LIV-LAM: LiDAR and Visual Localization and Mapping

Reza Radmanesh, Ziyin Wang, Vishnu S. Chipade, Gavriil Tsechpenakis, and Dimitra Panagou

Abstract— **This paper presents a framework for Simultaneous Localization and Mapping (SLAM) by combining a novel method for object discovery and localization from a monocular camera with depth information provided by Light Detection and Ranging (LiDAR). One major challenge in vision is discovering unknown objects without prior training/supervision, in the wild, and on-the-fly. In our framework, no training samples are available prior to the deployment. We develop an efficient proposal-matching method to discover object temporal saliency, and then fine-tune these frequently matched object proposals according to tracking information. Detected features of the objects are used as landmark features, and are merged with the LiDAR data in the proposed LIV-LAM (LiDAR and Visual Localization and Mapping). Compared to most visual SLAM or LiDAR-based SLAM, the novelty of this method is the computationally-efficient object detection and localization for feature set-and-match, in order to increase the accuracy of the generated map. The results show that the presented method is superior in both accuracy and efficiency of the maps generated by LiDAR.**

I. Introduction

Many SLAM studies have considered visual sensory data from either monocular cameras or RGB-D sensors, and a few others have been studying the effect of depth information obtained from LiDAR data for SLAM process [18]. For 3-D perception of outdoors robots and autonomous vehicles, the fusion of 3D LiDAR and camera data has been one of the most acceptable practical methods [10]. The combination of the available sensory information maintains the 3D spatial structure through SLAM, and it enables vehicle path planning, obstacle avoidance, and robot manipulation.

Using LiDAR measurements to enhance motion estimation and mapping is performed through conventional approaches such as Iterative Closest Point (ICP) [15], [16]. Recently, considerable progress has been made in motion estimation and mapping by using a monocular camera using both direct and indirect feature-based methods [13], [8], [6]. Proposed visual and LiDAR-based SLAM methods showed several issues such as scale-ambiguity of single camera, or inaccurate motion estimation due to the vertical measurement sparsity of the dense LiDAR point cloud [2], [7].

In this work, we propose a method called LIV-LAM, in which data collected from the monocular camera as well as the sparse depth information obtained from 3D LiDAR are combined into a SLAM process. In this paper, due to difference in Field-of-View (FOV) of each sensors (i.e. camera and LiDAR), we will consider unsupervised object discovery with narrow field-of-view (FOV) and then project 3D LiDAR data to guarantee the depth association to the discovered objects. The main characteristics of the proposed method can be itemized as follows:

- We propose a computationally efficient unsupervised learning for object discovery, for localization and tracking of potential targets using both monocular camera and LiDAR.
- Compared to other SLAM processes, the proposed method uses a pose-graph optimization for dealing with both computational performance and sensors with different FOVs.
- A new concept of key-frame is proposed to collect the key information from the environment, and then a key-frame-based loop constraint is introduced.
- The performance of the proposed method has been evaluated with ground-truth data, and data directly acquired from various environments.

A. Related Studies

In SLAM methods, the most dominant approach for motion estimation using cameras are the so-called indirect methods (feature-based methods) [9], [19], [12]. One of the drawbacks of these methods is the performance in simple (featureless) environments, or within environments with repeated patterns. The most recently developed direct methods for motion estimation have alleviated the existence of feature issues without requiring the feature correspondences. In direct methods, the feature-based projection error model has been replaced by the photometric error model that compares the intensity difference of a pixel during different time instances [7]. However, in the case of a monocular camera, bootstrapping method and Graphics Processing Unit (GPU) are used to provide estimation for both a full-depth map and motion estimation [14]. More recently, a CPU-based method was alternatively implemented [6]. Direct Sparse Odometry (DSO) is introduced in [5], [18], which uses a fixed number of points and performs a window-based optimization that ensures a high accuracy of motion estimation. Differing from the aforementioned methods, the use of LiDAR data would increase the motion estimation accuracy in a camera-LiDAR

M. Radmanesh is with NASA Jet Propulsion Lab (JPL), Pasadena, California 91109 and University of Michigan, Ann Arbor, MI, USA, Reza.Radmanesh@jpl.nasa.gov , Mradmane@umich.edu

