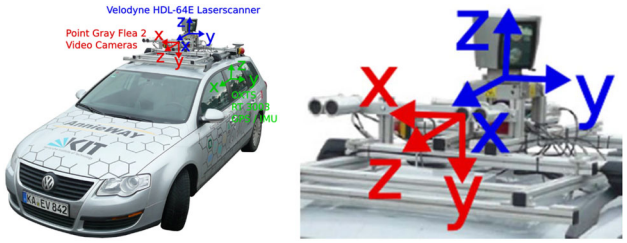


Fig. 10 **a** Vehicle used by the KITTI benchmark for data logging. The vehicle is mounted with color stereo cameras, monochrome stereo cameras, a Velodyne lidar, and a high accuracy GPS/INS for ground truth acquisition. Our method uses data from a single camera and the Velodyne lidar for motion estimation. **b** shows a zoomed in view of the sensors (Color figure online)

roads with vegetations in the scene, and "highway" where roads are wide and the surrounding environment is relatively clean. Figure 11 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 Sect. 6. 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 2. The three tests from left to right in Fig. 11 are respectively datasets 0, 3, and 1 in the table. Here, the accuracy is measured by averaging relative translation and rotation errors using segments at 100m, 200m, ..., 800m lengthes, based on 3D coordinates.

We analyze the results using the overall 11 datasets in Fig. 12. On the first row, we show translation errors as percentages of the traveled distances. We plot the errors with respect to the lengthes of the data segments in Fig. 12a, the linear speed of the vehicle in Fig. 12b, and the angular speed in Fig. 12c. On the second row, we present the corresponding rotation errors with respect the three terms.

And on the third row, we show distributions of the data used in calculation of the errors. Looking at the left column, we can obversely see that the translation and rotation errors are decreasing functions of the segment lengthes. We explain this as an effect of high frequency noise in the visual

