2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Madrid, Spain, October 1-5, 2018

Stereo Camera Localization in 3D LiDAR Maps

Youngji Kim[1], Jinyong Jeong[1] and Ayoung Kim[1*]

***Abstract*— As simultaneous localization and mapping (SLAM) techniques have flourished with the advent of 3D Light Detection and Ranging (LiDAR) sensors, accurate 3D maps are readily available. Many researchers turn their attention to localization in a previously acquired 3D map. In this paper, we propose a novel and lightweight camera-only visual positioning algorithm that involves localization within prior 3D LiDAR maps. We aim to achieve the consumer level global positioning system (GPS) accuracy using vision within the urban environment, where GPS signal is unreliable. Via exploiting a stereo camera, depth from the stereo disparity map is matched with 3D LiDAR maps. A full six degree of freedom (DOF) camera pose is estimated via minimizing depth residual. Powered by visual tracking that provides a good initial guess for the localization, the proposed depth residual is successfully applied for camera pose estimation. Our method runs online, as the average localization error is comparable to ones resulting from state-of-the-art approaches. We validate the proposed method as a stand-alone localizer using KITTI dataset and as a module in the SLAM framework using our own dataset.**

I. Introduction

Localization is essential in robotic systems such as self-driving cars. All the other navigation tasks (e.g., planning and control) then follow based on the accurate estimation of the robot's pose. Traditionally, autonomous vehicles have relied on GPS/inertial navigation system (INS) for localization [1], [2], [3]. GPS provides a drift-free global position which can be combined with the high-frequency relative pose estimated from INS. However, GPS often suffers from intermittent errors caused by multi-path effects. Typical examples include urban canyons and indoor environments, where accurate pose estimation is limited.

Despite an accurate map available [4], [5], [6], localization against these maps requires registration between the map and onboard sensors. An obvious and favored method would be to use the same sensor for mapping and localization. This line of research was found in [7], [8], [9], in which the researchers used LiDARs for 3D mapping and localization. Since LiDARs provide accurate 3D range data, direct matching between the given map and a current scan was possible using scan matching techniques such as Iterated Closest Point (ICP) [10]. However, due to the associated cost and physical requirements, studies prefer to achieve the localization performance using vision [11], [12], [13], [14], [15].

The major dilemma of camera localization in LiDAR maps originates from different modalities in the camera and LiDAR data. Fig. 1 illustrates how the information provided by the two sensors differs. LiDAR data consist of ranges and reflectance values whereas cameras offer intensity images in color or gray-scale. From the residuals in Fig. 1, reflectance values from LiDARs are mostly dissimilar to the camera intensity values. Depths estimated from stereo cameras and LiDARs are more consistent but still have a discrepancy on edges. Moreover, data density obtained from the two sensors are different. Typical LiDAR images show stripe density patterns that are horizontally dense and vertically sparse while cameras provide uniform density images.

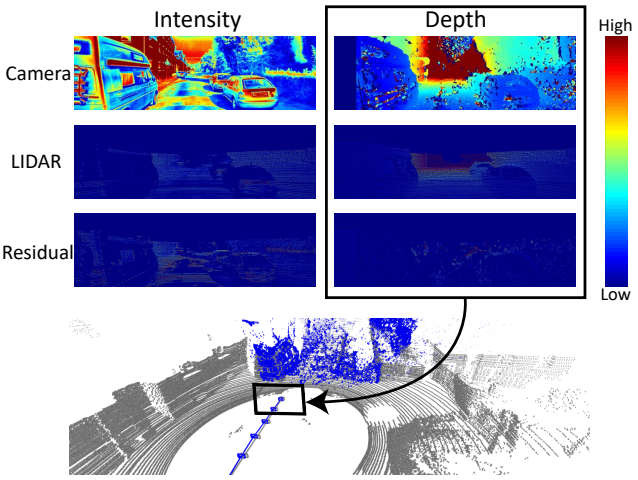


Fig. 1. Overview of the proposed algorithm. Instead of comparing camera and LiDAR intensities, we use stereo depth and compare it with depth from LiDAR map. The camera pose is estimated by minimizing depth residual. The figure below illustrates the estimated pose in the map where the gray points are map points and the blue points are points reconstructed from the stereo camera. The camera trajectory denoted as blue lines follows the ground truth camera trajectory marked as gray lines.

Recent approaches to camera localization in 3D maps have focused on overcoming the problems caused by the inherent differences in camera and LiDAR data. Matching photometry is one of the most common approaches. Wolcott and Eustice [12] used LiDAR reflectance images and matched them to a camera intensity image. Several candidate reflectance images that can be matched with a camera image were prepared beforehand. A camera pose corresponds to the synthesized image that has the maximum Normalized Mutual Information (NMI) selected as a current camera pose. Another approach proposed by Stewart and Newman [11] used Normalized Information Distance (NID) as a metric for matching the camera and LiDAR intensities. Instead of synthesizing candidate LiDAR images and naively performing a brute-force search, they solved the problem via

Y. Kim, J. Jeong and A. Kim are with the Department of Civil and Environmental Engineering, KAIST, Daejeon, S. Korea [youngjikim, jjy0923, ayoungk]@kaist.ac.kr