Ziyin Wang and Gavriil Tsechpanakis are with Indiana University - Purdue University Indianapolis, Indianapolis, USA wang2457@purdue.edu, gtsechpe@indiana.edu

Dimitra Panagou and Vishnu S. Chipade are with University of Michigan, Ann Arbor, MI, USA, dpanagou@umich.edu, vishnuc@umich.edu

M. Radmanesh and D. Panagou wish to acknowledge the support of an Early Career Faculty grant from NASAs Space Technology Research Grants Program.

system, and solve the scale-ambiguity problem [18]. Direct methods were also successfully implemented with RGB-D sensors. In other approach to enhance the performance of direct methods of feature extraction for visual SLAM, adding the geometric errors along with a dense depth of information represented promising results [22], [23]. One of the most recent and successful application of a direct method with RGB-D sensory information for motion estimation is the Depth-Enhanced Monocular Odometry (DEMO) [23]. DEMO results in high-precision visual odometry estimation using a tightly coupled camera and LiDAR data.

A low-drift, real-time LiDAR odometry and mapping method, known as LOAM, is proposed in [24] and [17]. Low computational complexity in real-time is the benefit of using LOAM [24], [25], [17]. LOAM can perform mapping in real time since it matches and registers the point cloud at a lower frequency, while performing odometry at a higher frequency. This method performs point feature to edge/plane scan-matching to find correspondences between scans. Features are extracted by calculating the roughness of a point in its local region. A variation of this method called LeGO-LOAM (Lightweight and Ground Optimized Lidar Odometry and Mapping) is proposed in [17]. Inspired by the performance of the low-complexity of LOAM and LeGO-LOAM, here we combine LOAM and a monocular camera to perform SLAM in various environments with varying complexity.

B. Paper Overview

The rest of the paper is organized as follows. The problem addressed and the overview of the proposed approach are presented in sections II and III, respectively. Section IV presents our approach for on-the-fly unsupervised discovery of unknown objects. Localization using visual and LiDAR data is discussed in Section V. The results on the performance of the algorithm are discussed in Section VI.

II. Problem Statement

Figure 1 shows the full setup of the considered problem. Subsets of SLAM problem, include placing of the objects in global map, are considered in the paper.

The proposed architecture in this paper, uses the pose-graph SLAM formulation, in which a set of odometry constraints u_i are imposed between two successive poses x_i and x_{i+1}, so that we have:

$$x_{i+1} = f(x_i, u_i) + w_i, \tag{1}$$

where f is a nonlinear function which defines the sensor's motion model. Usually in SLAM architectures, the front-end (i.e. the part where the heavy computation is generated) part of the system is responsible for detecting loop closures between two poses x_i and x_j. The front-end can detect these loop closures based on visual recognition (sensory vision), or by LiDAR as in [20], [3], [11]. Loop closure imposes another constraint into the optimization of SLAM as follows:

$$x_j = f(x_i, u_{ij}) + \lambda_{ij} \tag{2}$$

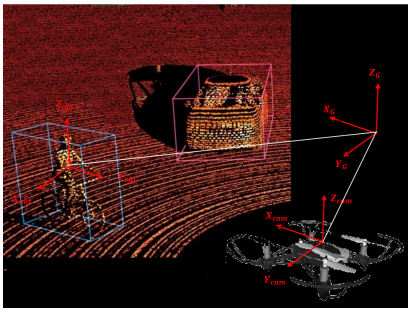


Fig. 1: General scheme of the problem. In this paper, every object has an attached body frame $\{x_{obj}, y_{obj}, z_{obj}\}$. Furthermore, each of the sensors on the robot would have another body frame $\{x_{cam}, y_{cam}, z_{cam}\}$ and there is a global frame $\{x_G, y_Y, z_Z\}$. The origin of global frame is located at initial location of the sensors

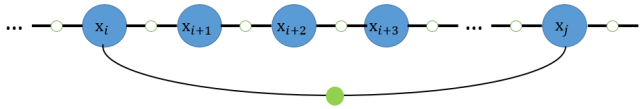


Fig. 2: Pose-graph SLAM problem. x_i represents the sequence of the collected unknown poses of the robot. Each pose is connected with the odometry constraint, as well as a loop-closure factor constraint between two non-successive poses of x_i and x_j.

