2015 IEEE International Conference on Robotics and Automation (ICRA)
Washington State Convention Center
Seattle, Washington, May 26-30, 2015

Visual-lidar Odometry and Mapping: Low-drift, Robust, and Fast

Ji Zhang and Sanjiv Singh

Abstract— **Here, we present a general framework for combining visual odometry and lidar odometry in a fundamental and first principle method. The method shows improvements in performance over the state of the art, particularly in robustness to aggressive motion and temporary lack of visual features. The proposed on-line method starts with visual odometry to estimate the ego-motion and to register point clouds from a scanning lidar at a high frequency but low fidelity. Then, scan matching based lidar odometry refines the motion estimation and point cloud registration simultaneously. We show results with datasets collected in our own experiments as well as using the KITTI odometry benchmark. Our proposed method is ranked #1 on the benchmark in terms of average translation and rotation errors, with a 0.75% of relative position drift. In addition to comparison of the motion estimation accuracy, we evaluate robustness of the method when the sensor suite moves at a high speed and is subject to significant ambient lighting changes.**

I. Introduction

Recent separate results in visual odometry and lidar odometry are promising in that they can provide solutions to 6-DOF state estimation, mapping, and even obstacle detection. However, drawbacks are present using each sensor alone. Visual odometry methods require moderate lighting conditions and fail if distinct visual features are insufficiently available. On the other hand, motion estimation via moving lidars involves motion distortion in point clouds as range measurements are received at different times during continuous lidar motion. Hence, the motion often has to be solved with a large number of variables. Scan matching also fails in degenerate scenes such as those dominated by planar areas.

Here, we propose a fundamental and first principle method for ego-motion estimation combining a monocular camera and a 3D lidar. We would like to accurately estimate the 6-DOF motion as well as a spatial, metric representation of the environment, in real-time and onboard a robot navigating in an unknown environment. While cameras and lidars have complementary strengths and weaknesses, it is not straightforward to combine them in a traditional filter. Our method tightly couples the two modes such that it can handle both aggressive motion including translation and rotation, and lack of optical texture as in complete whiteout or blackout imagery. In non-pathological cases, high accuracy in motion estimation and environment reconstruction is possible.

Our proposed method, namely V-LOAM, explores advantages of each sensor and compensates for drawbacks from the other, hence shows further improvements in performance over the state of the art. The method has two sequentially staggered processes. The first uses visual odometry running at a high frequency as the image frame rate (60Hz) to estimate motion. The second uses lidar odometry at a low frequency (1 Hz) to refine motion estimates and remove distortion in the point clouds caused by drift of the visual odometry. The distortion-free point clouds are matched and registered to incrementally build maps. The result is that the visual odometry handles rapid motion, and the lidar odometry warrants low-drift and robustness in undesirable lighting conditions. Our finding is that the maps are often accurate without the need for post-processing. Although loop closure can further improve the maps, we intentionally choose not to do so since the emphasis of this work is to push the limit of accurate odometry estimation.

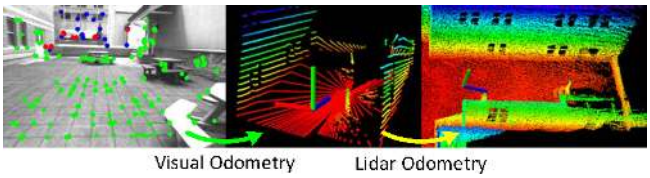


Fig. 1. The method aims at motion estimation and mapping using a monocular camera combined with a 3D lidar. A visual odometry method estimates motion at a high frequency but low fidelity to register point clouds. Then, a lidar odometry method matches the point clouds at a low frequency to refine motion estimates and incrementally build maps. The lidar odometry also removes distortion in the point clouds caused by drift of the visual odometry. Combination of the two sensors allows the method to accurately map even with rapid motion and in undesirable lighting conditions.

The basic algorithm of V-LOAM is general enough that it can be adapted to use range sensors of different kinds, e.g. a time-of-fly camera. The method can also be configured to provide localization only, if a prior map is available.

In addition to evaluation on the KITTI odometry benchmark [1], we further experiment with a wide-angle camera and a fisheye camera. Our conclusion is that the fisheye camera brings in more robustness but less accuracy because of its larger field of view and higher image distortion. However, after the scan matching refinement, the final motion estimation reaches the same level of accuracy. Our experiment results can be seen in a publicly available video.[1].

II. Related Work

Vision and lidar based methods are common for state estimation [2]. With stereo cameras [3], [4], the baseline provides a reference to help determine scale of the motion. However, if a monocular camera is used [5]–[7], scale of the motion is generally unsolvable without aiding from other sensors or assumptions about motion. The introduction of RGB-D cameras provides an efficient way to associate visual images with depth. Motion estimation with RGB-D cameras

J. Zhang and S. Singh are with the Robotics Institute at Carnegie Mellon University. Emails: zhangji@cmu.edu and ssingh@cmu.edu.

[1] www.youtube.com/watch?v=-6cwhPMAap8

[8], [9] can be conducted easily with scale. A number of RGB-D visual odometry methods are also proposed showing promising results [10]–[12]. However, these methods only utilize imaged areas where depth is available, possibly wasting large areas in visual images without depth coverage. The visual odometry method used in our system is similar to [8]–[12] in the sense that all use visual images with additionally provided depth. However, our method is designed to utilize sparse depth information from a lidar. It involves features both with and without depth in solving for motion.

For 3D mapping, a typical sensor is a (2-axis) 3D lidar [13]. However, usage of these lidars is difficult as motion distortion is present in point clouds as the lidar continually ranges and moves. One way to remove the distortion is incorporating other sensors to recover the motion. For example, Scherer et al.'s navigation system [14] uses stereo visual odometry integrated with an IMU to estimate the motion of a micro-aerial vehicle. Lidar clouds are registered by the estimated motion. Droeschel et al.'s method [15] employs multi-camera visual odometry followed by a scan matching method based on a multi-resolution point cloud representation. In comparison to [14], [15], our method differs in that it tightly couples a camera and a lidar such that only one camera is needed for motion recovery. Our method also takes into account point cloud distortion caused by drift of the visual odometry, i.e. we model the drift as linear motion within a short time (1s) and correct the distortion with a linear motion model during scan matching.