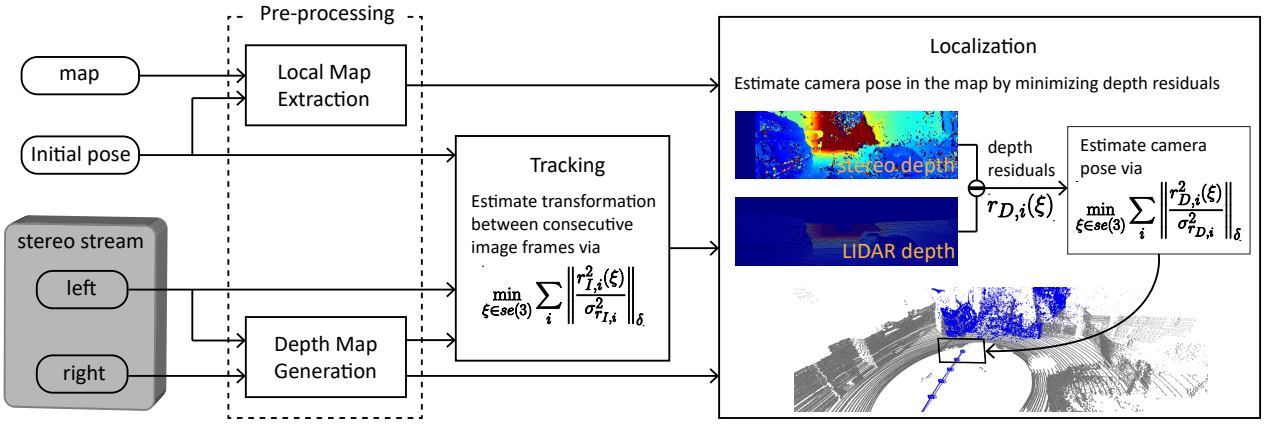


Fig. 2. A diagram of the proposed system. Inputs of the algorithm are a 3D LiDAR prior map, the initial camera pose, and the stereo image stream. The pre-processing includes local map extraction and depth map generation. In tracking, camera pose between consecutive left image frames is estimated and used as an initial guess for the localization. For localization, we find a camera pose whose depth residual is minimized.

Quasi-Newton optimization with analytical derivatives. This enables direct 6 DOF camera pose estimation.

The second strategy of camera localization in 3D maps is to exploit geometry. Once feature points observed from the camera are reconstructed via visual SLAM schemes such as bundle adjustments, the problem now becomes matching two sets of point clouds. Forster *et al.* [16] proposed a method of aligning 3D maps acquired from two different platforms equipped with a depth sensor and a monocular camera. Camera pose within the map was estimated through scan matching between the 3D map obtained by the depth sensor and the points reconstructed from bundle adjustment. Similarly, in [13], camera point clouds acquired from ORB-SLAM were used for matching with a priori map. To match point clouds with different densities, they proposed a method for filtering and associating corresponding points using voxel. A hybrid of photometric and geometric matching is one of the promising methods for camera localization within a given 3D map. Neubert *et al.* [14] extracted depth images from the prior map and matched them to camera intensity images. Conversely, Xu *et al.* [15] retrieved depth images from a stereo camera. They synthesized LiDAR reflectance images from the map and compared against the depth images.

We propose a novel hybrid matching scheme for stereo camera localization in a given prior map. Unlike the previous hybrid methods, which attempt to match geometric and photometric information, we primarily use geometric information since it is more consistent than photometric information as shown in Fig. 1. However, photometric matching using geometry is performed through alignment of images made with depth information. After obtaining depth images from the stereo camera and the given map, one could estimate camera pose by minimizing differences in the two depth images. Estimating pose by reducing depth residual is also reported in direct monocular SLAM methods such as LSD-SLAM [17], which uses depth residual for compensating drift while aligning keyframes. In lieu of using depth residual for aligning keyframes, we seek to match the current camera measurement and the given map. Minimizing depth residual leads to successful localization of the stereo camera.

The proposed localization approach presents substantial improvements over the previous methods. Rather than finding one DOF at a time or providing only a 3 DOF pose as in [12], [14], [15], our method searches a full 6 DOF camera pose simultaneously. Moreover, our algorithm requires less computational cost because it is unnecessary to synthesize images from the map. Indeed, synthesizing images demands extensive computation. In previous studies, GPU was leveraged for synthesizing images [12], or synthetic images were prepared in advance offline [14]. Another bottleneck in the algorithm is the metric calculation for matching. NMI used in [12] or NID in [11] takes an algorithm complexity of $O(n + m^2)$, where n is the number of observed 3D points and m is the number of intensity bins. Our metric is the sum of depth differences with complexity of $O(n)$.

Our method is similar to [13] in that a full 6 DOF camera pose is computed by matching geometry. However, as we use a stereo camera instead of a monocular camera, it is unnecessary to estimate the scale of the scene in addition to the 6 DOF camera pose. By reducing search space from 7D to 6D, we can obtain higher localization accuracy. Furthermore, instead of conducting ICP, our method accomplished photometric matching via minimizing depth residual. The major advantage of using depth residual is in obviating the correspondence searching phase that is repeatedly executed in ICP. Finding correspondence is cumbersome, especially when the densities of the two matched point clouds are discrepant. In [13], this problem was solved by point cloud filtering using voxel. Our method minimizes redundant processes so as to generally apply to 3D maps having various densities.

In summary, we propose a stereo camera localization

method in 3D LiDAR maps that has the following attributes:

- We solve the multi-modal matching problem by minimizing depth residual.
- Our algorithm does not focus on global-scale localization. We need to specify the initial camera pose in the map and estimate the relative camera pose between scenes through visual tracking.
- We estimate a full 6 DOF camera pose at the same time.
- The proposed method is lightweight and operates online on CPU.