In equation 1 and 2, $\lambda_{i,j}$ and w_i are the error model terms with covariances of $\sum_i$ and Γ_{ij}. As shown in figure 2, here we consider $X = \{x_i\}$ as the sensor poses, and $U = \{u_i \cup u_{ij}\}$ as the set of constraints of odometry and loop closures, respectively. Then, the optimal robot poses X^* expressed in the global frame are given by:

$$\begin{aligned} X^* &= \underset{X}{\text{argmin}} \sum_i ||f(x_i, u_i) - x_{i+1}||^2_{\sum_i} \\ &+ \sum_{ij} ||f(x_i, u_{ij}) - x_j||^2_{\Gamma_{ij}}. \end{aligned} \tag{3}$$

In equation 3, the $||.||^2_{\sum}$ represents the Mahalanobis distance with covariance $\sum$. Equation 3 is a nonlinear least-squares problem that can be solved with the Levenberg-Marquardt method [20].

III. Software Architecture and Breakdown

The proposed method consists of a LiDAR-based odometry algorithm, camera-based object detection and association, loop-closure detection, and pose-graph optimization, as shown in Figure 3.

Instead of constantly processing the images as it is explained in Section IV, the reduction in processing images originates from the fully decoupling camera-tracking (i.e. searching for the same features to match with the previous features of the collected frames) and LiDAR Mapping and Localization. This indirectly results in reducing the computational effort of the method. Here, the object detection

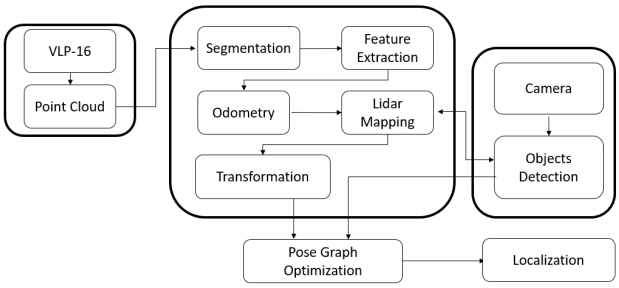


Fig. 3: Proposed hardware and software architecture

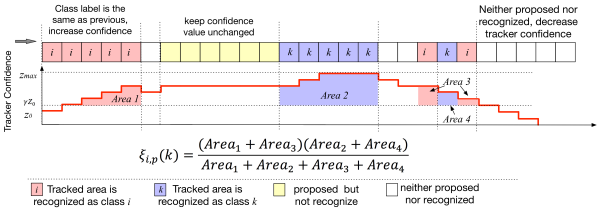


Fig. 4: Tracker Confidence Curve: Integration of object discovery and tracking for object correspondence inferring. The horizontal axis is the time and the vertical axis is the confidence value.

is performed only within the proposed key-frames, and the global positions of the key-frames are recorded within the map. Tracking the future camera images would be enabled within the vicinity of the global location of images. The key-frames are then added as a feature to localization and mapping pose-graph. The key-frame notion of this method is used for loop closures and in case of recovering the failed localization (i.e. loosing the location of the sensor), re-starting the whole process.

IV. Unsupervised Unknown Object Discovery On-the-Fly

We apply progressive visual learning at three levels: for low-level features, for temporally-salient objects, and for discovered class correspondences, all without prior knowledge, along the image stream. Due to the turbulence of the onboard camera that often yields blurring in the acquired frames and degrades the detection performance, we select key frames every 10 consecutive frames and we calculate the global image gradient. For each key frame we extract features (L2 normalized SIFT and L2 normalized color) from (weak) object proposals [26], and feed those features into stream clustering [21] for unsupervised feature learning, to progressively learn a feature code-book on-the-fly. This low-level visual code-book is continually updated and enlarged during data acquisition as new visual categories keep emerging. This way, we obtain a proposal window description using the distribution of its enclosed feature categories (bag-of-features paradigm), which we use for higher-level stream clustering to obtain proposal categories. We track each categorized proposal across consecutive key frames, and adapt tracking according to the detection results as explained below.