It has also shown that state estimation can be made with 3D lidars only. For example, Tong et al. match visual features in intensity images created by stacking laser scans from a 2-axis lidar to solve for the motion [16]. The motion is modeled with constant velocity and Gaussian processes. However, since this method extracts visual features from laser images, dense point clouds are required. Another method is from Bosse and Zlot [17], [18]. The method matches geometric structures of local point clusters. They use a hand-held mapping device composed of a 2D lidar and an IMU attached to a hand-bar through a spring [17]. They also use multiple 2-axis lidars to map an underground mine [18]. In this method, the trajectory is recovered by batch optimization processing segmented data with boundary constraints connecting in between the segments. The method is appropriate for offline survey but unsuitable for online real-time applications.

The proposed method is based on our work in [19], [20], where a visual odometry method, DEMO, and a lidar odometry method, LOAM, are proposed separately. LOAM requires smooth motion and relies on an IMU to compensate for high frequency motion. In this paper, LOAM is modified such that the new method, V-LOAM, takes the visual odometry output as motion prior followed by the lidar odometry. The camera model in the visual odometry is also modified and compatible with fisheye cameras. A new set of experiments are conducted and results show V-LOAM delivers lower drift. Incorporating high-frequency visual odometry and a fisheye camera also enables the system to handle rapid motion.

III. Coordinate Systems and Task

The problem addressed in this paper is to estimate the motion of a camera and lidar system and build a map of the traversed environment with the estimated motion. We assume that the camera is modeled by a general central camera model [21]. With such a camera model, our system is able to use both regular and fisheye cameras (see experiment section). We assume that the camera intrinsic parameters are known. The extrinsic parameters between the camera and lidar are also calibrated. This allows us to use a single coordinate system for both sensors, namely the sensor coordinate system. For simplicity of calculation, we choose the sensor coordinate system to coincide with the camera coordinate system – all laser points are projected into the camera coordinate system upon receiving. As a convention of this paper, we use left uppercase superscription to indicate coordinate systems. In the following, let us define

- Sensor coordinate system $\{S\}$ is originated 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.
- World coordinate system $\{W\}$ is the coordinate system coinciding with $\{S\}$ at the starting position.

With assumptions and coordinate systems defined, our odometry and mapping problem is stated as

Problem: Given visual images and lidar clouds perceived in $\{S\}$, determine poses of $\{S\}$ with respect to $\{W\}$ and build a map of the traversed environment in $\{W\}$.

IV. System Overview

Fig. 2 shows a diagram of the software system. The overall system is divided into two sections. The visual odometry section estimates frame to frame motion of the sensor at the image frame rate, using visual images with assistance from lidar clouds. In this section, the feature tracking block extracts and matches visual features between consecutive images. The depth map registration block registers lidar clouds on a local depthmap, and associates depth to the visual features. The frame to frame motion estimation block takes the visual features to compute motion estimates.

To summarize the lidar odometry section, let us define a sweep as the 3D lidar completes one time of full scan coverage. If the slow axis of the lidar spins continuously, a sweep is typically a full-spherical rotation. However, if the slow axis rotates back-and-forth, a sweep is a clockwise or counter-clockwise rotation toward the same orientation. In our system, a sweep lasts for 1s. The lidar odometry section is executed once per sweep, processing point clouds perceived within entire sweeps. First, the sweep to sweep refinement block matches point clouds between consecutive sweeps to refine motion estimates and remove distortion in the point clouds. Then, the sweep to map registration block matches and registers point clouds on the currently built map, and publishes sensor poses with respect to the map. The sensor pose outputs are integration of the transforms from both sections, at the high frequency image frame rate.

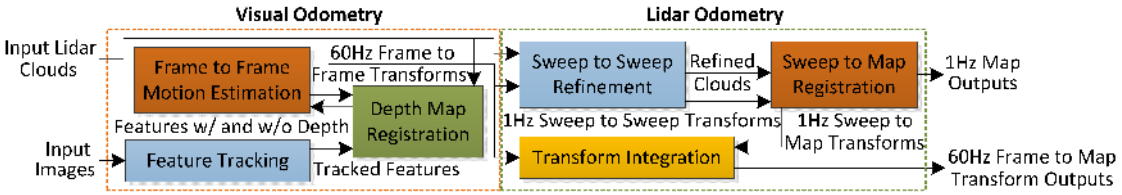


Fig. 2. Block diagram of the odometry and mapping software system.

V. Visual Odometry

This section summarizes the visual odometry method. With lidar clouds, the method registers and maintains a depthmap using estimated motion of the visual odometry. When computing motion, it involves three types of visual features in terms of the source of depth: depth from the depthmap, depth by triangulation using the previously estimated motion, and depth unavailable. However, to use fisheye cameras with over 180° field of view, let us use the term "distance" from now on (a feature's depth is the projection of its distance in ${}^S z$ direction). Let us use right superscript k, $k \in Z^+$ to indicate image frames, and $\mathcal{I}$ to indicate the set of visual features. For a feature i, $i \in \mathcal{I}$, that is associated with distance, its coordinates in $\{S^k\}$ are denoted as ${}^S\boldsymbol{X}_i^k = [{}^S x_i^k,\ {}^S y_i^k,\ {}^S z_i^k]^T$. For a feature with unknown distance, we use its normalized coordinates instead, ${}^S\bar{\boldsymbol{X}}_i^k = [{}^S\bar{x}_i^k,\ {}^S\bar{y}_i^k,\ {}^S\bar{z}_i^k]^T$, where $||{}^S\bar{\boldsymbol{X}}_i^k|| = 1$. We model the sensor motion as rigid body transformation. Let $\mathbf{R}$ and $\boldsymbol{T}$ be a 3×3 rotation matrix and a 3×1 translation vector describing the frame to frame motion. We formulate the motion as