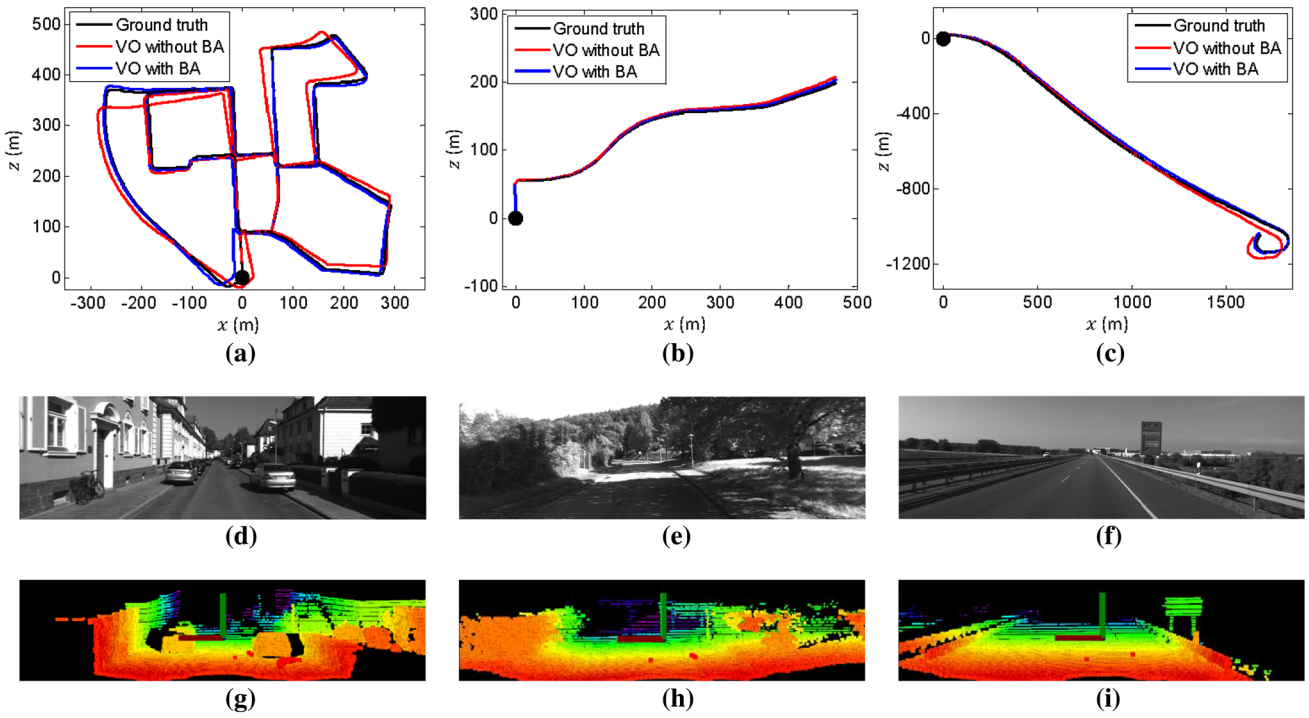


Fig. 11 Sample results of the proposed method using the KITTI datasets. The datasets are chosen from three types of environments: urban, country, and highway from left to right. In **a**–**c**, we compare results of the method with and without the bundle adjustment, to the GPS/INS ground truth. The *black 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 coded by depth, where *red color* indicates near objects and *blue color* indicates far objects (Color figure online)

Table 2 Configurations and results of the KITTI datasets

Data	Configuration		Error	
no.	Distance (m)	Environment	Trans. (%)	Rotation (°/s)
0	3714	Urban	1.05	0.0029
1	4268	Highway	1.87	0.0026
2	5075	Urban + country	0.93	0.0038
3	563	Country	0.99	0.0043
4	397	Country	1.23	0.0048
5	2223	Urban	1.04	0.0034
6	1239	Urban	0.96	0.0029
7	695	Urban	1.16	0.0043
8	3225	Urban + country	1.24	0.0047
9	1717	Urban + country	1.17	0.0035
10	919	Urban + country	1.14	0.0064

The translation error is calculated using segments of a trajectory at 100, 200,…, 800 m lengthes, as an averaged % of the segment lengthes based on 3D coordinates

odometry output and ground truth. Using the middle row, we perceive a trend that the translation error is an increasing function of the linear speed, however, the rotation error is an decreasing function. The exception is in the left-most part of Fig. 12b, which can possibly be caused by sparsity of data as indicated in the corresponding part of Fig. 12h. It is understandable that the translation error increases as the vehicle drivers faster. Our explanation of the fact that the rotation error decreases accordingly is that most rotations happen when the vehicle drives slowly, and the vehicle drives mostly straight at high speeds. In the right column, we see both translation and rotation errors as increasing func-