II. Proposed Method

We propose a system capable of localizing a stereo camera with respect to a previously given 3D map. We assume that the initial camera pose is provided and perform localization given a rough initial guess. Fig. 2 shows a diagram of the proposed localizer. Our system consists of four modules. In pre-processing, the raw data acquired from the map and the stereo image stream are processed to be used for the following tracking and localization module. In depth map generation, a depth map is generated by using stereo disparity. In local map extraction, a local 3D map that will be matched with the depth map is extracted from the global map. To determine the initial guess for the camera pose, tracking is added before the localization. Relative pose between consecutive image frames is estimated in this module. Then, 6 DOF camera pose is estimated by using the local map, depth map and putative pose from the tracking process.

Throughout the proposed method, we use homogeneous coordinates of points in 2D and 3D given below.

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \text{ and } \mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{1}$$

Camera pose is represented by $SE(3)$ transformation as

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}, \text{ where } \mathbf{T} \in SE(3), \tag{2}$$

which is related with its corresponding $se(3)$ Lie algebra $\xi^\wedge$ by exponential map

$$\mathbf{T}(\xi) = \exp(\xi^\wedge). \tag{3}$$

The wedge operator $\wedge$ turns $\xi \in \mathbb{R}^6$ into a member of the $se(3)$ lie algebra. For tracking and localization, camera pose $\mathbf{T}$ is iteratively updated as

$$\mathbf{T} \leftarrow \mathbf{T}(\xi) \cdot \mathbf{T}, \tag{4}$$

where the increment $\mathbf{T}(\xi)$ is applied by using left multiplication convention.

A. Local Map Extraction

Within global maps containing a large scale point clouds, we need to extract a local region of the global map points that can be observed from the current camera position. We define this sub-portion of the global map points as a local map. Local map extraction is implemented using octree [18]. The octree is a tree-based data structure commonly used for containing 3D point clouds. Each node in the octree possesses eight children that represent eight sub-cubes. Fast space partitioning and searching are performed by using octree. The point cloud library (PCL) [19] provides octree and various neighbors search methods. To extract points surrounded by current camera pose, neighbors within the radius search are used for the local map extraction.

B. Depth Map Generation

The main strategy is to exploit a depth map from the current stereo stream. The constructed depth map is primarily used in the upcoming tracking and localization. In the first step of depth map generation, the disparity map is created by stereo semi-global blob matching (SGBM) provided by OpenCV [20]. SGBM [21] is a stereo matching technique that estimates disparity by minimizing an energy function composed of the mutual information based pixel-wise cost and global smoothness cost. A dense stereo disparity map can be obtained by this SGBM. Then, scene depth is estimated via [22], which removes range dependent statistical bias in triangulation.

C. Visual Tracking

Visual tracking provides an initial guess for the localization. Given below is a simple visual tracking algorithm based on Gauss-Newton minimization of the photometric error. When the localizer is used as a stand-alone module, we applied this simple tracking algorithm. However, we leave which visual tracking algorithm to employ as the user's choice since the depth based localizer works regardless of the tracking choice. For instance, for the validation within a SLAM framework, we adopted tracking module ORB-SLAM [23] to tightly couple with the ORB-SLAM framework.

The relative pose between the previous frame and the current frame $\mathbf{T}_n^{n-1}$ is estimated by minimizing an energy function summing over all pixel i given as

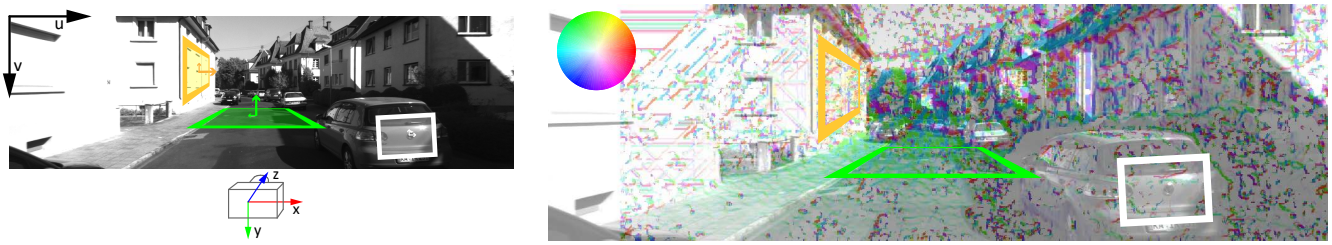
$$E(\xi) = \sum_i \left\| \frac{r_{I,i}^2(\xi)}{\sigma_{r_{I,i}}^2} \right\|_\delta. \tag{5}$$

The energy function is defined as the sum of the error residuals over intensity r_I, divided by their variances, $\sigma_{r_I}^2$. The Huber norm $\|\cdot\|_\delta$ lets the energy function converge well via ignoring outliers. We use the difference in image intensities as the error residual, which is given as

$$r_{I,i}(\xi) = I_{n-1}\left[\pi\left(\mathbf{T}(\xi)\mathbf{T}_n^{n-1} \cdot \pi^{-1}(\mathbf{x}_i, d_i)\right)\right] - I_n[\mathbf{x}_i]. \tag{6}$$