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

In the case that a feature's distance is available, we can associate the distance to ${}^S\boldsymbol{X}_i^{k-1}$. However, the distance of ${}^S\boldsymbol{X}_i^k$ is always unknown. Since the motion between frames $k-1$ and k is not computed at this stage, we are not able to retrieve the distance of ${}^S\boldsymbol{X}_i^k$ either from the depthmap or by triangulation. Let ${}^S d_i^k$ be the unknown distance of ${}^S\boldsymbol{X}_i^k$, ${}^S d_i^k = ||{}^S\bar{\boldsymbol{X}}_i^k||$. Substituting ${}^S\boldsymbol{X}_i^k$ with ${}^S d_i^k {}^S\bar{\boldsymbol{X}}_i^k$ in (1) and combining the 1st and the 2nd rows with the 3rd row, we can eliminate ${}^S d_i^k$. This gives us two equations as follows,

$$ ({}^S\bar{z}_i^k\boldsymbol{R}_1 - {}^S\bar{x}_i^k\boldsymbol{R}_3){}^S\boldsymbol{X}_i^{k-1} + {}^S\bar{z}_i^k T_1 - {}^S\bar{x}_i^k T_3 = 0, \tag{2} $$

$$ ({}^S\bar{z}_i^k\boldsymbol{R}_2 - {}^S\bar{y}_i^k\boldsymbol{R}_3){}^S\boldsymbol{X}_i^{k-1} + {}^S\bar{z}_i^k T_2 - {}^S\bar{y}_i^k T_3 = 0. \tag{3} $$

Here, $\boldsymbol{R}_l$ and T_l, $l \in \{1,2,3\}$, are the l-th rows of $\mathbf{R}$ and $\boldsymbol{T}$.

For a feature without distance, both distances of ${}^S\boldsymbol{X}_i^{k-1}$ and ${}^S\boldsymbol{X}_i^k$ are unknown. Substituting the terms in (1) with ${}^S d_i^{k-1} {}^S\bar{\boldsymbol{X}}_i^{k-1}$ and ${}^S d_i^k {}^S\bar{\boldsymbol{X}}_i^k$, respectively, and combining all three rows to eliminate ${}^S d_i^{k-1}$ and ${}^S d_i^k$, we can obtain,

$$ \begin{bmatrix} -{}^S\bar{y}_i^k T_3 + {}^S\bar{z}_i^k T_2 \\ {}^S\bar{x}_i^k T_3 - {}^S\bar{z}_i^k T_1 \\ -{}^S\bar{x}_i^k T_2 + {}^S\bar{y}_i^k T_1 \end{bmatrix} \mathbf{R}\ {}^S\bar{\boldsymbol{X}}_i^{k-1} = 0. \tag{4} $$

The above procedure tells that a feature with known distance provides two equations as (2)-(3), while a feature with unknown distance provides one equation as (4). When solving for motion, we stack all equations and formulate the motion estimation problem with six unknowns representing the 6-DOF motion. The problem is solved by the Levenberg-Marquardt method. The motion estimation is adapted to a robust fitting framework to handle feature tracking errors. A weight is assigned to each feature based on its residuals in (2)-(3) or (4). Features with larger residuals are assigned with smaller weights, while features with residuals larger than a threshold are considered outliers and assigned with zero weights. The optimization terminates if convergence is found or the maximum iteration number is met.

When maintaining the depthmap, new points are added to the depthmap upon receiving from lidar clouds. Only points in front of the camera are kept, and points that are received a certain time ago are forgotten. The depthmap is downsized to keep a constant point density, and projected to the last image frame whose transform to the previous frame is established, namely frame $k-1$. We represent points on the depthmap in spherical coordinates using a distance and two angles. The points are stored in a 2D KD-tree based on the two angular coordinates. When associating distances to the features, we find the three closest points on the depthmap from each feature. The three points form a local planar patch, and the distance is interpolated from the three points by projecting a ray from the camera center to the planar patch.

Further, if the distances are unavailable from the depthmap for some features but they are tracked more than a certain distance, we triangulate them using the sequences of images where the features are tracked. Fig. 3 shows an example of reconstructed features corresponding to Fig. 1 (left image). The green dots are features whose distances are associated from the depthmap, and the blue dots are by triangulation (the red dots in Fig. 1 have unknown distances).

VI. Lidar Odometry

The frame to frame motion estimated by the visual odometry is further refined by the lidar odometry method. The lidar odometry contains two major steps for coarse to fine processing of point clouds: a sweep to sweep refinement step matches point clouds between consecutive sweeps to

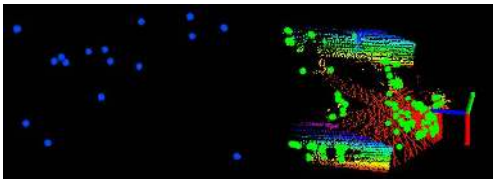


Fig. 3. An example of depthmap and reconstructed visual features corresponding to the left image in Fig. 1. The colored points represent the depthmap, where color codes elevation. The green dots are features whose distances are from the depthmap, and the blue dots are obtained by structure from motion (the red dots in Fig. 1 have unknown distances).

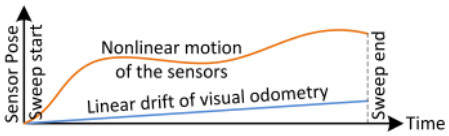


Fig. 4. Illustration of visual odometry drift. The orange curve represents nonlinear motion estimated by the visual odometry, and the blue line represents the visual odometry drift. We model the drift as linear motion within a sweep (lasting for 1s). The drift creates distortion in lidar clouds. The sweep to sweep refinement step corrects the distortion by matching lidar clouds between consecutive sweeps with a linear motion model.

refine motion estimates, and a sweep to map registration step matches and registers point clouds on the map.

