Sensors and Materials, Vol. 35, No. 1 (2023) 167–181
MYU Tokyo

S & M 3161

A Model of Real-time Pose Estimation Fusing Camera and LiDAR in Simultaneous Localization and Mapping by a Geometric Method

De Chen,[1,2] Qingdong Yan,[1*] Zhi Zeng,[3] Junfeng Kang,[4] and Junxiong Zhou[2]

[1]School of Mechanical Engineering, Beijing Institute of Technology,
South Street No. 5, Zhongguancun, Haidian District, Beijing 100081, China
[2]Guangdong Provincial Key Laboratory of Intelligent Lithium Battery Manufacturing Equipment Enterprises,
Xinpeng Rd. 4, Ma'an County of Huicheng District, Huizhou 516057, China
[3]School of Computer Science and Engineering, Huizhou University,
Yanda Rd. 46, Huicheng District, Huizhou, 516007, China
[4]School of Civil and Surveying & Mapping Engineering, Jiangxi University of Science and Technology,
Hongqi Rd. 86, Zhanggong District, Ganzhou 341000, China

(Received October 31, 2022; accepted January 16, 2023)

Keywords: light detection and ranging (LiDAR), RGB-D (RGB-depth map), robot, simultaneous localization and mapping (SLAM), pose estimation, minimum bounding rectangle (MBR)

Simultaneous localization and mapping (SLAM) is the key technology for achieving autonomous navigation and stable walking for robots. For addressing a dynamic and special environment indoors and outdoors, there are still some limitations in using a single sensor to estimate and locate a robot's position and orientation. To further improve the accuracy of SLAM positioning in real time, in this study, we combine the advantages of the RGB-depth map (RGB-D) and light detection and ranging (LiDAR) and propose a model of a two-stage deep fusion framework named convolutional neural network (CNN)–LiDAR vision inertial measurement unit (CNN–LVI) for real-time pose estimation by a geometric method. Unlike existing methods that use either a two-stage framework or multistage pipelines, the proposed framework fuses image and raw 3D point cloud data after multisensor joint calibration, and then uses 3D point clouds as spatial anchors to predict the pose between two sequence frames. By using a CNN algorithm to identify and extract a 3D bounding box, the target object projection of an RGB image is tracked to obtain the target minimum bounding rectangle (MBR). Finally, the rotation angle and translation distance are calculated by a geometric method using the centroid of the target MBR, so as to combine an *inertial measurement unit* to perform joint optimization, achieve the pose estimation of a robot, and further improve the model's location accuracy. Experiments show that the proposed model achieves significant performance improvement compared with many other methods in the car class and achieves the best trade-off between state-of-the-art performance and accuracy on the benchmark with the KITTI dataset.

*Corresponding author: e-mail: yanqd@bit.edu.cn
https://doi.org/10.18494/SAM4225

ISSN 0914-4935
https://myukk.org/

1. Introduction

Simultaneous localization and mapping (SLAM) provides environment map data for the autonomous navigation of robots, making it possible for robots to perceive the environment using sensors and perform autonomous movement.[1] Although light detection and ranging (LiDAR) technology is relatively mature and can obtain good depth information, the vertical resolution of LiDAR is still low, only a sparse point cloud can be obtained, and the features provided are limited. However, LiDAR also has strong robustness to environmental changes and is widely used in various indoor navigation scenes. Even so, when a robot is placed in a dynamic environment with high peripheral similarity, the ability of LiDAR to perceive the environment and identify objects declines, which may prevent the robot from walking normally and cause robot navigation to fail. In contrast, with vision, LiDAR can provide robust and accurate depth information regardless of lighting and texture conditions. This is directly related to LiDAR's stable and reliable 3D scene data acquisition technology, which makes it the preferred device for SLAM. Therefore, many mature laser SLAM systems such as LOAM[2] and Lego LOAM[3] can obtain intensive and accurate maps and show stable performance despite environmental changes. At present, LiDAR with high spatial resolution is expensive, whereas low-cost LiDAR often has a low spatial resolution and high noise, making it difficult to find stable features, because the information content is less rich than that of images. Therefore, it is difficult to extract advanced semantic features using LiDAR, which makes it difficult to relocate to an accurate location when tracking is lost. In addition, LiDAR lacks loop detection ability, and it is difficult to eliminate accumulated error. The reason is that vision has strong scene recognition ability, and it contains a massive amount of texture information, and thus, the visual SLAM is easily affected when light or texture is lost. In particular, rapid motion will cause image blur, resulting in tracking failure. In terms of image processing, map construction based on nonlinear optimization is very complex and time-consuming. Even so, although the system framework of laser SLAM is mature, many engineering application problems remain unsolved. Many scholars have proposed the use of multisensor fusion or even deep learning to improve the positioning accuracy of laser SLAM.[4] So far, laser SLAM based on graph optimization is the mainstream technology. In view of the above reasons, we propose a SLAM solution fusing a multisensor, which combines different detection results at the post-fusion phase. The proposed solution is simpler and more effective than the previous fusion and does not require additional transformation of the original data. At present, achieving accurate and real-time 3D object detection is still a great challenge. Most existing works such as the conversion of the 3D point cloud to images by projection[4,5] or to volumetric grids by quantization[5–7] and the application of some neural networks[8] have been presented. In fact, from the perspective of computing performance, many scholars have proposed to process point clouds directly without converting them to other formats,[9] such as PointNets,[10] which has resulted in better performance and efficiency in the understanding of several 3D tasks, including object classification and semantic segmentation. Because of the computational complexity in 3D, we take advantage of mature 3D object detectors to extract the bounding target box, and project onto the 2D images. In contrast to previous works that address LiDAR data using convolutional neural networks (CNNs), our objective is to obtain a bounding box

regardless of the 3D projection or 2D extraction. Our method achieves the best trade-off between performance and accuracy on the KITTI benchmark. The contributions of this work can be summarized as below:

(1) A CNN–LiDAR vision inertial measurement unit (CNNL–LVI) framework with the graph optimization method was proposed, which is a two-stage deep fusion model to solve the problem of real-time pose estimation.
(2) A target minimum bounding rectangle (MBR) calculation by using a geometric method is presented after fusing 3D point clouds and RGB-depth (RGB-D) images, using a Canny operator to obtain a 2D bounding box by tracking the target object projection, so as to calculate the robot's position and pose.
(3) Our method for pose estimation was verified using the results of experiments in real scenarios to show that the method is feasible and that the best trade-off between the state-of-the-art performance and accuracy is achieved.