A. Matching proposals on-the-fly.

The first problem we tackle is discovering visual saliency on-the-fly while the feature space increases over time. Let histogram h_i of low-level feature cluster assignments with dimension d_i encode a proposal window. Since the feature space is growing over time, d_i is also increasing. The histograms of the same proposed region at two different instances along the stream have the same d elements, with d being the size of the preceding histogram, since they correspond to common feature cluster assignments (in the preceding and the updated feature code-book).

Given a proposal histogram h_k with dimension d_k, and the corresponding temporally preceding histogram h_i, we update h_k as

$$ {}^{(d)}h_k^{(new)} = \begin{cases} \frac{{}^{(d)}h_k n_{kd} + {}^{(d)}h_i}{n_{kd}+1}, & \text{if } d \leq d_k; \\ {}^{(d)}h_i, & \text{if } d > d_k; \end{cases} \quad (4) $$

where ${}^{(d)}h_k$ and ${}^{(d)}h_i$ are the d^{th} elements of h_k and h_i, respectively; n_{kd} is the number of matched data points for the d^{th} dimension of h_k: we record the number of image features within the proposal that are assigned to each cluster (d-th element/dimension of the histogram), which in turn indicates how many times the d-th element has been updated so far. We set $n_{kd} = 1$ for $d > d_k$. Note that q. (4) degrades to the moving average if $d_k \equiv d_i$. To compare two proposal histograms, we use the cosine similarity $S(h_k, h_i) = cos(h_k, h_{id_k})$, where h_{id_k} is the first d_k elements of h_i.

B. Correspondence Inference by Tracking

Tracking provides valuable information about visual diversity, where each 'confidently' tracked proposal/image patch is considered to belong to the same class (cluster) over time. For each tracker/patch, we maintain a Tracker Confidence Curve as shown in Figure 4 (also see below). We perform object discovery and recognition every m frames (in each key frame), and track the categorized patches through these m frames without explicit encoding and clustering. The Tracker Confidence Curve has two values for each key frame: categorized object class l_t at time t, and tracker confidence c_t. The tracker confidence is initialized to z_0 and ranges from 0 to z_{max} (see Figure 4). We stop tracking when the tracker confidence decreases to 0. Given the value of tracker confidence is initialized or updated at time t, its value is updated at time $t+m$ according to the following 4 situations: (i) The tracked object is successfully categorized at time $t + m$ as the corresponding object class at t. In this case, we increase the confidence value by 1. If the confidence value increases to z_{max} we keep the value unchanged. (ii) The tracked object is proposed but not categorized (does not belong to any existing proposal clusters). In this case, we keep tracking and wait for the tracked object proposal to be learned later (if a new proposal cluster emerges over time). In this paper, confidence value is not changed in this case. (iii) The tracked area is categorized at $t + m$ as a different

class than the corresponding object at t. This case directly indicates class correlation. However, any uncertainty may also result into such class transition. Thus, we decrease the tracker confidence by 1. If the tracked area is categorized as the new class again in the next key frame, the tracker confidence will increase by 1. (iv) The tracked object is neither proposed nor recognized. We decrease the tracker confidence by 1 and keep tracking.

As shown in Figure 4, we can use the area under the curve for each particular object class to describe the quality of tracking. Given a tracker initialized at time t_0 and diminished at time T, we define 'object confidence' of class i as,

$$A_i = \sum_{t=t_0}^{T} \sigma\big(l_t, i\big) \cdot \max\big(c_t - \gamma z_0, 0\big) \tag{5}$$

where $\sigma(l_t, i) = 1$, if $l_t = i$, and $\sigma(l_t, i) = 0$ otherwise, l_t and c_t are the recognized class label and the tracker confidence of time t, respectively, and γ is a constant tolerance value. A_i is the area under the curve when the tracked patch is recognized as class i. Obviously, large values of A_i indicate that object class i is stably tracked. Thus we can describe the semantic correspondence between class i and k of tracker p as:

$$\xi_{i,p}(k) = \frac{A_i A_k}{\sum_{j=1}^{K} A_j} \tag{6}$$

where K is the total number of classes discovered so far.