Fig. 4 illustrates functionality of the sweep to sweep refinement step. The orange curve represents nonlinear motion of the sensor estimated by the visual odometry. The drift of the visual odometry is usually considered as slow motion. We model the drift with contact velocity within a sweep (lasting for 1s), represented by the blue line. When using motion recovered by the visual odometry to register lidar clouds, the drift causes distortion in the lidar clouds. The sweep to sweep refinement step incorporates a linear motion model in lidar cloud matching to remove the distortion.

Let us use right superscript m, $m \in Z^+$ to indicate sweeps, and $\mathcal{P}^m$ to indicate the lidar cloud perceived during sweep m. For each $\mathcal{P}^m$, we extract geometric features combing points on sharp edges, namely edge points, and points on planar surfaces, namely planar points, by computation of the curvature in local scans. We avoid selecting points whose neighbor points are selected, and points on boundaries of occluded regions or local surfaces that are roughly parallel to the laser beams. These points are likely to contain large noises or change positions over time. Fig. 5 gives an example of edge points and planar points detected from a sweep when the sensor navigates in front of a building.

Let $\mathcal{E}^m$ and $\mathcal{H}^m$ be the sets of edge points and planar points extracted from $\mathcal{P}^m$. We match $\mathcal{E}^m$ and $\mathcal{H}^m$ to the lidar cloud from the previous sweep, $\mathcal{P}^{m-1}$. Here, note that after completion of sweep $m-1$, the distortion in $\mathcal{P}^{m-1}$ is corrected. Hence, we only need to apply the linear motion model for the current sweep. Define $\boldsymbol{T}'$ as a 6×1 vector describing the visual odometry drift during sweep m, and define t^m as the starting time of this sweep. For a point i, $i \in \mathcal{E}^m \cup \mathcal{H}^m$, perceived at time t_i, the corresponding drift between t^m and t_i is linearly interpolated as,

$$\boldsymbol{T}'_i = \boldsymbol{T}'(t_i - t^m)/(t^{m+1} - t^m). \tag{5}$$

For each point in $\mathcal{E}^m$, we find the two closest edge points

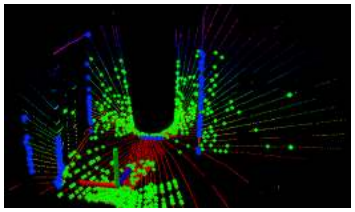


Fig. 5. An example of detected edge points (blue) and planar points (green) from a sweep. The sensor points to a building during data collection.

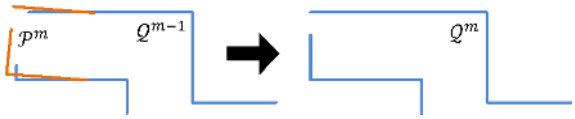


Fig. 6. Illustration of sweep to map registration step. For each sweep m, the lidar cloud $\mathcal{P}^m$ is matched with the existing map cloud $\mathcal{Q}^{m-1}$. Then, the two point clouds are merged to form a new map cloud $\mathcal{Q}^m$.

in $\mathcal{P}^{m-1}$ which form an edge line segment. For each point in $\mathcal{H}^m$, we find the three closest planar points which form a local planar patch. This process employs two 3D KD-trees, one storing edge points and the other storing planar points in $\mathcal{P}^{m-1}$. With correspondences of the edge points and the planar points found, an equation is derived to describe the distance between a point and its correspondence,

$$f({}^S\boldsymbol{X}_i^m, \boldsymbol{T}'_i) = d_i, \tag{6}$$

where ${}^S\boldsymbol{X}_i^m$ is the coordinates of point i, $i \in \mathcal{E}^m \cup \mathcal{H}^m$, in $\{S^m\}$, and d_i is the distance to its correspondence. Combining (5) and (6), we obtain a function of $\boldsymbol{T}'$. The process of solving for $\boldsymbol{T}'$ is stacking the function of each edge point and planar point and then minimizing the overall distances. The nonlinear optimization uses the Levenberg-Marquardt method adapted to a robust fitting framework. With $\boldsymbol{T}'$ computed, we remove the distortion in $\mathcal{P}^m$.

Finally, the sweep to map registration step matches and registers the distortion-free lidar clouds on the currently built map. Define $\mathcal{Q}^m$ as the map cloud at the end of sweep m. As illustrated in Fig. 6, this step matches $\mathcal{P}^m$ with $\mathcal{Q}^{m-1}$ and merges the two point clouds to build a new map cloud $\mathcal{Q}^m$. The same types of edge points and planar points are extracted from $\mathcal{P}^m$. Considering the density nature of the map cloud, correspondences of the feature points are determined by examining distributions of local point clusters in $\mathcal{Q}^{m-1}$, through computation of eigenvalues and eigenvectors. Specifically, one large and two small eigenvalues indicate an edge line segment, and two large and one small eigenvalues indicate a local planar patch. The scan matching involves an iterative closest point method [22], similar to the sweep to sweep refinement step without the motion model.

After registration of $\mathcal{P}^m$ on the map, a transform is also published regarding sensor poses on the map, in the world coordinate system $\{W\}$. Since these transforms are only computed once per sweep, we combine them with the high frequency frame to frame motion transforms from the visual odometry. As illustrated in Fig. 7, the result is high frequency integrated pose outputs at the image frame rate.