2. Related Works

Since the stochastic solution to probabilistic SLAM was first proposed,[11] scholars have proposed Bayesian filtering, graph optimization, and Gauss Newton optimization methods to build a global 2D laser SLAM. After that, the extended Kalman filter (EKF) was used to solve the SLAM problem. Owing to the use of probability statistics, there exist large computational resources, poor algorithm stability, and the established feature map, which cannot be directly used for the autonomous navigation of robots. Later, to decompose the problem of a robot's pose estimation with respect to SLAM, the Monte Carlo method, which contains a particle filter and a Kalman filter, was used[12] and improved versions of fast SLAM, gmapping[13,14] and CoreSLAM,[15] were proposed. However, the SLAM algorithm based on the Kalman filter or particle filter cannot achieve ideal results when constructing a large-scale map because of the absence of loop detection. In the process of achieving accurate positioning and real-time map construction, methods of graph optimization[16,17] have been presented one after another. Combined with the characteristics of radar, vision, and other sensors, starting from the aspect of efficiency and accuracy, researchers have proposed various solutions to the problem of SLAM. More attention is being paid to the fusion of LiDAR point clouds and images. Li *et al.*[18] proposed the fusion of 2D LiDAR and a depth camera so as to improve the antijamming ability, detection range, and mapping accuracy of the sensor. However, achieving accurate and real-time 3D object detection is the precondition for successful autonomous navigation and stable walking of robots. Hence, in many research studies, such as those on MV3D[19] and AVOD,[20] the two-stage framework was adopted to detect 3D objects based on point cloud and RGB images to improve the efficiency and accuracy of SLAM. Compared with the time-consuming two-stage 3D detection model, some groups[21,22] utilized the one-stage framework to detect 3D objects. However, the one-stage framework yields a worse accuracy than the two-stage methods. After analysis, if we take raw point cloud and RGB images as inputs, convert point clouds into 2D bird's-eye-view (BEV) images, and fuse the 2D images from the camera to improve the accuracy, the computational complexity will be slightly reduced.

In this study, we build a complementary system scheme based on LiDAR, supplemented by a multisensor such as depth camera RGB-D and IMU. Our system can improve the acquisition of front-end data. Meanwhile, we let the system carry out tight coupling depth fusion at the back end and constantly correct the robot's pose. Finally, the accumulated error is eliminated by loop detection, so as to realize the first environmental exploration by a robot walking steadily and realize accurate positioning for rapid SLAM construction.

3. Framework of CNN-LVI upon Graphic Optimization Method

For the purpose of fusing the multisensor data to improve the computational efficiency and real-time robot free walking, we propose a framework for tightly coupled LiDAR-Visual-Inertial odometry upon graphic optimization method based on LVI-SLAM,[23] which is described in detail in Fig. 1. We can see that the camera and LiDAR contain three parallel processes. Even if one of the parts fails, the system can still work effectively. We can further improve the system accuracy and robustness by combining the camera and LiDAR and by using the method for pose estimation with joint optimization. To guarantee a real-time response and decrease the volume of feature point cloud processing, an algorithm for CNN-based edge detection for targets was applied. After data fusion, we track the feature points in the front and back frames to estimate the camera pose. Then, we combine the IMU pose estimation and carry out the loop closure detection to perform joint optimization and further improve the accuracy of pose estimation. Therefore, to be exact, our framework is a graph-optimized tightly coupled SLAM system with a multisensor.

3.1 Multisensor joint calibration

Usually, before fusing LiDAR and camera data, the coordinate reference system must be initialized for each sensor. However, we need to build a multiparty coordinated robot system with multiple sensors. The relationship between the camera and the robot is not only a problem of coordinate transformation, but also a problem of intrinsic parameter calibration.

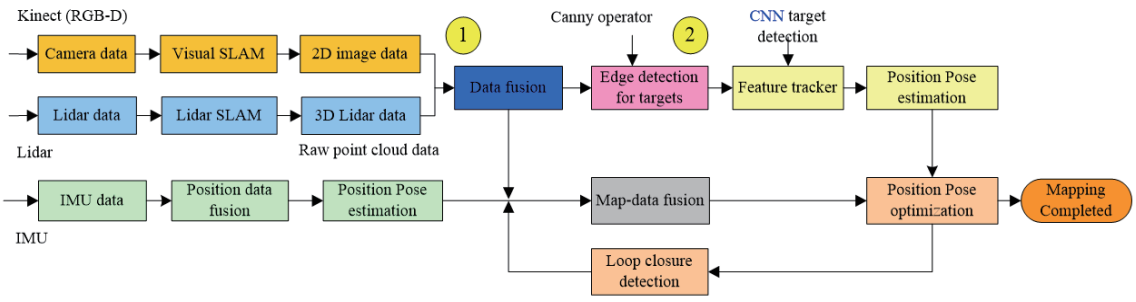


Fig. 1. (Color online) Framework of CNN-LVI upon graphic optimization method.

Because LiDAR and the camera use two different coordinate reference systems, it is necessary to define a world coordinate system to realize the synchronization of sensor data. To simplify the calculation, we assume that the reference frame of the mobile robot is the same as that of the LiDAR. A flow chart of coordinate transformation is shown in Fig. 2.