Here, we do not rely on the information from a single tracker. Rather, we model the semantic correspondences over all discovered object classes. First, we select an object class i as the dominant class, such that $\frac{A_i}{\sum_{k=1}^{K} A_k}$ is maximized (the most stably tracked class). The value of $\xi_{i,p}(k)$ indicates the number of instances that class i and a class k are correlated. We use those instances to learn (maximize) a Dirichlet posterior distribution, where we consider a conjugate Dirichlet Prior $p(\boldsymbol{\alpha_i}|\boldsymbol{\beta}) = Dir(\boldsymbol{\beta})$, where $\boldsymbol{\alpha_i} = [\alpha_{i,k}]_{k=1,...,K-1,k\neq i}$ is the semantic correspondence vector between class i and all other discovered classes k, and $\boldsymbol{\beta} = \beta[1, ..., 1]$. The solution to this model is,

$$\alpha_{i,k} = \frac{\beta + m_{i,k} - 1}{(K-1)(\beta-1) + \sum_{k=1,k\neq i}^{K} m_{i,k}}, \tag{7}$$

where $m_{i,k} = \sum_p \xi_{i,p}(k)$ indicates the total correspondence between class i and class k over all previous trackers p. For each discovered object class i, the semantic correspondence vector $\boldsymbol{\alpha_i}$ is updated along the stream. Each element $\alpha_{i,k}$ indicates the learned correspondence strength between class i and k. We use the entropy of $\boldsymbol{\alpha_i}$ to discover whether a correspondence pattern emerges. Entropy is defined as $E(\boldsymbol{\alpha_i}) = -\sum_{k\neq i} \alpha_{i,k} \ln(\alpha_{i,k})$, which we normalize with respect to K, the number of the discovered classes,

$$E_n(\boldsymbol{\alpha_i}) = \frac{E(\boldsymbol{\alpha_i})}{-\sum_{k\neq i} \frac{1}{K-1} \ln(\frac{1}{K-1})} = \frac{E(\boldsymbol{\alpha_i})}{\ln(K-1)} \tag{8}$$

$E_n(\boldsymbol{\alpha_i})$ ranges from 0 to 1. When $E_n(\boldsymbol{\alpha_i})$ becomes sufficiently small (e.g., $E_n(\boldsymbol{\alpha_i}) \leq 0.2$), the correspondence pattern is confidently discovered from previous trackers. The final semantic correspondences are calculated by finding the connected components in a dynamic undirected graph, where each vertex represents a discovered object class and each edge represents a semantic correspondence. Initially, this graph has no edges, and it is updated every time a tracker diminishes.

V. Object Localization in the Environment

1) Part 1: LiDAR Odometry and Data Collection: We define the entire scheme of the LiDAR odometry based on [25] and [17]. Consider $Pcl_t = \{p_{1,t}, p_{2,t}, \cdots, p_{n,t}\}$ to be point cloud (Pcl) data (raw data) acquired from the LiDAR at time t. Pcl_t is projected to a range image with resolution of 1800 by 16 (due to the LiDAR VLP-16 features). A column-wise evaluation of the range image (output from the LiDAR data) [1] is applied to the series of range images to form a cluster the PCls with unique labels. Revisiting an object in two consecutive LiDAR scans is the criterion for forming a LiDAR-based cluster in a noisy environment. By removing PCls with small sizes which form clusters, large objects within the range image are formed and can be utilized for LiDAR-based feature extraction. Similar to [25] and [17], we consider the roughness of the segmented points S for the purpose of point clustering. We select feature points that are on sharp edges, and on planar surface patches. Let $p_{i,t}$ be a point at Pcl_t, and S be the set of consecutive points of $p_{i,t}$ returned by the VLP-16 in the same scan. Since the VLP-16 laser scanner generates points and returns the data in CW or CCW order, S contains half of its points on each side of $p_{i,t}$. Several other parameters are involved for generating s which is out of scope of this paper and can be found in [17]. The roughness of the segmented points c is given as [25]:

$$c = \frac{1}{|S|.\|r_i\|} \left\| \sum_{j\in S, j\neq i} (r_j - r_i) \right\|, \tag{9}$$