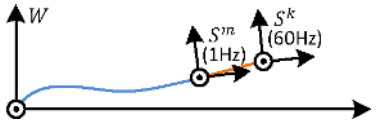


Fig. 7. Illustration of transform integration. The blue segment represents transforms published by the lidar odometry at a low frequency, regarding sensor poses in the world coordinate system $\{W\}$. The orange segment represents transforms published by the visual odometry at a high frequency containing frame to frame motion. The two transforms are integrated to generate high frequency sensor pose outputs at the image frame rate.

VII. Experiments

The study of this paper is validated on two sensor systems, one using a custom-built camera and lidar sensor as shown in Fig. 8, and the other using configuration of the KITTI benchmark datasets [1]. Through the paper, we have used data collected from the custom-built sensor to illustrate the method. The camera is a uEye monochrome camera configured at 60Hz frame rate. The 3D lidar is based on a Hokuyo UTM-30LX laser scanner. The laser scanner has $180°$ field of view and $0.25°$ resolution with 40 lines/sec scanning rate. A motor actuates the laser scanner for rotational motion to realize 3D scan. The motor is controlled to rotate at $180°/s$ angular speed between $-90°$ and $90°$ with the horizontal orientation of the laser scanner as zero. An encoder measures the motor rotation angle with $0.25°$ resolution.

The software program processing author-collected data runs on a laptop computer with 2.5GHz quad cores in Linux. The method consumes about two and a half cores: the visual odometry takes two cores, and the lidar odometry takes half of a core as it is only executed once per sweep. The method tracks maximally 300 Harris corners using the Kanade Lucas Tomasi (KLT) method [23]. To evenly distribute the visual features, an image is separated into 5×6 identical subregions, while each subregion provides up to 10 features.

When evaluating on the KITTI odometry benchmark [1], the method uses data from a single camera and a Velodyne lidar. It outperforms other methods irrespective of sensing modality, including the lidar only method, LOAM [20]. This is mostly because V-LOAM uses images to compute motion prior for scan matching, while LOAM only processes laser data. Our results for both methods are publicly available[2].

A. Accuracy Tests

We first conduct accuracy tests using two camera setups, one with a wide-angle lens ($76°$ horizontal field of view) and the other with a fisheye lens ($185°$ horizontal field of view). To acquire both images at the same time, another camera is mounted underneath the original camera in Fig. 8, and set at the same configuration except the resolution is slightly different. The original camera is at 752×480 pixels while the second camera is at 640×480 pixels. This is because the fisheye lens provides pixel information in a circular region (see examples in Fig. 9(a)) and further extending the camera horizontal resolution only enlarges the black region.

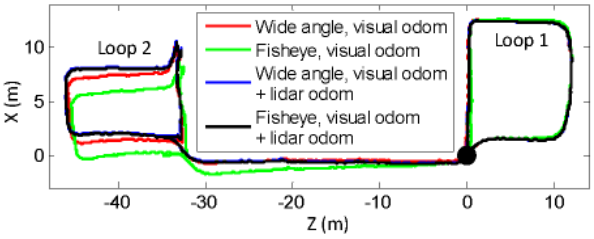
[2] www.cvlibs.net/datasets/kitti/eval_odometry.php

Fig. 8. Custom-built camera and lidar sensor. The camera is a uEye monochrome camera configured at 60Hz frame rate. The 3D lidar consists of a Hokuyo UTM-30LX laser scanner driven by a motor and an encoder that measures the rotation angle. The motor rotates back-and-forth at $180°/s$.

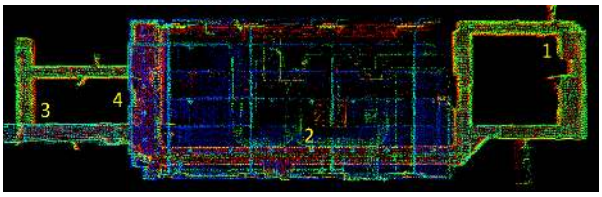
Fig. 9 and Fig. 10 show results of accuracy tests in an indoor and an outdoor environments. In both tests, the sensor is held by a person who walks at 0.7m/s. Fig. 9-10(a) present sample images from the tests. In Fig. 9(a), the first row is from the wide-angle camera, and the second row is corresponding images from the fisheye camera. In Fig. 10(a), we only show images from the wide-angle camera due to limited space. Fig. 9-10(b) show results of motion estimation. We compare four trajectories: two from the visual odometry with the wide-angle camera and the fisheye camera, respectively, and the other two refined by the lidar odometry. We see the fisheye camera results in faster drift (green curves) than the wide-angle camera (red curves) as a result of heavier image distortion. However, the trajectories refined by the lidar odometry (blue and black curves) have little difference, indicating that the lidar odometry is able to correct the visual odometry drift regardless of the drift amount. Fig. 9-10(c) show maps built corresponding to the blue curves in Fig. 9-10(b). The images in Fig. 9-10(a) labeled with numbers 1-4 are respectively taken at locations 1-4 in Fig. 9-10(c).

Additionally, we conduct one test including indoor and outdoor environments. As shown in Fig. 11, the path starts in front of a building, passes through the building and exits to the outside, traverses two staircases and follows a small

(a)



(b)



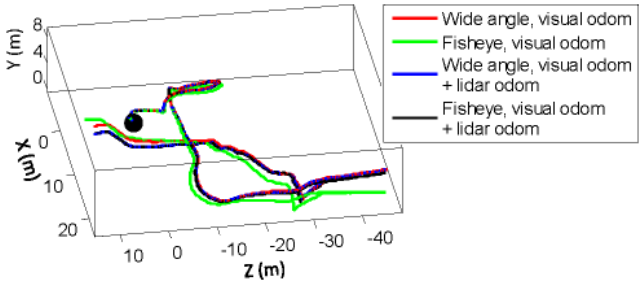
(c)

Fig. 9. Result of Test 1: indoor accuracy. (a) Sample images from the test. The top row is from the wide-angle camera and the bottom row is from the fisheye camera. (b) The red and green trajectories are outputs from the visual odometry (1st section in Fig. 2) with different camera setups, and the blue and black trajectories are refined motion estimates by the lidar odometry (2st section in Fig. 2). The black dot is the starting position. Although using the fisheye camera leads to larger drift (green) than the wide-angle camera (red), the refined trajectories by the lidar odometry (blue and black) have little difference. (c) Map built corresponding to the blue curve in (b). The images in (a) labeled with 1-4 are taken at locations 1-4 in (c).