In this equation, $\mathbf{x}_i = [u_i, v_i, 1]^\top$ is a three-dimensional vector which represents the image coordinates of the i^{th} pixel in homogeneous coordinates. The depth corresponds to the i^{th} pixel is denoted as d_i. A corresponding point in 3D space, $\mathbf{p} = [x, y, z, 1]^\top$, is projected onto an image plane via the image projection function $\pi(\cdot)$ as

$$\mathbf{x} = \pi(\mathbf{p}). \tag{7}$$



(a) Three planes in a sample scene

(b) Overlaid depth gradient

Fig. 3. A sample scene and its depth gradients. (a) shows a sample scene and a camera coordinate. Three orthogonal planar surfaces (green, yellow, and white) and their expected depth gradients are depicted with arrows. The depth gradient is expected to be dominant in the u-direction for the yellow plane and in the v-direction for the green plane. The white plane has little depth gradient and no strong value is expected. (b) illustrates the computed depth gradient overlaid with the sample scene. The colormap in the upper left corner represents the two-dimensional vector color code. The depth gradient associated with each pixel follows the color, depending on its direction. The computed depth gradients belonging to each plane are consistent with each other, showing the same color. Furthermore, for pixels belonging to three orthogonal planes, the expected depth gradients are correctly matched with the computed depth gradient in (b).

Vice versa, $\pi^{-1}(\cdot)$ indicates the inverse of the image projection function. Image intensities of the previous and current frames are given as I_{n-1} and I_n. Thus, $I[\mathbf{x}]$ denotes the intensity of an image point at $\mathbf{x}$.

We solved the non-linear optimization problem using *g2o* [24], the general graph optimization framework. Starting with a good initial guess, the relative transformation, $\mathbf{T}_n^{n-1}$, is updated via

$$\mathbf{T}_n^{n-1} \leftarrow \mathbf{T}(\xi) \cdot \mathbf{T}_n^{n-1}, \tag{8}$$

where the increment ξ is calculated from

$$\mathbf{J}^\top \mathbf{\Omega} \mathbf{J} \xi = -\mathbf{J}^\top \mathbf{\Omega}^\top r(0). \tag{9}$$

Here, $\mathbf{J}$ is composed of $\mathbf{J}_i$, the Jacobian matrix of the i^{th} residual $r_{I,i}$. The information matrix $\mathbf{\Omega}$ is composed of diagonal terms with inverse of the error variances.

D. Localization

Visual localization is performed by matching the map points and the current depth map obtained from the stereo camera. While the objective energy function is identical to (5), a depth residual r_D is as described below.

$$r_{D,i}(\xi) = [\mathbf{T}(\xi)\mathbf{T}_M^C \cdot \mathbf{p}_i](3) - \mathcal{D}(\pi(\mathbf{T}(\xi)\mathbf{T}_M^C \cdot \mathbf{p}_i)) \tag{10}$$

This residual on depth is optimized in the localization module instead of the previously adopted intensity residual (6) in the tracking phase. The depth residual is defined as the difference between the depth of the map point and the corresponding stereo depth. After transforming the map point $\mathbf{p}_i$ into the camera coordinates via $\mathbf{T}_M^C$, the depth of the map point can be obtained as $[\mathbf{T}(\xi)\mathbf{T}_M^C \cdot \mathbf{p}_i](3)$, denoting the z-axis value of the transformed map point. For the corresponding stereo depth, the point in the camera coordinates is projected onto the image coordinates via the camera projection function $\pi(\cdot)$, and then the depth is acquired via the depth mapping $\mathcal{D}(\cdot)$.

Optimization is then performed similarly to the way it is performed in the tracking module. The Jacobian matrix of the depth residual is

$$\mathbf{J}_i = [0,0,1,0](\mathbf{T}_M^C \mathbf{p}_i)^\odot - \frac{\partial D(\mathbf{x}_i)}{\partial \mathbf{x}_i} \cdot \frac{\partial \pi(\mathbf{p}_i)}{\partial \mathbf{p}_i} (\mathbf{T}_M^C \mathbf{p}_i)^\odot, \tag{11}$$

where the derivative of the camera projective function $\partial\pi(\mathbf{p}_i)/\partial\mathbf{p}_i$ is given as

$$\frac{\partial \pi(\mathbf{p}_i)}{\partial \mathbf{p}_i} = \begin{bmatrix} \frac{f_x}{z} & 0 & -\frac{f_x}{z^2}x & 0 \\ 0 & \frac{f_y}{z} & -\frac{f_y}{z^2}y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12}$$

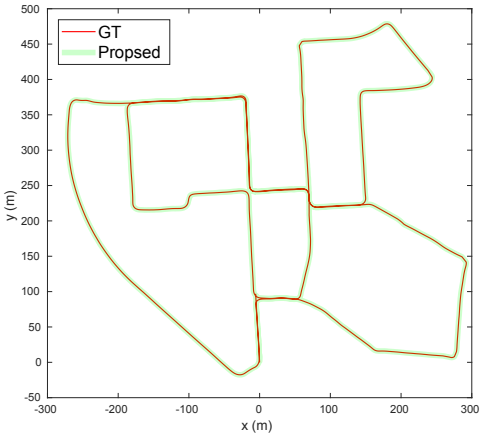
as we assume a projective camera model. The gradient of depth $\partial D(\mathbf{x}_i)/\partial \mathbf{x}_i$ can be retrieved by applying Scharr filtering to the depth map.