where r_i is the distance between the LiDAR and the each point in Pcl_i. The roughness threshold, as explained in [17], indicates the planar-feature ($c < c_{threshold}$) and edge-feature ($c \geq c_{threshold}$) of the segmented PCls, and is used to determine the features of PCls at each scan [17], [24]. The presented features, i.e., planar and edge features, are used as an input to find the LiDAR odometry, for estimating the motion between two consecutive scans. This process is performed by finding the features of points in the current scan (for both edges and plannar features, represented by F_e^t and F_p^t, respectively) from feature sets of previous scan $\mathbb{F}_e^{t-1} = \{F_{e_1}^{t-1}, F_{e_2}^{t-1}, \cdots, F_{e_{N1}}^{t-1}\}$ and $\mathbb{F}_p^{t-1} = \{F_{p_1}^{t-1}, F_{p_2}^{t-1}, \cdots, F_{p_{N2}}^{t-1}\}$. $N1$ and $N2$ represents the number of edges and planar features in the map, respectively.

Part 2: Visual Inference and Detection The proposed mapping consists of: matching features of $\{\mathbb{F}_e^t, \mathbb{F}_p^t\}$, and synchronizing the camera object discovery to proposing new feature O^t from vision. These features are then added to the point-cloud map to find the position (refer to pose Pos) transformation (i.e. movement from the current point of the

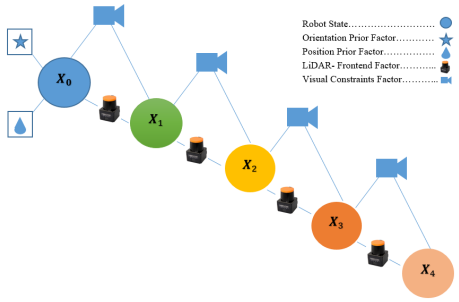


Fig. 5: Optimization method using LiDAR and camera with GTSAM representation

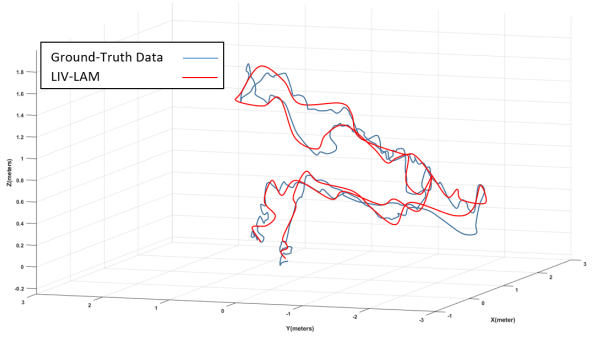


Fig. 6: Ground-truth vs LIV-LAM trajectories. The key-frames are generated every 1 second, which contributes to the drift of the LiV-LAM estimates from the ground-truth data.

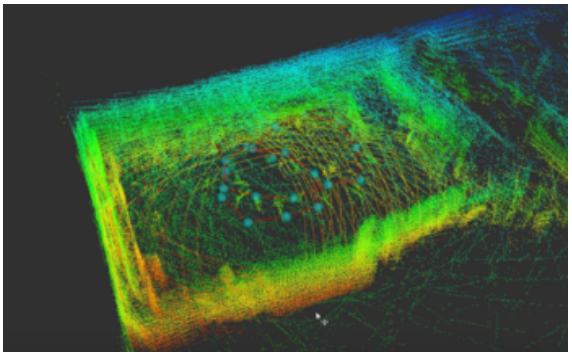


Fig. 7: Point-cloud data generated by the LIV-LAM without loop-closure.

LiDAR comparing to the previous collected scan pose). Let us define ${}^{t}_{G}T_{t-1}$ as the transformation of sensor system frame (LiDAR and camera) with respect to global-fixed frame from time step $t-1$ to t. O^t is the feature-set of objects discovered from camera, in key-frames, which can be saved upon discovery at certain time t. This transformation can be expressed as the following equation:

$$ {}^{t}_{G}Pos = {}^{t}_{G}T_{t-1} \quad {}^{t-1}_{G}Pos, \tag{10}$$

where ${}^{t}_{G}T_{t-1}$ represents the transformation of sensor system frame (LiDAR and camera) with respect to global-fixed frame from time step $t1$ to t. O^t is the feature-set of objects discovered from camera, in key-frames, which can be saved upon discovery at certain time t.