LiDAR and the camera detect the ground object in different data forms. As shown in Fig. 3, point P represents the particle of the ground object. Under the coordinate system (O_L, X_L, Y_L, Z_L), the laser radar coordinate is (x_L, y_L, z_L), under the coordinate system (O_C, X_C, Y_C, Z_C), the Kinect camera coordinate is (x_c, y_c, z_c), and the projection coordinate of point P in the image plane coordinate system (X_P, O_P, Y_P) is $P'(u, v)$. However, the target data of point P collected by LiDAR is not the coordinates (X_L, Y_L, Z_L), but the distance r and angle α. The object data of point P collected by the Kinect camera is not the coordinates (x_c, y_c, z_c), but the projection coordinates (u, v) and the corresponding depth information (Z_c). The detailed relationships of image coordinates between LiDAR and the camera are shown in Fig. 3.

From the geometric relationship between the LiDAR coordinate system (O_L, X_L, Y_L, Z_L) and the Kinect camera coordinate system (O_C, X_C, Y_C, Z_C), we can obtain the transformation relationship of the ground object P in different coordinate systems.

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} R & T \\ O^T & 1 \end{bmatrix} \begin{bmatrix} x_L \\ y_L \\ z_L \end{bmatrix}. \tag{1}$$

Here, R is the rotation array, and T is the flat motion array. In view of the classic model of the camera, RGB-D can collect the coordinate data $P'(u, v)$ and depth information z using the Kinect camera, where P' is the projection of the ground object P on the image plane. Here, the relationship between the coordinate system of the Kinect camera and the data u, v, z of the LiDAR is shown as

$$z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}. \tag{2}$$

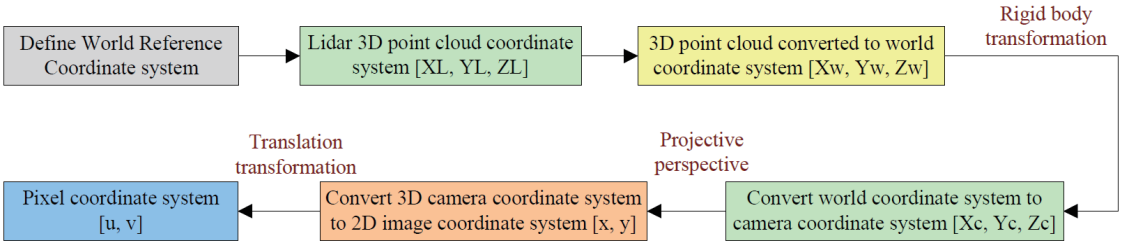


Fig. 2. (Color online) Flow chart of coordinate transformation.

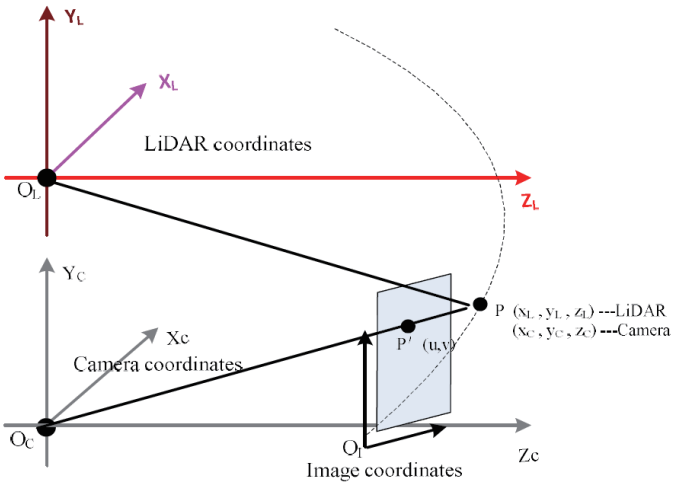


Fig. 3. (Color online) LiDAR and camera coordinate systems.

In Eq. (2), f_x and f_y are the equivalent focal lengths in pixels on the x- and y-axes, respectively. c_x and c_y are the reference points on the x- and y-axes, respectively. All of them belong to the intrinsic parameter of the camera. According to Eqs. (1) and (2), we can convert the ground object information collected by the Kinect camera into the coordinate under the LiDAR coordinate system. Here, we need to maintain the coordinate origin on the y-axis between LiDAR and the camera and let the vertical height difference be h; then, we have

$$z_L = r\cos a,\ x_L = r\sin a,$$
$$z_c\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} R & T \\ O^T & 1 \end{bmatrix}\begin{bmatrix} r\sin a \\ h \\ r\cos a \end{bmatrix}. \quad (3)$$

After determining the camera parameters of f_x, f_y, c_x, and c_y, we take multiple pairs of Kinect data u, v, and z and LiDAR data r and a and apply them to Eq. (3) by solving the linear equations to obtain the matrixes R and T. Then, we determine the rotation and translation relationship between the Kinect camera and the LiDAR coordinate system to complete the joint calibration.

3.2 RGB and LiDAR data fusion

LiDAR can be used to measure the shape and contour of objects. Therefore, to track a target and obtain accurate pose estimation, we first fuse the image and point cloud data, obtain the target polygon through a 2D image edge detection algorithm, and then select several groups of feature points in the polygon to calculate the pose. This can considerably reduce the amount of calculation for feature point data and improve the computational efficiency.

For LiDAR point cloud data, how the data are fused in different spatial dimensions is very important. Considering that data fusion contains the data in the early stage and the results in the

later stage, the main idea in the early stage is to project the 3D LiDAR point clouds onto the 2D plane[24] and to check whether the point clouds belong to the 2D bounding box. On the other hand, postfusion refers to fusing the results of independent detection, including projecting the 2D bounding box of the image onto the 3D bounding box, then fusing these bounding boxes during the LiDAR detection process.