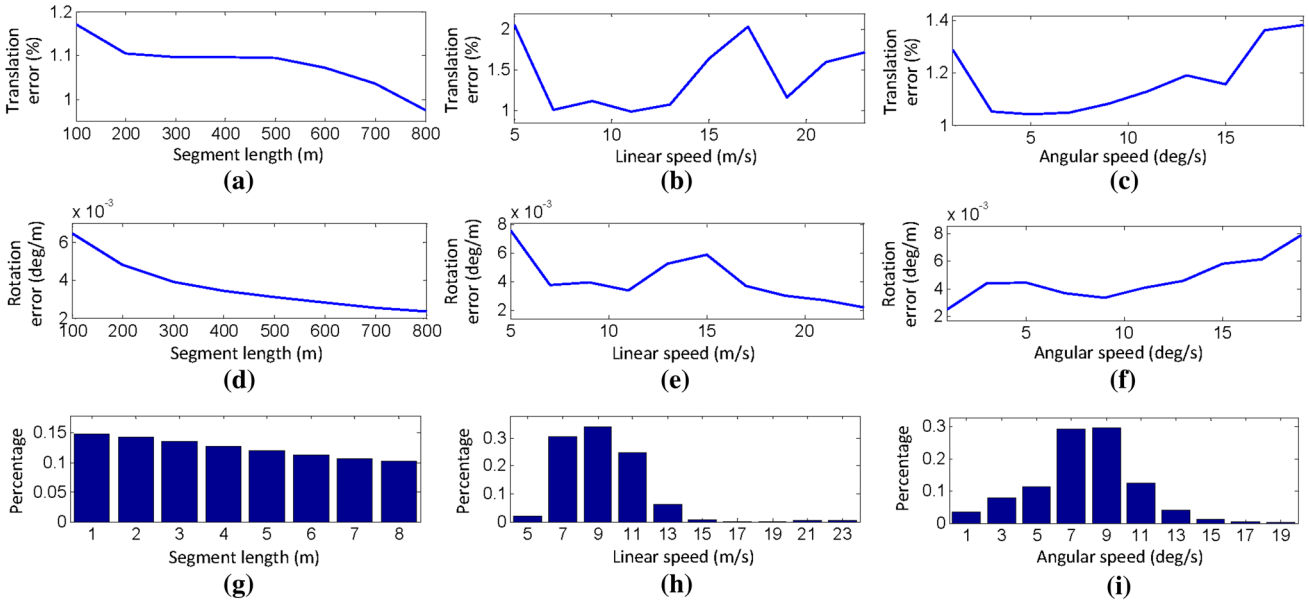


Fig. 12 Analysis of accuracy with the KITTI datasets. On the *top* and *middle rows*, we show translation and rotation errors, by averaging errors in the 11 datasets listed in Table 2. The translation errors are represented as fractions of the distance traveled, using data segments in 100, 200,…, 800 m lengthes based on 3D coordinates. On the *bottom row*, we present distributions of the data used in calculation of the errors. In the *left*, *middle*, and *right columns*, the errors are shown with respect to the segment length, linear speed, and angular speed, respectively

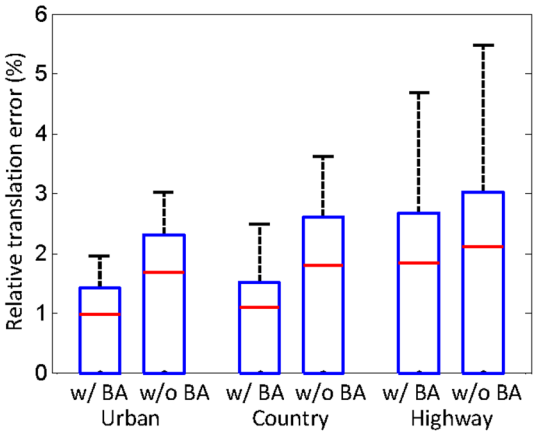


Fig. 13 Comparison of relative translation errors in urban, country, and highway environments, tested with the KITTI datasets. In each environment, we compare errors with and without the bundle adjustment. The *black*, *blue*, and *red lines* indicate 100, 75 %, and median of the errors (Color figure online)

tions of the angular speed (again, with an exception in the left most part of Fig. 12c, possibly due to sparsity of data), indicating that the motion estimation is less accurate when more turnings occur.

Further, to inspect the effect of the bundle adjustment, we compare accuracy of the results in the three environments. 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. Figure 13 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.

Finally, we compare results using the top 5 ranked methods on the KITTI odometry benchmark[1] in Table 3. This is using datasets 11–21 where the GPS/INS ground truth is not provided. By uploading results to the benchmark server, the accuracy and ranking are automatically calculated. Our method, namely DEMO, is ranked #4 among the overall evaluated methods by the benchmark, irrespective of sensing modality. In comparison, the #1 ranked method, V-LOAM (Zhang and Singh 2015), uses the same frame to frame motion estimation with DEMO. However, instead of using a bundle adjustment step for refining motion estimates, a scan matching step is used. Hence, a higher accuracy is achieved. The #2 ranked method, LOAM (Zhang and Singh 2014) does not use vision and completely relies on scan matching. The #3 method, CV4X (reference unavailable yet), and the #5 method, MFI (Badino and Kanade 2013, 2011), are both stereo visual odometry methods. In particular, CV4X relies on GPU for feature matching while motion estimation runs on CPU. Carefully comparing the five methods, we find that the rotation accuracy of DEMO is lower than the other four. It is understandable that scan matching (V-LOAM and LOAM) based methods deliver low rotation drift because the error of a laser scanner is irrespective of the distance measured and

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

Table 3 Configurations and results of the top five ranked method on the KITTI odometry benchmark

Rank	Method	Configuration	Error	
			Trans. (%)	Rotation (°/s)
1	V-LOAM	Monocular + laser	0.75	0.0018
2	LOAM	Laser	0.88	0.0022
3	CV4X	Stereo	1.09	0.0029
4	DEMO	Monocular + laser	1.14	0.0049
5	MFI	Stereo	1.30	0.0030

The errors are calculated in the same way as Table 2, but by the benchmark server using datasets 11–21 where the ground truth is not provided. Our method is ranked #4 among the overall evaluated methods

distanced points help correct rotation error when matching the laser points. The hypothesis that stereo methods outperform DEMO is that stereo cameras (with horizontal baseline) provide tighter constrains in yaw rotation estimation as one feature is seen by both cameras in a stereo pair but by only a monocular camera in DEMO.

8 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 visual odometry methods.

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.

Acknowledgements The paper is based upon work supported by the National Science Foundation under Grant No. IIS-1328930. Special thanks are given to S. Scherer, M. Bergerman, D. Huber, S. Nuske, Z. Fang, and D. Maturana for their insightful inputs.