The diversity in this depth gradient is important for the localization performance. This relationship can be understood by examining a sample scene with a small depth residual. Fig. 3 shows the sample scene and the depth gradient that we computed from this scene by using the Scharr filtering. Each pixel in Fig. 3(b) is color-coded depending on the depth gradient direction. From Fig. 3, we can understand how the geometry characterized by the depth gradient affects the overall process of camera pose estimation. For instance, when the gradient of the u-axis is larger than that of the v-axis, as in the case of the yellow rectangle, this observation chiefly affects updates in the x-directional pose. This is because the Jacobian dominated by u-axis gradient has a large first element, which encourages x-directional updates. Similarly, when the gradient of the v-axis is dominant, as in the case of the green rectangle, the y-directional pose is updated. If either the u-axis or v-axis gradient is small, as in the case of the third rectangle, the camera pose in the z-axis is updated.

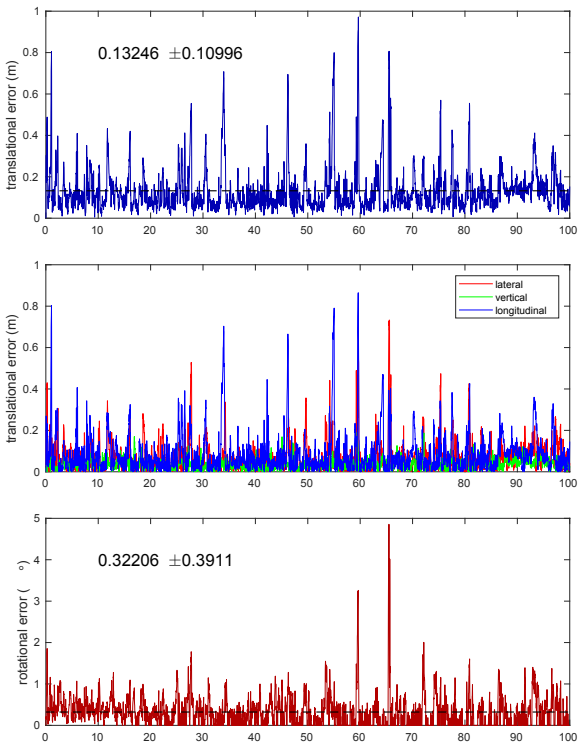
In light of this example, we can see that observations with various depth gradients are suitable for our localization method. However, overly complicated environments full of abrupt differences in depth generated by 3D edges prevents balanced pose estimation. This is because the exact position of edges estimated from stereo depth can sometimes be misleading but the depth gradient caused by an edge facilitates

Sequence #	Translational error [m]	Rotational error [°]
00	0.1325 ± 0.1100	0.3221 ± 0.3911
02	0.2205 ± 0.2918	0.3262 ± 0.5355
03	0.2368 ± 0.2267	0.4133 ± 0.3286
04	0.4496 ± 0.3537	0.8758 ± 0.3857
05	0.1462 ± 0.1427	0.3402 ± 0.4033
06	0.3753 ± 0.6653	0.8485 ± 1.8009
07	0.1305 ± 0.1157	0.4872 ± 0.4718
08	0.1440 ± 0.1755	0.3279 ± 0.5453
09	0.1799 ± 0.2165	0.3375 ± 0.3440
10	0.2398 ± 0.6767	0.4934 ± 0.7690

(a) Localization errors (avg ± std)



(b) Trajectories



(c) Localization Errors

Fig. 4. Result on KITTI `00`. (a) summarizes the localization error in average with standard deviation. (b) shows the ground truth trajectory and the estimated trajectory from our localization algorithm. The three graphs in (c) exhibit localization errors with respect to the percentage of the trajectory. The first graph shows translational error which can be divided into translational errors in lateral, vertical and longitudinal directions as in the second graph. Rotational error is shown in the third graph. The dashed line in the first and third graph indicates the average error value as the average ± standard deviation is displayed at the top of the graph.

a sudden update in the pose. Thus, our strategy is to focus on planes whose positions we can reliably estimate. For this purpose, we set variances in depth residual to be proportional to the magnitude of the depth gradients so that edges can have less effect on the overall energy.

III. Experiment

We validated the proposed method via two different datasets. We used the KITTI odometry dataset [25] and compared our localization performance to that of the recent work of Caselitz *et al.* [13]. To prove the feasibility of the proposed algorithm in a more realistic environment, we further applied the algorithm on the dataset collected by our urban mapping platform. [26]. Localization results on both dataset can also be found from the attached multimedia `camloc.mp4`.

We validated the proposed method in two scenarios, as an independent localization module and as a module in SLAM framework. We first validated our visual localizer module by testing with the publically available KITTI dataset. Each visually localized pose is compared against the baseline provided in KITTI for quantitative evaluations. For the second evaluation, we piped our localizer into the ORB SLAM framework. For a seamless integration, we imported the tracking module from ORB SLAM as the initializer for the localizer.

A. Evaluation on KITTI Dataset

The KITTI odometry dataset provides sequences of images from the synchronized stereo camera, Velodyne HDL-64E LiDAR scans, and the ground truth poses. Extrinsic calibration between the camera and the LiDAR as well as intrinsic/extrinsic calibration for the stereo camera are given in the dataset. We evaluated our algorithm on the provided sequences with the ground truth poses (sequence `00-10`), except for the sequence `01`. Because sequence `01` is a highway sequence, which contains a limited amount of features in the scene, our tracking algorithm failed in this sequence. Since we focus more on urban areas, where GPS is unreliable, we examine other sequences in the evaluation.

We performed the localization by employing the prior map, the initial pose of the camera in the map and the sequence of stereo images. Even if the global map needs to be prepared before the localization, we could detour the mapping process by exploiting the ground truth robot poses. The LiDAR scans, acquired from the sequential robot poses, were transformed into the global coordinates through the corresponding ground truth robot poses and used as local maps. We evaluated the localization accuracy by computing differences between the estimated camera pose and the ground truth pose.