According to the literature,[25,26] the 3D radar point cloud and RGB image are used as input data. In fact, BEV and RGB-D images can be obtained easily. After multisensor joint calibration and coordinate transformation, the height information of a 3D radar point cloud will be projected onto the RGB image plane with its depth information. Therefore, for BEV, the image plane is sliced on the basis of the point cloud, and then the attributes of pixel points (x, y, z) are identified by calculating the density features and height features. For the RGB-D image, only the height information of point cloud projection needs to be embedded into the original RGB image. The whole process is divided into three steps.

First, the point cloud (X, Y, Z) is mapped onto the original image ($W \times H$) plane as

$$\begin{aligned} &\begin{pmatrix} u & v & 1 \end{pmatrix}^T = M \begin{pmatrix} X & Y & Z & 1 \end{pmatrix}^T, \\ &M = P_{roj} \begin{pmatrix} R_{velo}^{cam} & t_{velo}^{cam} \\ 0 & 1 \end{pmatrix}. \end{aligned} \tag{4}$$

Here, (u, v) is the image coordinate, P_{roj} is a project matrix, R_{velo}^{cam} is the rotation matrix from LiDAR to the camera, t_{velo}^{cam} is a translation vector, and $\boldsymbol{M}$ is the homogeneous transformation matrix from LiDAR to the camera.

Second, the points $\{(x, y, z) \mid x \in X, y \in Y, z \in Z\}$ located in the $W \times H$ image size are kept. Meanwhile, the LiDAR points are projected to the camera coordinates and denoted as (x_c, y_c, z_c).

$$\begin{pmatrix} x_c & y_c & z_c \end{pmatrix}^T = \boldsymbol{M} \cdot \begin{pmatrix} x & y & z & 1 \end{pmatrix}^T \tag{5}$$

Finally, z_c is mapped between 0 and 255 and then assigned to the corresponding image coordinate (u, v).

3.3 MBR generation

Generally, the robot's pose estimation involved registering the point clouds of adjacent frames to obtain the relative transformation relationship of the target, so as to achieve positioning in the environment. One method is to achieve positioning through 2D target detection by RGB-D image, and then use its corresponding point cloud data to track features on the basis of the detection results.[27] The other method is to use the PointNet 3D target edge detection method to complete target tracking to achieve the mobile robot's pose estimation. The former requires a large amount of calculation. Therefore, to reduce the computational cost and improve efficiency, we plan to use the second method to implement the edge detection of the object in the fused image, obtain the MBR, and further estimate the position and the robot's pose

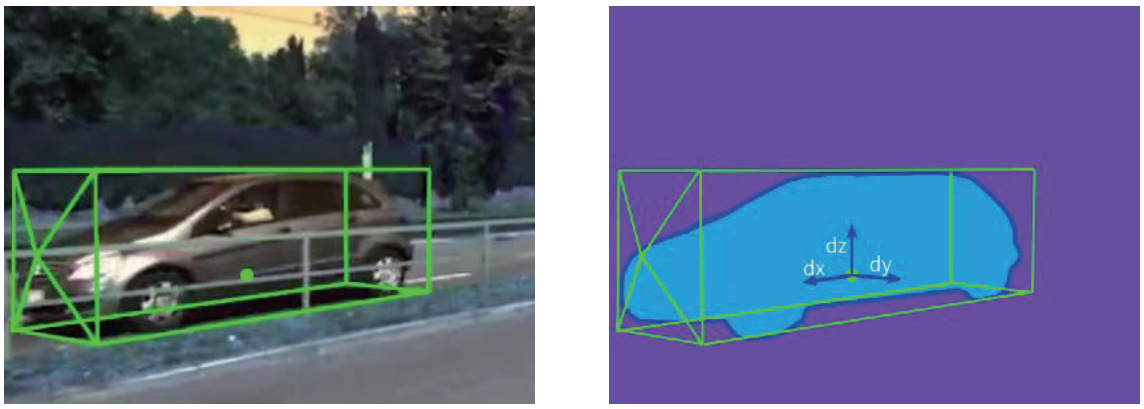
by calculating the coordinates of the centroid of the ground object in the previous and last two frames of images.

The edge detection of objects in 2D images is an important base in the field of image segmentation, object recognition, and region extraction. Because image noise is sensitive to the noise of edge detection operators, the current edge detection algorithm incorporates improved templates such as a neural network[28] and heuristic algorithm,[29] requiring a large number of training samples to establish models. In fact, the cost of the above algorithms is high and the efficiency is low. In comparison, the Canny operator[30] is the most principled edge detection operator, and it has become the standard for evaluating other edge detection methods. Therefore, in this research, we will use an improved Canny operator with a stronger search ability to achieve target detection for fusing the image and point cloud data, and ultimately, to build the MBR. Figure 4 shows the detection result of the bounding box obtained using CNN and the MBR obtained using the Canny operator for the vehicle.

To solve the problem of the target box often being larger than the real target object, image segmentation could be made more accurately by matching the projection points and pixels. In this study, we adopt the former method of fusing tracks, which requires intersection of union (IoU) matching metrics in the time domain. When tracking the position of the 3D bounding box, IoU is generally used as a metric for data association. Of course, the depth convolution feature can also be used to further ensure consistency for the targets.

3.4 Posture calculation and optimization

The least square optimization function of the pose estimation can be constructed between multiple groups of barycenter points matched with the target edges of adjacent frames in the world coordinate system. The optimization method is used to iteratively solve the optimal solution of the least square optimization function, and the optimal solution is taken as the final state quantity of the pose estimation. Moreover, if the pose estimation returns to the previous



(a) (b)

Fig. 4. (Color online) Detected object in green bounding box and MBR for the vehicle. (a) Green bounding box for the vehicle. (b) MBR for the vehicle.

position within the current period, the frame data of the RGB-D camera at the current time and the previous position are used as constraints to globally optimize the pose estimation state variables within the current period; then globally consistent pose estimation state variables will be obtained.