2) Part 3: Optimization Process: In the presented format, the front-end consists of the method of LOAM and the back-end fuses the camera inference with the LiDAR data in GTSAM framework. At certain time step t, the feature set $\left\{\{F_p^1, F_e^1, O^1\}, \{F_p^2, F_e^2, O^2\}, \cdots, \{F_p^{t-1}, F_e^{t-1}, O^{t-1}\}\right\}$ is the set of the features which the current scan should be compare with. The optimization for adding the new new node (pose) with feature set of $\{F_p^t, F_e^t, O^t\}$ to the pose-graph containing nodes with feature set of $\left\{\{F_p^1, F_e^1, O^1\}, \{F_p^2, F_e^2, O^2\}, \cdots, \{F_p^{t-1}, F_e^{t-1}, O^{t-1}\}\right\}$ would be performed by GTSAM. This solver is capable of solving the optimization of pose-graphs through different methods such as Levenberg-Marquardt (LM) [4].

VI. Results and Discussion

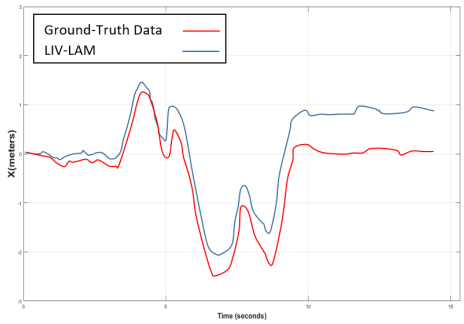
LIV-LAM is validated using a monocular camera and a LiDAR sensor. The proposed method was applied to our portable camera-LiDAR sensor system. All experiments were performed on a Desktop with an Intel Core i5 processor and 8GB RAM. The USB camera used is of resolution 800×600, frame rate of 30 Hz, and a horizontal FOV of $90°$. The LiDAR used is a VLP-16 that provides a sparse point cloud from 16-ray range measurements. Another key for correct performance of the LIV-LAM is the precise calibration between the sensors. We used the calibration method in [18] to achieve intrinsic calibration of the camera and extrinsic calibration between the camera and LiDAR.

A. Indoor Camera-LiDAR Dataset with Ground-Truth Data

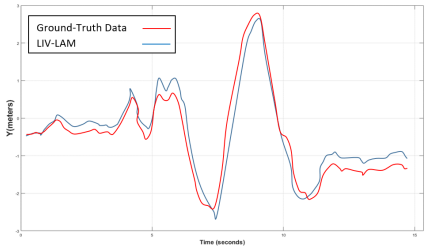
Ground truth data for evaluating the proposed localization technique in an indoor environment were collected from a VICON motion camera system. In this experiment, a LiDAR and camera system is moved in an indoor environment with several objects to discover. The qualitative results of the overall trajectory and the resulting point cloud maps are shown in Figure 6. Figure 6 illustrates that the ground-truth measurements and the LIV-LAM estimates are fairly consistent. The point cloud map obtained from the experiment is shown in figure 7. As the points are accumulated by the motion of the sensor system (LiDAR and camera), the map data are acquired for the whole area. From the results, it can be seen that there is a drift in the map which can be corrected later on by finding a loop closure but in this data-set due to the relatively small range of motion, the loop-closure was not considered in this experiment. As it can be seen, for short duration of motion, LIV-LAM is yielding a point cloud map of acceptable quality. Figure 8 represents the position trajectories of the sensors over time expressed in the global frame as obtained from the LIV-LAM technique, and from the VICON motion capture system, respectively.