Fig. 4 shows the localization result from the KITTI

sequence 00. The camera trajectory estimated by our localization algorithm is mostly similar to the ground truth trajectory. The graphs below exhibit translational and rotational errors along the sequence. The translational error is always lower than 1.0 m while the rotational error is less than 5°. The average translational error is 0.13 m, and the average rotational error is 0.3°. When we analyzed the error graphs, we found that a larger error occurred at the crossroad where the camera rotates as the car turns while adjacent buildings and parked cars disappear from the scene. We thus infer that the proposed algorithm is not applicable to a situation with fast rotational movements and without nearby structures. However, the proposed method's performance is comparable to that of other methods. Our result on sequence 00 shows lower average translational and rotational errors than the result proposed in [13][1].

The table in Fig. 4(a) summarizes localization errors on the other KITTI sequences. In this table, the errors are averaged over all poses in each sequence and given in average ± standard deviation format. Average translational errors are below 0.5 m and average rotational errors are under 1.0°. Our algorithm shows the best performance on sequence 00 while the worst performance on sequence 04. We conjectured that the limited number of structures in the scene might be the major reason for the performance degradation.

B. Evaluation on Our Dataset

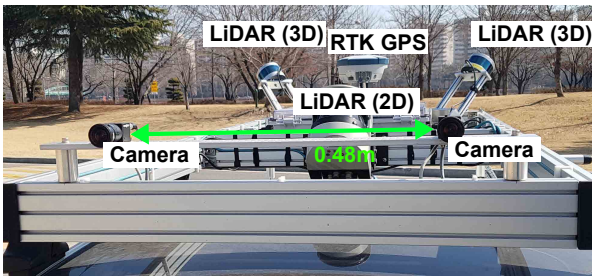
Evaluation of the KITTI dataset revealed sub-meter localization accuracy. However, in the KITTI dataset, most of the scenes were captured from residential areas or highways, with one lane that were surrounded by low-rise buildings. Moreover, the temporal difference between the mapping and localization phase was not sufficiently captured, as we localized against the same sequence. To further evaluate our algorithm in more challenging real-world scenarios and validate the method as a localization module in a SLAM framework, we performed the second experiment using our own dataset by integrating the localizer in the ORB-SLAM. The target environment is challenging due to significant structural variation, wide roads, environmental changes over time, and dynamic objects. Fig. 8(b) depicts the number of satellites varying substantially from location to location, which indicates structural variation.

1) Experimental Setup: Our sensor system described in [26] was equipped with 3D and 2D LiDARs together with navigational sensors. For prior map generation, we exploited baseline trajectory provided in the dataset. The 3D LiDAR map was reconstructed based on this baseline trajectory. For the localization, we additionally mounted a stereo camera on the vehicle's front side, as shown in Fig. 5. For the stereo rig, we mounted two PointGrey Flea3 cameras that provide 1384 x 1032 resolution images at 18 frames per second. Two cameras were mounted facing forward as in Fig. 5 with 0.48 meters apart between them.

2) Intrinsic/Extrinsic Calibration: We estimated the stereo camera's intrinsic and extrinsic parameters using MATLAB stereo camera calibrator application. Additionally, to find the relative pose of the stereo camera in the sensor system, we conducted extrinsic calibration between the left camera and 3D LiDARs. Because the stereo camera observed the front which was opposite to the LiDARs' point of view, direct matching between the camera and LiDARs was impossible. Instead, we used a method of [27], which compared the camera intensity image with the LiDAR intensity image synthesized from the partial map reconstructed through vehicle movement. The relative pose between the camera and LiDARs was obtained by minimizing NID, which measured the dissimilarity between the two images. Fig. 6 exemplifies the result of the extrinsic calibration.

3) Data Acquisition: To apply our algorithm in a more realistic scenario, we obtained two datasets for the same location on different dates (2018-04-11 and 2018-07-13). We generated a map using one dataset and tested our localization algorithm on the map made from th other dataset. The 3D map constructed from the 2018-07-13 dataset appears in Fig. 7. The 2018-04-11 dataset was used for the SLAM validation using the proposed visual localization. In this dataset, the car traveled 15.7 km while taking 34,075 stereo images within the map. We employed pose-graph SLAM again to generate the ground truth camera poses that served as a baseline for the performance evaluation of the

[1]In [13], the average translational and rotational errors of the KITTI sequence 00 were reported as 0.30 m and 1.65°.



(a) Mobile mapping system

Fig. 5. A stereo camera rig mounted on our mapping platform.

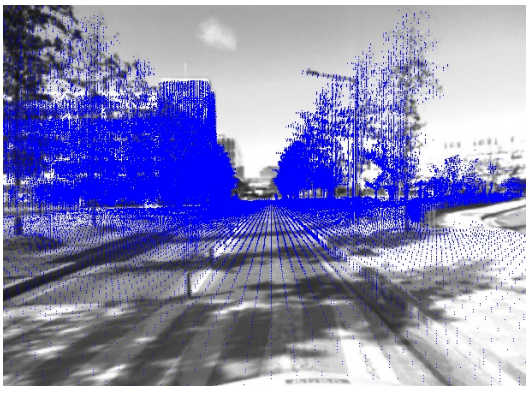


Fig. 6. Camera-LiDAR calibration result. The camera intensity image is overlaid with LiDAR points by using the estimated relative pose between the two sensors.