3.4.1 Posture calculation

The calculation of the robot's pose requires its rotation angle and translation distance in each axis. As shown in Fig. 5, we take the direction of the vehicle as the x-axis, which is the maximum length of the MBR, and the barycenter point C indicates the initial position of the vehicle. When a vehicle travels from the initial location C to C′ in the direction from x to x', the x-axis rotates θ and translates a distance d to x'. If we know the coordinate in advance, the values of the corner coordinate and θ can be easily calculated.

3.4.2 Posture optimization

Perspective-n-point (PnP) is the problem of pose estimation with a calibrated camera given a set of n 3D points in the world and their corresponding 2D projections in the image.[31] At present, there are many methods to solve PnP problems, including direct linear transformation (DLT), P3P, EPnP, UPnP, and nonlinear optimization methods. However, in SLAM, P3P or EPnP[32] and other methods are often used as the first step in estimating the robot's pose, then to build the least squares optimization problem and further complete the bundle adjustment (BA). Therefore, we can construct the PnP problem as a nonlinear least squares problem defined on Lie algebra to solve the optimal solution of camera pose. The residual error (predicted value − observed value) or reprojection error is now defined as

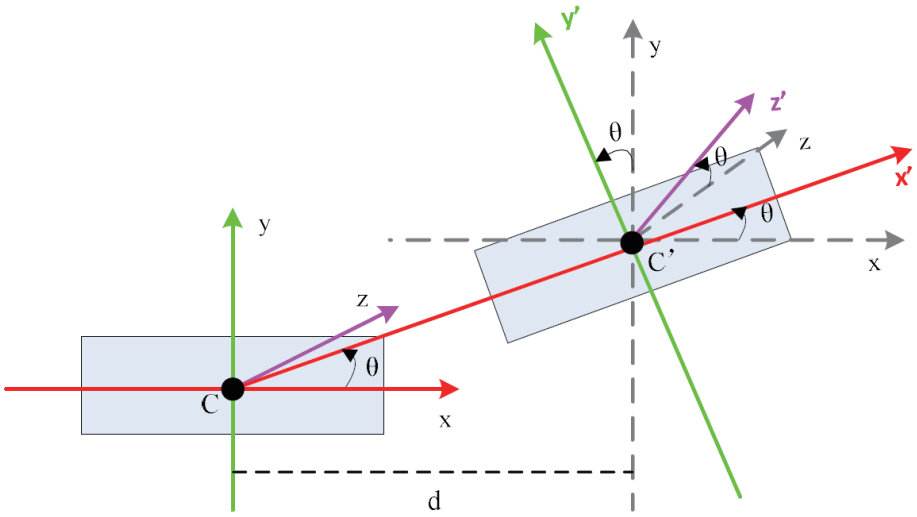


Fig. 5. (Color online) Vehicle travel direction for posture calculation.

$$r(\delta) = K \exp\left(\hat{\delta}\right) P - u. \tag{6}$$

The least squares problem is shown by Eq. (7). Usually we solve the least squares problem using the Jacobi function as

$$\delta^* = \arg\min_{\delta} \frac{1}{2} \sum_{i=1}^{n} \left\| r(\delta) \right\|_2^2. \tag{7}$$

The error function determines the direction of the next optimal iterative estimation of the position and attitude increment for the Jacobian matrix of the position and attitude. According to the above process of pose transformation, we can use the following chain rule of Eq. (8) to express *J*.

$$\begin{aligned} J(\xi) &= \frac{\partial r(\xi)}{\partial \xi} \\ &= \frac{\partial r(\xi)}{\partial P'} \cdot \frac{\partial P'}{\partial P'_{norm}} \cdot \frac{\partial P'_{norm}}{\partial P} \cdot \frac{\partial P'}{\partial \xi} \\ &= J_0 \cdot J_1 \cdot J_2 \cdot J_3 \end{aligned} \tag{8}$$

From the above calculation, it can be seen that the Jacobian matrixes with the direct method and the characteristic point method are only different in the parameter J_0. For the specific derivation steps, see the error function for the Jacobian matrix of the pose when SLAM is optimized, which is ignored here. The result is shown as

$$\begin{aligned} J &= \frac{\partial r(\xi)}{\partial \xi} \\ &= \begin{bmatrix} \frac{£_x}{Z'} & 0 & -\frac{X'£_x}{Z'^2} & -\frac{X'Y'£_x}{Z'^2} & f_x + \frac{X'^2£_x}{Z'^2} & -\frac{Y'£_x}{Z'} \\ 0 & \frac{£_y}{Z'} & -\frac{Y'£_y}{Z'^2} & -f_y - \frac{Y'^2£_y}{Z'^2} & \frac{X'Y'£_y}{Z'^2} & \frac{X'£_y}{Z'} \end{bmatrix} \in R^{2\times 6}. \end{aligned} \tag{9}$$

4. Evaluation Metrics for IoU

The output result of the target detection model is unstructured, and the parameters such as number, location, and size cannot be known in advance, so the evaluation algorithm of target detection is slightly more complex. For a specified target, we can measure the quality of detection from the degree of overlap between the prediction box and the ground truth box. Generally, IoU is used to quantify the degree of overlap.[33] In target detection, IoU mainly refers to the difference between the bounding box and the ground truth predicted using the model.