Compliance with Ethical Statement

Conflict of Interest All authors are affiliated with Carnegie Mellon University. Author Michael Kaess was affiliated with Massachusetts Institute of Technology within the last three years. Michael Kaess also serves as an associate editor for IEEE Transactions on Robotics. Author Sanjiv Singh is the editor in chief of Journal of Field Robotics.

Funding This study is funded by National Science Foundation (Grant No. IIS-1328930).

References

Andersen, R. (2008). Modern methods for robust regression.*Sage University paper series on quantitative applications in the social sciences*.

Badino, H., & Kanade, T. (2011). A head-wearable short-baseline stereo system for the simultaneous estimation of structure and motion. In *IAPR conference on machine vision application, Nara*.

Badino, A.Y.H., & Kanade, T. (2013). Visual odometry by multi-frame feature integration. In *Workshop on computer vision for autonomous driving (collocated with ICCV 2013), Sydney*.

Corke, P., Strelow, D., & Singh, S. (2004). Omnidirectional visual odometry for a planetary rover. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems, Sendai* (pp. 149–171).

de Berg, M., M., Cheong, O., van Kreveld, M., & Overmars, M. (2008). *Computation geometry: Algorithms and applications* (3rd ed.). New York: Springer.

Dryanovski, I., Valenti, R., Xiao, J. (2013). Fast visual odometry and mapping from RGB-D data. In *IEEE international conference on robotics and automation (ICRA), Karlsruhe*.

Engel, J., Sturm, J., Cremers, D. (2013). Semi-dense visual odometry for a monocular camera. In *IEEE international conference on computer vision (ICCV), Sydney*.

Engelhard, N., Endres, F., Hess, J., Sturm, J., & Burgard, W. (2011). 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, Vasteras*.

Forster, C., Pizzoli, M., & Scaramuzza, D. (2014). SVO: Fast semi-direct monocular visual odometry. In *IEEE international conference on robotics and automation (ICRA), Hong Kong*.

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? The kitti vision benchmark suite. In *IEEE conference on computer vision and pattern recognition* (pp. 3354–3361).

Geiger, A., Ziegler, J.,& Stiller, C. (2011). Stereoscan: Dense3D reconstruction in real-time. In *IEEE intelligentvehicles symposium, Baden-Baden*.

Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research*, *32*, 1229–1235.

Hartley, R., & Zisserman, A. (2004). *Multiple view geometry in computer vision*. New York: Cambridge University Press.

Henry, P., Krainin, M., Herbst, E., Ren, X., & Fox, D. (2012). RGB-D mapping: Using kinect-style depth cameras for dense 3D modeling of indoor environments. *The International Journal of Robotics Research*, *31*(5), 647–663.

Howard, A. (2008). Real-time stereo visual odometry for autonomous ground vehicles. In *IEEE international conference on intelligent robots and systems, Nice*.

Hu, G., Huang, S., Zhao, L., Alempijevic, A., & Dissanayake, G. (2012) A robust RGB-D slam algorithm. In *IEEE/RSJ international conference on intelligent robots and systems, Vilamoura*.

Huang, A., Bachrach, A., Henry, P., Krainin, M., Maturana, D., Fox, D., & Roy, N. (2011). Visual odometry and mapping for autonomous flight using an RGB-D camera. In *International symposium on robotics research (ISRR), Flagstaff*.

Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J., & Dellaert, F. (2012). iSAM2: Incremental smoothing and mapping using the Bayes tree. *International Journal of Robotics Research*, *31*, 217–236.

Kerl, C., Sturm, J., & Cremers, D. (2013). Robust odometry estimation for RGB-D cameras. In *IEEE international conference on robotics and automation, Karlsruhe*.

Klein, G., & Murray, D. (2007). Parallel tracking amd mapping for small AR workspaces. In *Proceedings of the international symposium on mixed and augmented reality, Nara* (pp. 1–10).

Konolige, K., Agrawal, M., & Sol, J. (2011). Large-scale visual odometry for rough terrain. *Robotics Research*, *66*, 201–212.

Lucas, B.,& Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proceedings of imaging understanding workshop* (pp. 121–130).

Maimone, M., Cheng, Y., & Matthies, L. (2007). Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics*, *24*(2), 169–186.

Murray, R. M., Li, Z., & Sastry, S. S. (1994). *A mathematical introduction to robotic manipulation*. Boca Raton: CRC Press.

Newcombe, R., Davison, A., Izadi, S., Kohli, P., Hilliges, O., Shotton, J., Molyneaux, D., Hodges, S., Kim, D., & Fitzgibbon, A. (2011). KinectFusion: Real-time dense surface mapping and tracking. In *IEEE international symposium on mixed and augmented reality* (pp. 127–136).