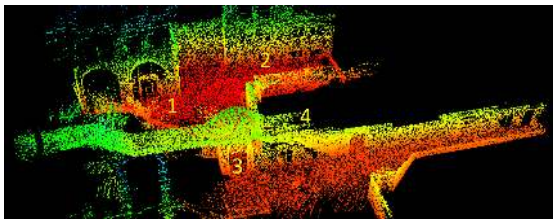
road to come back to the starting position after 538m of travel. Due to space issue, we eliminate the trajectories and only show the map built. The images in Fig. 11(a) are taken at corresponding locations 1-6 in Fig. 11(b).

Table I compare motion estimation accuracy for the three tests. The accuracy is calculated based on 3D coordinates. For Test 1, the path contains two loops. We measure gaps on the trajectories at loop closures to determine relative position errors as fractions of the distance traveled along the loops. For Test 2, the lidar perceives the same objects at the start and the end of the path. We manually extract and correlate 15 points in lidar clouds to calculate the position error. For Test 3, the position error is measured between the starting and the ending positions. From Table I, we conclude that even though the visual odometry is less accurate with the fisheye camera than the wide-angle camera, the lidar odometry is able to boost the accuracy to the same level.

(a)



(b)



(c)

Fig. 10. Result of Test 2: outdoor accuracy. (a)-(c) are in the same arrangement as Fig. 9. Similar to Test 1 (Fig. 9), using the fisheye camera (green curve) results in larger visual odometry drift than the wide-angle camera (red curve). However, the lidar odometry is able to correct the drift and generate trajectories with little difference (blue and black curves). The images in (a) labeled with 1-4 are taken at locations 1-4 in (c).

TABLE I

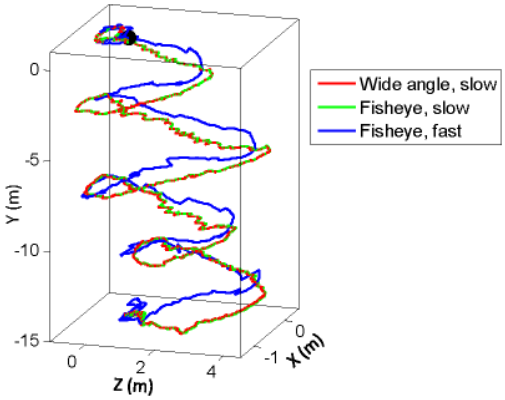
RELATIVE POSITION ERRORS IN ACCURACY TESTS

W: WIDE-ANGLE, F: FISHEYE, V: VISUAL ODOM (1ST SECTION IN FIG. 2), VL: VISUAL ODOM + LIDAR ODOM (BOTH SECTIONS IN FIG. 2).

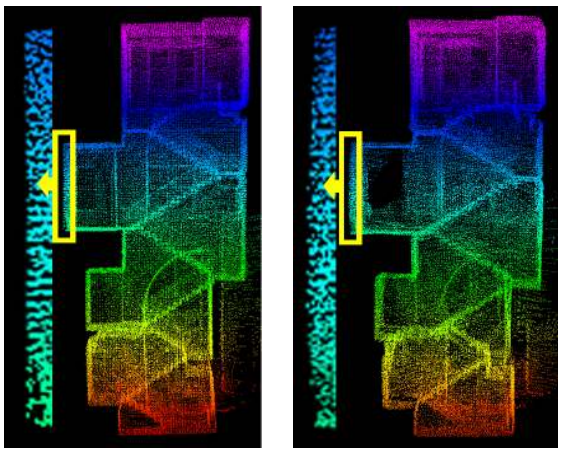
Test No.	Dist.	Relative Position Error			
		W-V	F-V	W-VL	F-VL
Test 1 (Loop 1)	49m	**1.1%**	1.8%	**0.31%**	**0.31%**
Test 1 (Loop 2)	47m	**1.0%**	2.1%	**0.37%**	**0.37%**
Test 2	186m	**1.3%**	2.7%	**0.63%**	0.64%
Test 3	538m	**1.4%**	3.1%	**0.71%**	0.73%

B. Robustness Tests

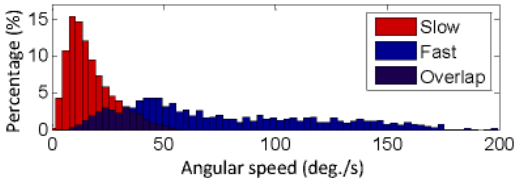
We further conduct experiments to inspect robustness of the method with respect to fast motion. We first choose a staircase environment as in Fig. 12, which includes seven



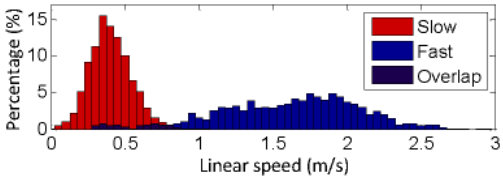
(a)



(b) (c)

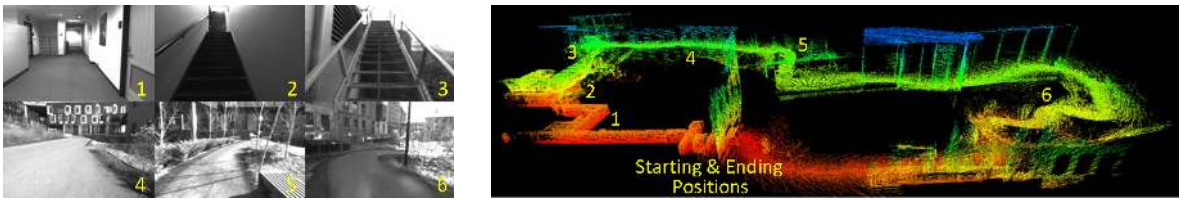


(d)