As shown in Fig. 6, the corresponding 3D point cloud of an image bounding box is obtained by finding all points in the scene that can be projected onto the box. However, the spatial location

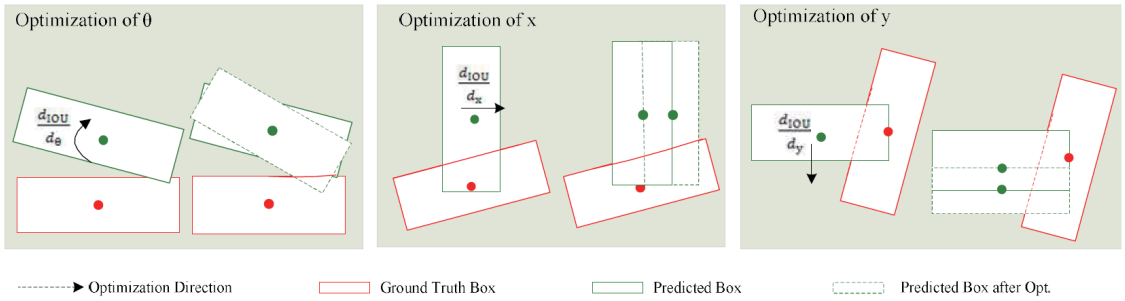


Fig. 6. (Color online) 2D IoU metric between ground truth box (red) and predicted box (green) in BEV with central point.

of the 3D points is highly correlated with the 2D box location. We crop the 2D box as a region of interest (ROI) to calculate the IoU, which describes the degree of coincidence between the adjacent frames for correcting the edge of the object to ensure that the two objectives are the same. We use the IoU to evaluate the overlap region of a detected polygon (s_{dp}) and a ground truth polygon (s_{gp}) divided by the union area (s_{dp}) and (s_{gp}). The metric of IoU is shown as

$$IoU = \frac{area\left(s_{dp} \cap s_{gp}\right)}{area\left(s_{dp} \cup s_{gp}\right)}. \tag{10}$$

When the IoU between the detected polygon and a ground truth is larger than 0.5, we can regard the polygon to be the same as the object. In fact, the matching principle of IoU in the time domain is similar to that in the space domain. If the bounding boxes of the target objects in the first and second frames coincide, it means that the two targets are the same.[34] Once the object is ensured, the centroid of the polygon can be calculated for the objects. Using the centroid coordinates, we can calculate the values of rotation and transformation for the objects to realize the preliminary step of pose estimation. The data values can be further optimized by solving multiple groups of target objects.

To further verify the accuracy of the predicted value of the tracking target in the rotation and translation directions, the MBR obtained by dividing the 2D bounding box and RGB image projected by the 3D box on the plane is optimized by calculating the IoU value. Compared with 3D box prediction, the algorithm performance is significantly improved because the computation of the intersection area between two rotated 3D bounding boxes is much more complex than that of their 2D counterparts without rotation. By referring to the former two-stage methods, we utilize IoU guided supervision to generate more reliable confidence predictions in an end-to-end pipeline to directly predict pixel-level object categories and bounding boxes.

5. Experiments and Discussion

To validate the performance of the proposed model, several experiments were performed in both outdoor and indoor environments. The sensor consists of a Velodyne VLP-64 LiDAR and a

monochrome global-shutter Blackfly BFLYPGE-23S6M (RGB-D) camera. The model is trained and evaluated on the KITTI dataset.[35] Each frame is composed of a point cloud, RGB-D images, and calibration data. Considering that the KITTI object dataset possesses a massive frameset of the dataset for training, we extract the character from 2D images by using ResNet and the global and local characters of every point by using PointNet. The KITTI dataset usually includes the three categories of vehicles, pedestrians, and bicycles related to the environment of an autonomous vehicle. For simplicity of the experiments, we herein focus the shape and centroid of the object on the ground truth to estimate the pose for the car category, where our goal is the pose calculation to obtain the parameter rotation R and transformation T. For evaluation, we compare the proposed LIC-Fusion against the state-of-the-art visual-inertial and LiDAR odometry methods, including the proposed implementation of the standard MSCKF-based VIO[36] and the open-sourced implementation of LOAM LiDAR odometry.[2] The experimental results are shown in Table 1.

In all experiments, a fair comparison of the results under the same environment is needed. In the field of multimodal 3D object detection, as the pioneer, and MV3D was employed in other subsequent articles, have received certain effects. In addition, there is no one-stage method that publicly provides results for the pedestrian/cyclist classes for 3D object detection using LiDAR and RGB images. Hence, the comparison is only for the car class. In accordance with the bounding box scale, truncation levels, and occlusion classes of objects, we use the frames in the same scenario to perform the evaluation. Inspired by 3D object detection, KITTI's object detection metric is defined as an 11-point average precision. IoU is the generic evaluation criterion for 2D and 3D detections and is at a threshold of 0.7 for car class detection. Before that, the RGB camera images must be cropped to a uniform size to be used as a training dataset by ResNet or for the raw 3D point cloud by PointNet.[37] Thus, a computer with a TensorFlow framework, which fixed NVIDIA 1080 Ti GPU with a batch size of 1, is needed for our experiments.

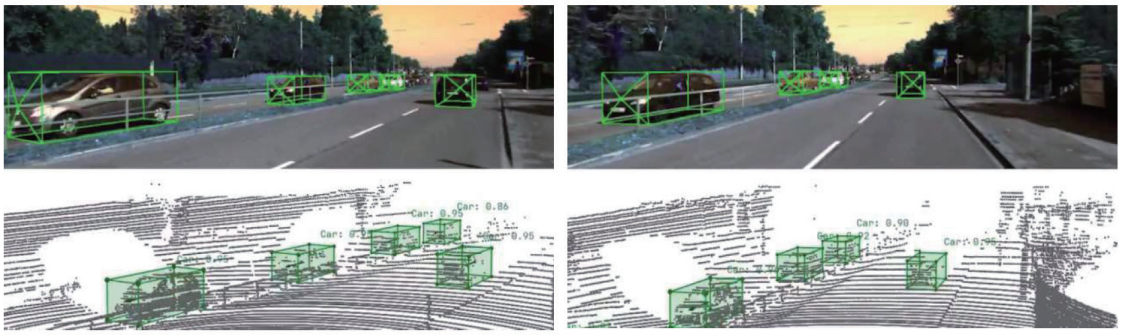
For a fair comparison, we only compare the computational efficiency with the recent state-of-the-art methods that use the LiDAR point cloud and images as input data. In our model, 2D detection is more important than 3D detection. Therefore, we only focus on the comparison of the runtime in the first stage of the detection methods. While the end-to-end one-stage method is much more elegant and effective than the two-stage method, because our model has two stages, the average absolute trajectory error (ATE) is a good metric that can be calculated in the same runtime environment. In comparison, it is imperative to improve the single-stage detector for a better trade-off between performance and accuracy. Table 2 shows that the 2D ATE of mAP is lower than BEV among the methods published in recent years. Although some of the methods use CNN or deep learning to improve their accuracy regardless of the number of stages to be used, we can see that our proposed method achieves better performance in 2D and BEV. We still