VII. Conclusion and Future Work

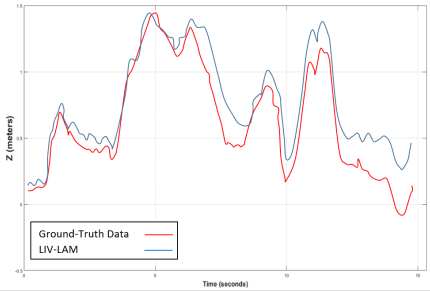
In this work, we introduced a SLAM method using a camera-LiDAR sensor system. The method consists of learn-



(a) Motion in X axis over time



(b) Motion in Y axis over time



(c) Motion in Z axis over time

Fig. 8: Position trajectories with respect to time.

ing unknown objects using camera data in an unsupervised manner, and using the learned objects as landmarks for the SLAM process. The pose-graph scheme is used for formulating the SLAM problem to add new poses to the previously collected nodes. The new pose contains features of the detected objects and the LiDAR data. The method is evaluated using experimental trials in several indoors environments. In future work, we plan to extend our approach to outdoors environments and featureless areas such as tunnels. Furthermore, we plan to exploit LIV-LAM in closing the loop for visually-driven planning and control of autonomous vehicles.

REFERENCES

[1] I. Bogoslavskyi and C. Stachniss. Fast range image-based segmentation of sparse 3D laser scans for online operation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 163–169. IEEE, 2016.

[2] J. Civera, A. J. Davison, and J. M. Montiel. Inverse depth parametrization for monocular SLAM. *IEEE transactions on robotics*, 24(5):932–945, 2008.

[3] M. Cummins and P. Newman. FAB-MAP: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008.

[4] F. Dellaert. Factor graphs and GTSAM: A hands-on introduction. Technical report, Georgia Institute of Technology, 2012.

[5] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017.

[6] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: large-scale direct monocular SLAM. In *European conference on computer vision*, pages 834–849. Springer, 2014.

[7] C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 15–22. IEEE, 2014.

[8] C. Kerl, J. Sturm, and D. Cremers. Robust odometry estimation for RGB-D cameras. In *2013 IEEE International Conference on Robotics and Automation*, pages 3748–3754. IEEE, 2013.

[9] B. Kitt, A. Geiger, and H. Lategahn. Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme. In *2010 ieee intelligent vehicles symposium*, pages 486–492. IEEE, 2010.

[10] R. Li, S. Wang, and D. Gu. Ongoing evolution of visual SLAM from geometry to deep learning: Challenges and opportunities. *Cognitive Computation*, 10(6):875–889, 2018.

[11] W. Maddern, M. Milford, and G. Wyeth. CAT-SLAM: probabilistic localisation and mapping using a continuous appearance-based trajectory. *The International Journal of Robotics Research*, 31(4):429–451, 2012.

[12] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[13] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*, pages 2320–2327. IEEE, 2011.

[14] M. Pizzoli, C. Forster, and D. Scaramuzza. REMODE: Probabilistic, monocular dense reconstruction in real time. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2609–2616. IEEE, 2014.

[15] A. Segal, D. Haehnel, and S. Thrun. Generalized-ICP. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA, 2009.

[16] J. Serafin and G. Grisetti. NICP: Dense normal based point cloud registration. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 742–749. IEEE, 2015.

[17] T. Shan and B. Englot. LeGO-LOAM: lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765. IEEE, 2018.

[18] Y.-S. Shin, Y. S. Park, and A. Kim. Direct visual SLAM using sparse depth for camera-lidar system. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[19] H. Strasdat, A. J. Davison, J. M. Montiel, and K. Konolige. Double window optimisation for constant time visual SLAM. In *2011 International Conference on Computer Vision*, pages 2352–2359. IEEE, 2011.

[20] N. Sünderhauf and P. Protzel. Towards a robust back-end for pose graph SLAM. In *2012 IEEE International Conference on Robotics and Automation*, pages 1254–1261. IEEE, 2012.

[21] Z. Wang and G. Tsechpenakis. Stream clustering with dynamic estimation of emerging local densities. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2100–2105. IEEE, 2018.

[22] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald. Real-time large-scale dense rgb-d slam with volumetric fusion. *The International Journal of Robotics Research*, 34(4-5):598–626, 2015.

[23] J. Zhang, M. Kaess, and S. Singh. A real-time method for depth enhanced visual odometry. *Autonomous Robots*, 41(1):31–43, 2017.

[24] J. Zhang and S. Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, page 9, 2014.

[25] J. Zhang and S. Singh. Low-drift and real-time Lidar odometry and mapping. *Autonomous Robots*, 41(2):401–416, 2017.

[26] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *European conference on computer vision*, pages 391–405. Springer, 2014.