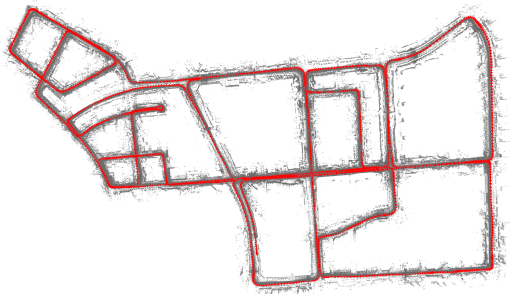


Fig. 7. Global map (gray) and ground truth camera poses (red) acquired from our own sensor system.

TABLE I

STATISTICS FOR EACH MODULE

Module	Localizer	Local ICP
Computational time [ms]	27	1,813
# of frames	2,404	1,459

localization. The red trajectory in Fig. 7 depicts the camera trajectory within the prior map.

4) SLAM using Depth-based Localizer: During the SLAM, we mainly ran the depth-based localizer online while selectively performing local ICP between the reconstructed point cloud from the stereo rig to the prior map only when the depth residual was large. Local ICP is computationally expensive, as can be seen in Table. I, but our selective leverage of this module allowed the online performance of the entire framework.

5) Evaluation: We compared the estimated camera trajectory against the ground truth trajectory generated together with the prior map. Overlaying two trajectories as in Fig. 8(a) provides the qualitative performance evaluation. Note that the estimated trajectory was solely achieved using vision only while the consistency between the two trajectories is preserved (i.e., trajectory estimated by visual localization and the ground truth). The error graphs in Fig. 8(c) give translational and rotational errors over the trajectory. The average translational error is 5.34 m, and the average rotational error is $4.95°$.

Although the localizer in the SLAM performed reliably in most of the regions, localization failures occurred in structure-less regions. The most challenging case for the depth-based localizer was when the roads were wide and nearby structural information was scarce. We restarted the system whenever a localization failure was detected similarly as in [28]. In total 28 restarts were used out of 15.7 km of path length. The magenta lines in Fig. 8(a) and Fig. 8(c) represent the restart points.

Even though the result from our dataset shows lower accuracy than the results from the KITTI dataset, this experiment demonstrated the feasibility of our algorithm in a real scenario. We would like to note that our main objective was to evaluate a vision-only solution within an urban environment. The restarts occurred predominantly when the GPS was reliably received (C, D, and E in Fig. 8(a)). In the most of regions, the algorithm utilized visual information in the structure-rich environment (A and B in Fig. 8(a)), thereby allowing the 15.7 km of the route within a complex urban environment to be covered. We think, therefore, a switchable system between the visual localizer and GPS could be a feasible complementary solution in an urban area.

IV. CONCLUSIONS

We suggested a lightweight stereo camera localization algorithm applied to the previously acquired 3D map prior. Using an initial guess from the tracking, we estimated a full 6 DOF camera pose within the map by minimizing the depth residual. Results from various datasets reveal that our method applies to various platforms, including the publicly available KITTI and our own dataset. The proposed approach could be a complementary solution to consumer-level GPS, especially on narrow streets within a complex urban area, where GPS is highly sporadic.

Future work could enhance this lightweight localizer by integrating it with additional sensors such as GPS and inertial measurement unit (IMU). Further improvement toward invariance to illumination change and robustness to dynamic objects should be considered.

ACKNOWLEDGMENT

This work is supported through a grant from the Korea MOTIE (No.10051867) and [High-Definition Map Based Precise Vehicle Localization Using Cameras and LIDARs] project funded by Naver Labs Corporation.

REFERENCES

[1] H. Carvalho, P. Del Moral, A. Monin, and G. Salut, "Optimal nonlinear filtering in GPS/INS integration," *IEEE Trans. Aerospace and Electronic Sys.*, vol. 33, no. 3, pp. 835–850, 1997.

[2] A. Mohamed and K. Schwarz, "Adaptive kalman filtering for INS/GPS," *Journal of geodesy*, vol. 73, no. 4, pp. 193–203, 1999.

[3] H. Qi and J. B. Moore, "Direct kalman filtering approach for GPS/INS integration," *IEEE Trans. Aerospace and Electronic Sys.*, vol. 38, no. 2, pp. 687–693, 2002.

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

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

[6] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Proc. Robot.: Science & Sys. Conf.*, vol. 2, 2014.

[7] T. Suzuki, M. Kitamura, Y. Amano, and T. Hashizume, "6-DOF localization for a mobile robot using outdoor 3D voxel maps," in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2010, pp. 5737–5743.

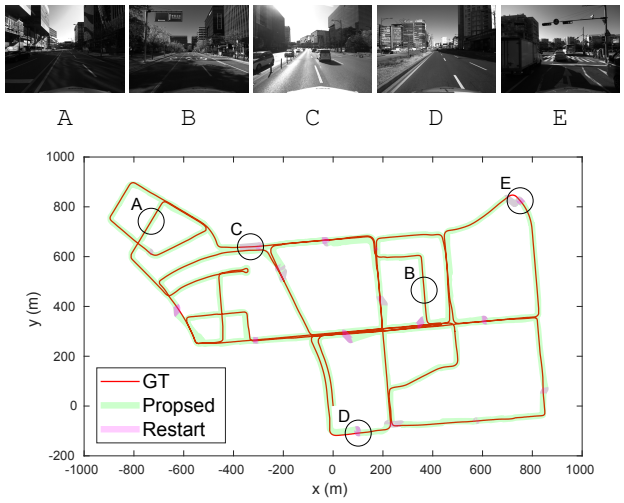
[8] K. Yoneda, H. Tehrani, T. Ogawa, N. Hukuyama, and S. Mita, "Lidar scan feature for localization with highly precise 3-D map," in *Proc. IEEE Intell. Vehicle Symposium*, 2014, pp. 1345–1350.