(e)

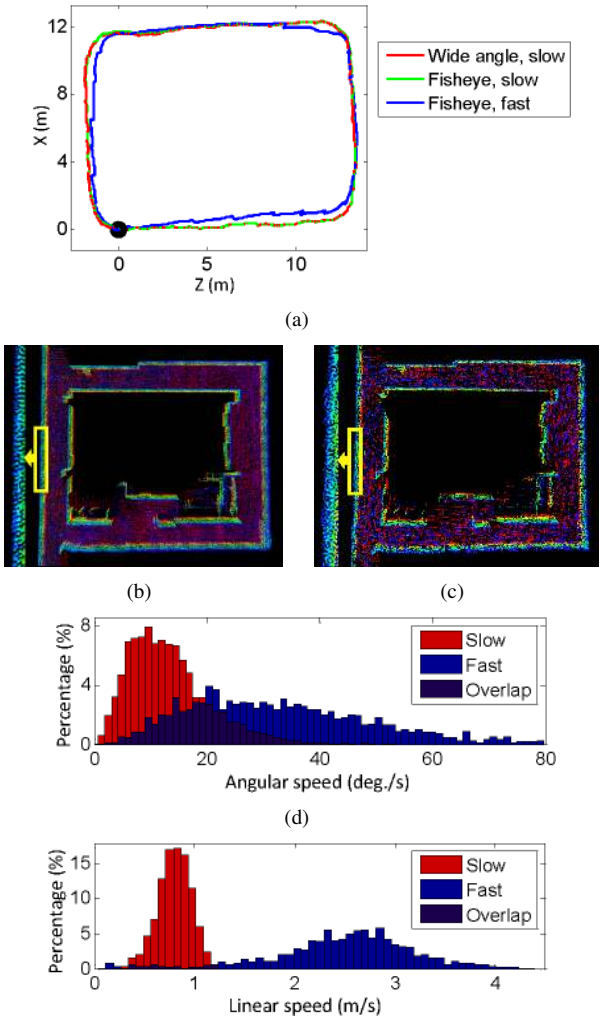
Fig. 12. Result of Test 4: robustness w.r.t. fast rotation. The test contains two trials, one in slow motion and the other in fast motion, following the same path. (a) shows estimated trajectories. The red and the green curves are from the same trial in slow motion, while the blue curve is from the other trial in fast motion. When using the wide-angle camera in fast motion, the motion estimation fails due to visual features loose tracking during fast turnings. The trajectory is removed. (b)-(c) are maps built corresponding to the green and the blue curves, respectively. Careful comparison finds that the point cloud in (c) is blurred as an effect of fast motion. (d)-(e) present distributions of angular speed and linear speed for the two trials.



(a) (b)

Fig. 11. Result of Test 3: indoor and outdoor accuracy. The path starts in front of a building, passes through the building and exits to the outside, traverses two staircases and follows a small road to come back to the starting position. The images in (a) are taken at corresponding locations 1-6 in (b).

180° turns. Walking on the stairs introduces continuous rotation to the sensor. Second, we choose a corridor environment as in Fig. 13. Traveling along the corridor brings in continuous translation. In each environment, a person holds the sensor and follows the same path twice, one in slow motion and the other in fast motion. In Fig. 12-13(a), we show estimated trajectories. The red and the green curves are from the slow motion trials, using the wide-angle camera and the fisheye camera, respectively. The blue curves are from the fast motion trials. In both tests, when using the wide-angle camera in fast motion, we encounter issues that visual features loose tracking during fast turnings, resulting in failure of motion estimation. The trajectories are removed. In Fig. 12-13(b), we present the maps corresponding to the green curves, and in Fig. 12-13(c), we show the maps with respect to the blue curves. When comparing carefully, one finds that the point cloud in Fig. 12(c) is blurred and the walls in Fig. 13(c) are bended due to fast motion.



(a) (b) (c) (d) (e)

Fig. 13. Result of Test 5: robustness w.r.t. fast translation. Same as Test 4 (Fig. 12), it contains two trials, one in slow motion and the other in fast motion. The maps in (b) and (c) correspond to the green and the blue curves in (a). One finds the walls in (c) are bended as a result of fast motion.

The distributions of angular speed and linear speed are shown in Fig. 12-13(d) and Fig. 12-13(e), respectively. The angular speed is calculated using spatial rotation, and the linear speed is based on 3D translation. We can see significant difference in speed between the slow and the fast trials. In Fig. 12(d), the angular speed covers up to $170^\circ/s$ for the fast trial, and in Fig. 13(e), the average linear speed is around 2.6m/s. Table II compares relative position errors. For Test 4, the ground truth is manually calculated assuming the walls on different floors are exactly flat and aligned. We are able to measure how much the walls are bended and therefore determine position error at the end of the trajectory. For Test 5, the error is calculated using the gap at loop closure. From these results, we draw the conclusion that using the fisheye camera looses slight accuracy compared to the wide-angle camera, but gains more robustness in rapid motion.

Finally, we experiment on robustness of the method with respect to dramatic lighting changes. As shown in Fig. 14(a), the light is turned off four times. At locations 1-2, the sensor navigates inside a room, and at locations 3-4, the sensor moves along a corridor. When the light goes off, the visual odometry stops working and constant velocity prediction is used instead. The drift is corrected by the lidar odometry once per sweep. Fig. 14(b) presents the map built. Fig. 14(c)-(d) show the amount of corrections applied by the lidar odometry. The four peaks represent large corrections corresponding to the red segments in Fig. 14(a), caused by the fact that constant velocity prediction drifts faster than the visual odometry. The result indicates that the method is able

TABLE II
RELATIVE POSITION ERRORS IN FAST MOTION TESTS
W: WIDE-ANGLE, FI: FISHEYE, S: SLOW, FA: FAST.