Table 1
Experimental results: Average absolute trajectory errors (ATEs) and their standard deviation/variability.

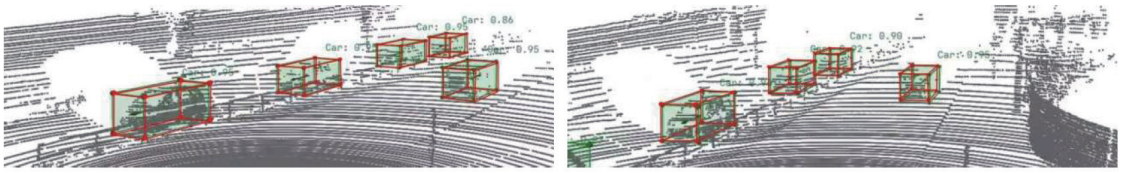
	MSCKF	LIC-Fusion	LOAM	Proposed (ours)
Average ATEs (m)	10.75	4.06	23.08	8.76
1 Sigma (m)	3.56	3.42	2.63	2.75

Table 2
Experimental results: ATE calculation.

Method	Publication year	Stage (s)	Runtime (ms)	2D (%) mAP	BEV (%) mAP
MV3D	2017	Two	360	63.51	80.44
PC-CNN	2018	Two	500	53.59	76.86
AVOD	2018	Two	**83**	74.99	—
MCF3D	2019	Three	160	77.84	84.75
Point-GNN	2020	Two	133	86.73	82.1
LiDAR-RCNN	2021	Two	107	83.50	**89.30**
VoTr-TSD	2021	Two	97	86.30	—
BtcDet	2022	Two	116	**87.10**	—
VoxelNet	2018	Single	123	75.60	88.20
PointPillar	2019	Single	107	81.53	87.10
3DSSD	2020	Single	129	78.42	—
SE-SSD	2021	Single	106	79.05	—
Proposed (ours)	—	Two	**93**	**79.53**	**83.16**



(a)



(b)

Fig. 7. (Color online) 3D detection results of fusing RGB and point clouds for car class between two sequential frames. (a) 3D detection results of fusing RGB and point clouds for cars between two sequential frames. (b) IoU metric of coverage for ground truth and predictions on point clouds.

achieve the best trade-off in terms of performance and accuracy.

Figure 7(a) shows the detection and localization upon fusing RGB and point clouds for cars between the two sequence frames on the KITTI dataset. Figure 7(b) shows the IoU metric of coverage for ground truth and predictions on point cloud data. Experiments show that our proposed method is suitable for detection and prediction.

With our proposed model, the car class of the precision–recall curve is as shown in Fig. 8.

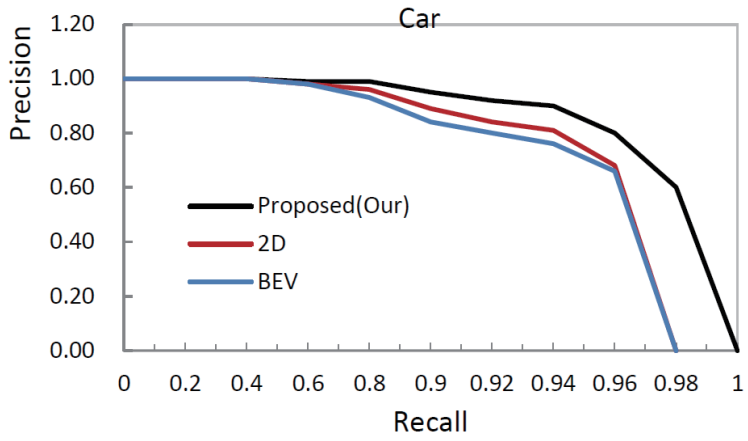


Fig. 8. (Color online) Precision–recall curves for car class obtained by different methods.

The curves of 2D, BEV, and our proposed method denote the relationship between precision and recall of the car class detections. We can see that the proposed method still achieves a comparable speed by taking the precision–recall curve into account.

6. Conclusions

We proposed a two-stage deep fusion framework named CNN-LVI to solve real-time pose estimation using a geometric method, which can be used with the data from sensors of LiDAR point clouds and RGB images. First, on the basis of multisensor joint calibration, we use raw 3D point clouds and images as input data to predict the pose by referencing the CNN algorithm to extract the 3D bounding box and by tracking the projection of the target object to obtain a target MBR. Then, we calculate the pose estimation using the centroid of the target MBR. In addition, we adopt PnP to build the least squares optimization problem to adjust the BA and joint optimization to further improve the accuracy of pose estimation. The proposed methods without any regional proposed pipeline belong to end-to-end training and could ensure real-time response in the reasoning process. Our system is evaluated using the official KITTI benchmark tests for different IoU thresholds and recommended best operation point to achieve the trade-off effects between real-time performance and the best accuracy is determined. Our future work includes combining the 2D detector and the PointFusion network into a single end-to-end 3D detector, as well as extending our model with deep learning to perform simultaneous detection and tracking after fusing point clouds and RGB images.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 42261071), the Key Projects of Guangdong Provincial Department of Education (No. 2022ZDZX3030), the Key Laboratory of Intelligent Lithium Battery Manufacturing Equipment

Enterprises of Guangdong Province (No. 2022B1212020003), and the Science and Technology Project of Huizhou City of Guangdong Province (No. 2016X0423038).