[9] P. Ruchti, B. Steder, M. Ruhnke, and W. Burgard, "Localization on openstreetmap data using a 3D laser scanner," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2015, pp. 5260–5265.

[10] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Proc. Robot.: Science & Sys. Conf.*, 2009.

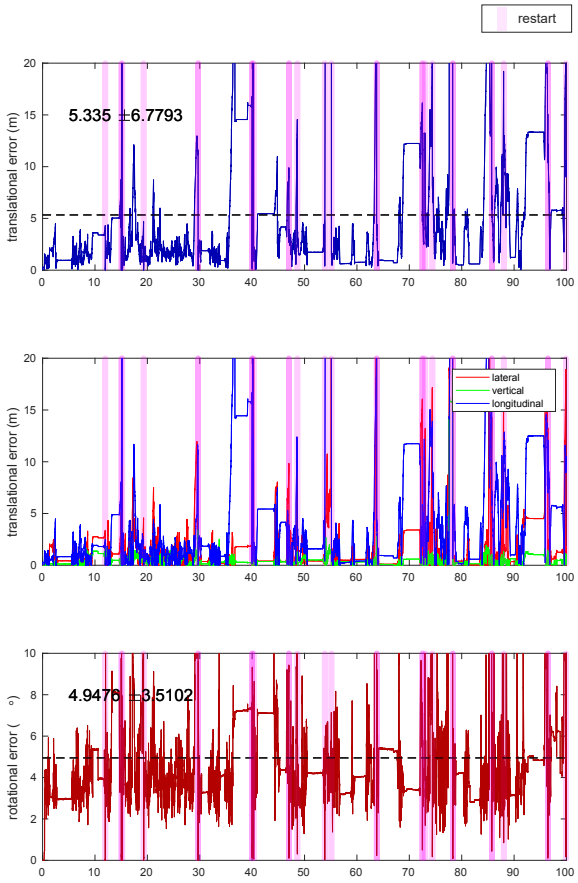
[11] A. D. Stewart and P. Newman, "LAPS-localisation using appearance of prior structure: 6-dof monocular camera localisation using prior pointclouds," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2012, pp. 2625–2632.

[12] R. W. Wolcott and R. M. Eustice, "Visual localization within LIDAR maps for automated urban driving," in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2014, pp. 176–183.



(a) Trajectories

(b) Number of available satellites



(c) Localization Errors

Fig. 8. Result on our dataset. (a) shows the ground truth trajectory and the estimated trajectory from our localization algorithm. Sample images of selected regions on the path are provided in the top row. The number of available satellites along the route is given in (b). The three graphs in (c) exhibit localization errors with respect to the percentage of trajectory. The first graph shows translational error which can be divided into translational errors in lateral, vertical an longitudinal directions as in the second graph. Rotational error is shown in the third graph. The magenta bar over the graphs represents each restart.

[13] T. Caselitz, B. Steder, M. Ruhnke, and W. Burgard, "Monocular camera localization in 3D lidar maps," in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2016, pp. 1926–1931.

[14] P. Neubert, S. Schubert, and P. Protzel, "Sampling-based methods for visual navigation in 3D maps by synthesizing depth images," in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2017, pp. 2492–2498.

[15] Y. Xu, V. John, S. Mita, H. Tehrani, K. Ishimaru, and S. Nishino, "3D point cloud map based vehicle localization using stereo camera," in *Proc. IEEE Intell. Vehicle Symposium*, 2017, pp. 487–492.

[16] C. Forster, M. Pizzoli, and D. Scaramuzza, "Air-ground localization and map augmentation using monocular dense reconstruction," in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2013, pp. 3971–3978.

[17] J. Engel, T. Sch, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. European Conf. on Comput. Vision*, 2014, pp. 834–849.

[18] D. Meagher, "Geometric modeling using octree encoding," *Computer graphics and image processing*, vol. 19, no. 2, pp. 129–147, 1982.

[19] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2011, pp. 1–4.

[20] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*, 2008.

[21] H. Hirschmuller, "Accurate and efficient stereo processing by semi-global matching and mutual information," *IEEE Trans. Pattern Analysis and Machine Intell.*, vol. 30, no. 2, pp. 328–341, 2008.

[22] G. Sibley, L. Matthies, and G. Sukhatme, "Bias reduction and filter convergence for long range stereo," *Robot. Res.*, pp. 285–294, 2007.

[23] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular slam system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, 2015.

[24] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g 2 o: A general framework for graph optimization," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2011, pp. 3607–3613.

[25] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *Proc. IEEE Conf. on Comput. Vision and Pattern Recog.*, 2012, pp. 3354–3361.

[26] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim, "Complex urban lidar data set," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, Brisbane, May. 2018, in print.

[27] T. Scott, A. A. Morye, P. Piniés, L. M. Paz, I. Posner, and P. Newman, "Exploiting known unknowns: Scene induced cross-calibration of lidar-stereo systems," in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2015, pp. 3647–3653.

[28] A. D. Stewart, "Localisation using the appearance of prior structure," Ph.D. dissertation, University of Oxford, 2014.