Newcombe, R.A., Lovegrove, S.J.,& Davison, A.J. (2011). DTAM: Dense tracking and mapping in real-time. In *IEEE International Conference on Computer Vision*, 2011, (pp. 2320–2327).

Nister, D., Naroditsky, O., & Bergen, J. (2006). Visual odometry for ground vechicle applications. *Journal of Field Robotics*, *23*(1), 3–20.

Paz, L., Pinies, P., & Tardos, J. (2008). Large-scale 6-DOF SLAM with stereo-in-hand. *IEEE Transactions on Robotics*, *24*(5), 946–957.

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., & Ng, A. (2009). ROS: An open-source robot operating system. In *Workshop on open source software (collocated with ICRA 2009), Kobe*.

Rusinkiewicz, S.,& Levoy, M. (2001). Efficient variants of the ICP algorithm. In *Third international conference on 3D digital imaging and modeling (3DIM), Quebec City*.

Sattler, T., Leibe, B., & Kobbelt, L. (2011). Fast image-based localization using direct 2D-to-3D matching. In *IEEE international conference on computer vision (ICCV), Barcelona*.

Scaramuzza, D. (2011). 1-point-ransac structure from motion for vehicle-mounted cameras by exploiting non-holonomic constraints. *International Journal of Computer Vision*, *95*, 74–85.

Sturm, J., Bylow, E., Kerl, C., Kahl, F.,& Cremer, D. (2013). Densetracking and mapping with a quadrocopter. In *Unmanned aerialvehicle in geomatics (UAV-g), Rostock*.

Vogiatzis, G., & Hernandez, C. (2011). Video-based, real-time multiview stereo. *Image and Vision Computing*, *29*(7), 434–441.

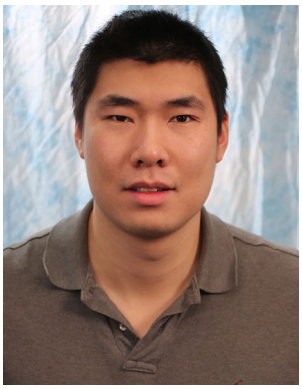
Weiss, S., Achtelik, M., Lynen, S., Achtelik, M., Kneip, L., Chli, M., et al. (2013). Monocular vision for long-term micro aerial vehicle state estimation: A compendium. *Journal of Field Robotics*, *30*(5), 803–831.

Whelan, T., Johannsson, H., Kaess, M., Leonard, J.,& McDonald, J. (2013). Robust real-time visual odometry for dense RGB-D mapping. In *IEEE international conference on robotics and automation, Karlsruhe*.

Zhang, J., Kaess, M., & Singh, S. (2014). Real-time depth enhanced monocular odometry. In *IEEE/RSJ international conference on intelligent robots and systems (IROS), Chicago*.

Zhang, J.,& Singh, S. (2014). LOAM: Lidar odometry and mapping in real-time. In *Robotics: Science and systems conference (RSS), Berkeley*.

Zhang, J.,& Singh, S. (2015). Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *Submitted to IEEE international conference on robotics and automation (ICRA), Seattle*.



Ji Zhang is a Ph.D. candidate at the Robotics Institute of Carnegie Mellon University. His research interest is focused on robot navigation, perception and localization, lidar mapping, and computer vision.

Michael Kaess received the M.S. and Ph.D. degrees in Computer Science from the Georgia Institute of Technology in 2002 and 2008, respectively. He was a postdoctoral associate and research scientist at the Massachusetts Institute of Technology (MIT) from 2008 to 2013. He is now an Assistant Research Professor at the Robotics Institute, Carnegie Mellon University. His research interests include navigation, localization, mapping and efficient inference. He is currently an associate editor for the IEEE Transactions on Robotics.

Sanjiv Singh is a Research Professor at the Robotics Institute with a courtesy appointment in Mechanical Engineering. He is the founding editor of the Journal of Field Robotics. His research relates to the operation of robots in natural and in some cases, extreme environments. His recent work has two main themes: perception in natural and dynamic environments and multi-agent coordination. Currently, he leads efforts in collision avoidance for air vehicles (near earth and between aircraft) and ground vehicles using novel sensors that can look through obscurants. Another research area seeks to provide accurate tracking and situational awareness in dynamic environments, such as those encountered in search and rescue, using radio signals to compute location. This research seeks to assist emergency response personnel in their operations.