Test No.	Dist.	Relative Position Error			
		W-S	Fi-S	W-Fa	Fi-Fa
Test 4	66m	**0.67%**	0.68%	Failed	**1.3%**
Test 5	54m	**0.27%**	0.28%	Failed	**0.39%**

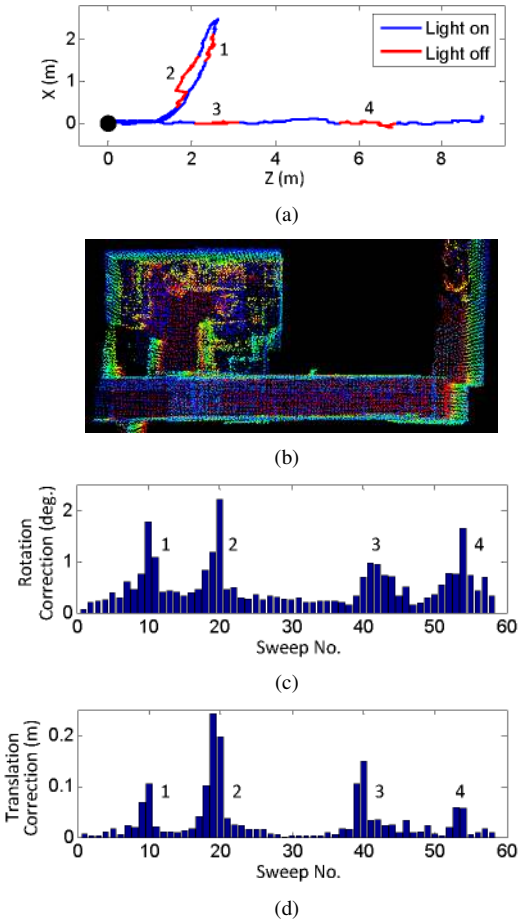


Fig. 14. Result of Test 6: lighting changes. During the test, light is turned off four times as indicated by the red segments in (a), each lasting for 2s. In undesirable lighting conditions, the visual odometry stops working and constant velocity prediction is used instead. The drift is corrected by the lidar odometry once per sweep. (b) presents the map built in top-down view. (c)-(d) show the amount of rotation and translation corrections applied by the lidar odometry. The corrections become much larger when light is off due to the fact that constant velocity prediction drifts faster than the visual odometry. The four peaks in (c)-(d) correspond to locations 1-4 in (a).

to handle temporary light outrage (however, for continuous darkness, the proposed method is unsuitable and readers are recommended to use our lidar only method, LOAM [20]).

VIII. Conclusion

We propose a real-time method for odometry and mapping using a camera combined with a 3D lidar. This is through a visual odometry method that estimates ego-motion at a high frequency and a lidar odometry method that refines motion estimates and corrects drift at a low frequency. Cooperation of the two components allows accurate and robust motion estimation to be realized, i.e. the visual odometry handles rapid motion, and the lidar odometry warrants low-drift. The method is tested indoors and outdoors using datasets collected in our own experiments, with a wide-angle camera and a fisheye camera. The method is further evaluated on the KITTI odometry benchmark with an average of %0.75 relative position drift. Our experiment results also show robustness of the method when the sensor moves at a high speed and is subject to significant lighting changes.

References

[1] 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.

[2] S. Shen and N. Michael, "State estimation for indoor and outdoor operation with a micro-aerial vehicle," in *International Symposium on Experimental Robotics (ISER)*, June 2012.

[3] 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.

[4] 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.

[5] G. Klein and D. Murray, "Parallel tracking amd mapping for small AR workspaces," in *International Symposium on Mixed and Augmented Reality*, Nov. 2007.

[6] 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 (ICCV)*, Nov 2011.

[7] 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.

[8] 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*, April 2011.

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

[10] 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 *International Symposium on Robotics Research (ISRR)*, Aug. 2011.

[11] 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.

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

[13] A. Nuchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6D SLAM–3D mapping outdoor environments," *Journal of Field Robotics*, vol. 24, no. 8-9, pp. 699–722, 2007.

[14] S. Scherer, J. Rehder, S. Achar, H. Cover, A. Chambers, S. Nuske, and S. Singh, "River mapping from a flying robot: state estimation, river detection, and obstacle mapping," *Autonomous Robots*, vol. 32, no. 5, pp. 1–26, May 2012.

[15] D. Droeschel, J. Stuckler, and S. Behnke, "Local multi-resolution representation for 6D motion estimation and mapping with a continuously rotating 3D laser scanner," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2014.

[16] C. H. Tong, S. Anderson, H. Dong, and T. Barfoot, "Pose interpolation for laser-based visual odometry," *Journal of Field Robotics*, vol. 31, no. 5, pp. 731–757, 2014.

[17] M. Bosse, R. Zlot, and P. Flick, "Zebedee: Design of a spring-mounted 3-D range sensor with application to mobile mapping," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1104–1119, 2012.

[18] R. Zlot and M. Bosse, "Efficient large-scale three-dimensional mobile mapping for underground mines," *Journal of Field Robotics*, vol. 31, no. 5, pp. 758–779, 2014.

[19] J. Zhang, M. Kaess, and S. Singh, "Real-time depth enhanced monocular odometry," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2014.

[20] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Robotics: Science and Systems Conference (RSS)*, July 2014.

[21] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A flexible technique for accurate omnidirectional camera calibration and structure from motion," in *IEEE International Conference on Computer Vision Systems*, New York City, NY, Jan. 2006.

[22] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP variants on real-world data sets," *Autonomous Robots*, vol. 34, no. 3, pp. 133–148, 2013.

[23] 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.