References

1 J. Tsai, C.-C. Chang, Y.-C. Ou, B.-H. Sieh, and Y.-M. Ooi: Appl. Sci. **12** (2022) 7775. https://doi.org/10.3390/app12157775
2 J. Zhang and S. Singh: Auton. Robots **12** (2017) 401.
3 T. Shan and B. Englot: 20 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS) (2018) 4758.
4 H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller: Proc. 2015 ICCV (IEEE, 2015). https://doi.org/10.1109/ICCV.2015.114
5 Q. Charles R., H. Su, M. Nießner, A. Dai, M. Yan, and L.J. Guibas: Proc. 2016 CVPR (IEEE, 2016).
6 Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao: Proc. CVPR (IEEE, 2015) 1912.
7 D. Maturana and S. Scherer: Proc. IEEE/RSJ Int. CIRS (2015).
8 A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka: Proc. CVPR (IEEE, 2017). https://doi.org/10.48550/arXiv.1612.00496
9 Q. Charles R., Y. Li, H. Su, and G. J. Leonidas: arXiv (2017). https://arxiv.org/pdf/1706.02413.pdf
10 Q. Charles R., W. Liu, C. X. Wu, H. Su, and L.J. Guibas: arXiv (2017a). https://arxiv.org/pdf/1711.08488.pdf
11 R. C. Smith and P. Cheeseman: Int. J. Rob. Res. (1986) 56.
12 Y. Liu, S. U. Junfeng, and M. Zhu: Chin. Inf. Control. **42** (2013) 632. https://doi.org/10.3724/SP.J.1219.2013.00632
13 M. Michael, T. Sebastian, D. Koller, and W. Ben: Proc. 18th Int. Joint Conf. Artificial Intelligence ACM Press (2003) 1151–1156.
14 G. Grisetti, C. Stachniss, and W. Burgard: IEEE Trans. Rob. **23** (2007) 34.
15 B. Steux and O. El Hamzaoui: Proc. ICARCV (IEEE, 2010) 1975–1979.
16 K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, and R. Vincent: Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IEEE, 2010) 22–29.
17 L. Carlone and B. Bona: Proc. Robotics: Science and Systems VII (MIT Press, Los Angeles, USA, 2011) 41–48.
18 L. Li, Z. Zhang, L. Han, C. Y. Li, and Y. N. Ding: Chin. J. Intell. Comput. Appl. **10** (2020) 87.
19 X. Chen, H. Ma, J. Wan, B. Li, and T. Xia: Proc. ICVPR (IEEE, 2017) 6526–6534.
20 J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander: Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (2018) 1–8.
21 M. F. Mozifian. Master thesis, Univ. Waterloo: http://hdl.handle.net/10012/13267 (accessed Oct. 2022).
22 M. Li, Y. Hu, N. Zhao, and Q. Qian: IEEE Sens. J. **19** (2019) 1434.
23 T. Shan, B. Englot, C. Ratti, and D. Rus: Proc. ICRA (IEEE, 2021) 5692–5698. https://doi.org/10.48550/arXiv.2104.10831
24 X. Zuo, Y. Yang, J. Lv, Y. Liu, G. Huang, and M. Pollefeys: https://doi.org/10.1109/IROS45743.2020.9340704 (accessed Sept. 2022).
25 L.-H. Wen and, K.-H. Jo: IEEE Trans. Indus. Inf. **17** (2021) 6655. https://doi.org/10.1109/TII.2020.3048719
26 Z. Y. Zhai, Z. X. Pan, Z. T. Gao, and W. L Hu: IEEE Sens. J. **22** (2022) 7473.
27 T. Yin, X. Zhou, and P. Krhenbühl: https://doi.org/10.48550/arXiv.2006.11275 (accessed Oct. 2022).
28 Q. Zou, Z. Zhang, Q. Q. Li, X.B. Qi, Q. Wang, and S. Wang: IEEE Trans on Image Processing. **28** (2019) 1498. https://doi.org/10.1109/TIP.2018.2878966
29 Q. B. Hou, M. M. Cheng, W. Xiao, Ali. Borji, W. Zhuo, and Philip Torr: IEEE Trans. Pattern Anal. Mach. Intell. **41** (2019) 815. https://doi.org/10.1109/TPAMI.2018.2815688
30 S. Chen and D. F. Li: J. Mech. Electr. Eng. **37** (2020) 821.
31 Perspective-n-Point: https://en.wikipedia.org/wiki/Perspective-n-Point (accessed Oct. 2022).
32 V. Lepetit, F. Moreno-Noguer, and P. Fua: Int. J. Comput. Vision. **81** (2009) 155.
33 H. Sh, S.J Cai, N. Zhao, B. Deng, J. Q. Huang, X. S. Hua, M. J. Zhao, and G. H. Lee: IEEE/ECCV **2022** (2022). http://arxiv.org/pdf/2207.09332
34 Y. H. Shen, J. L. Liu, and X. Du: J. Zhejiang University (Science A: An Int. Applied Physics & Engineering J.). **9** (2008) 489.
35 A. Geiger, P. Lenz, and R. Urtasun: Proc. 2012 CVPR (IEEE, 2012) 3354.
36 K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. K. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar: IEEE Rob. Autom. Lett.. **3** (2018) 965. https://doi.org/10.1109/LRA.2018.2793349
37 D. Xu, D. Anguelov, and A. Jain: Proc. 2018 IEEE/CVF Conf. CVPR (IEEE, 2018) 244–253.