ISPRS Journal of Photogrammetry and Remote Sensing 194 (2022) 209–221

Contents lists available at ScienceDirect

ISPRS Journal of Photogrammetry and Remote Sensing

journal homepage: www.elsevier.com/locate/isprsjprs

I2D-Loc: Camera localization via image to LiDAR depth flow

Kuangyi Chen [a], Huai Yu [a,*], Wen Yang [a], Lei Yu [a], Sebastian Scherer [b], Gui-Song Xia [c]

[a] School of Electronic Information, Wuhan University, Wuhan 430072, China
[b] Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
[c] School of Computer Science, Wuhan University, Wuhan 430072, China

ARTICLE INFO

Keywords:
Camera localization
2D–3D registration
Flow estimation
Depth completion
Neural network

ABSTRACT

Accurate camera localization in existing LiDAR maps is promising since it potentially allows exploiting strengths of both LiDAR-based and camera-based methods. However, effective methods that robustly address appearance and modality differences for 2D–3D localization are still missing. To overcome these problems, we propose the I2D-Loc, a scene-agnostic and end-to-end trainable neural network that estimates the 6-DoF pose from an RGB image to an existing LiDAR map with local optimization on an initial pose. Specifically, we first project the LiDAR map to the image plane according to a rough initial pose and utilize a depth completion algorithm to generate a dense depth image. We further design a confidence map to weight the features extracted from the dense depth to get a more reliable depth representation. Then, we propose to utilize a neural network to estimate the correspondence flow between depth and RGB images. Finally, we utilize the BPnP algorithm to estimate the 6-DoF pose, calculating the gradients of pose error and optimizing the front-end network parameters. Moreover, by decoupling the intrinsic camera parameters out of the end-to-end training process, I2D-Loc can be generalized to the images with different intrinsic parameters. Experiments on KITTI, Argoverse, and Lyft5 datasets demonstrate that the I2D-Loc can achieve centimeter localization performance. The source code, dataset, trained models, and demo videos are released at https://levenberg.github.io/I2D-Loc/.

1. Introduction

Accurate localization techniques are critical towards autonomous robots such as self-driving cars and drones. It refers to the process of finding the 6 degree-of-freedom (DoF) poses of robots by comparing online sensor measurements with the reference maps such as 3D point clouds. Currently, although GPS is the most widely used global positioning system, its accuracy and stability are limited due to the lack of direct line-of-sight to the satellites in urban environments. To alleviate these problems, some researchers propose to utilize sensors such as LiDARs, cameras, Inertial Measurement Units (IMUs) to achieve autonomous localization. LiDAR-based localization and mapping methods (Yew and Lee, 2018; Behley and Stachniss, 2018; Deng et al., 2018; Li and Lee, 2019) have achieved good performance because of the accurate range measurements and robustness to the changes of lighting condition and view angle. However, LiDAR sensors are currently expensive and heavy to be mounted on micro drones. An alternative is to use low-cost, lightweight, and commonly available cameras. Recent camera-based localization methods (Balntas et al., 2018; Ding et al., 2019; Kendall and Cipolla, 2017; Brachmann et al., 2017; Brachmann and Rother, 2018, 2019, 2021; Acharya et al., 2019; Liu et al., 2020) are well-developed and achieve promising accuracy. However, visual-based methods lack robustness over challenging conditions such as illumination, weather, and season changes over time. Therefore, if we can establish direct correspondences between visual images and existing LiDAR maps, the vision-based localization challenges will be greatly mitigated since the visual factors cannot influence LiDAR maps. Meanwhile, light-weight cameras can be located in the accurate 3D maps without a LiDAR sensor, thus the strengths of both LiDAR-based and camera-based methods can be maintained.

If we have the 2D–3D feature matching between LiDAR maps and images, the camera poses can be accurately estimated with PnP solver (Lepetit et al., 2009). However, the estimation of 2D–3D feature correspondences is challenging for two aspects: (1) Feature repeatability. Although there exists a large overlap between point clouds and the corresponding image, it is still hard to make sure the extracted image features (e.g., SIFT Lowe, 1999) and the 3D features (e.g., ISS Zhong, 2009) have a large proportion of repetitiveness and high location accuracy. Consequently, poor feature repeatability will result in many

* Corresponding author.
E-mail addresses: chenky721@whu.edu.cn (K. Chen), yuhuai@whu.edu.cn (H. Yu), yangwen@whu.edu.cn (W. Yang), ly.wd@whu.edu.cn (L. Yu), basti@andrew.cmu.edu (S. Scherer), guisong.xia@whu.edu.cn (G.-S. Xia).

https://doi.org/10.1016/j.isprsjprs.2022.10.009
Received 20 March 2022; Received in revised form 8 October 2022; Accepted 13 October 2022
Available online 2 November 2022

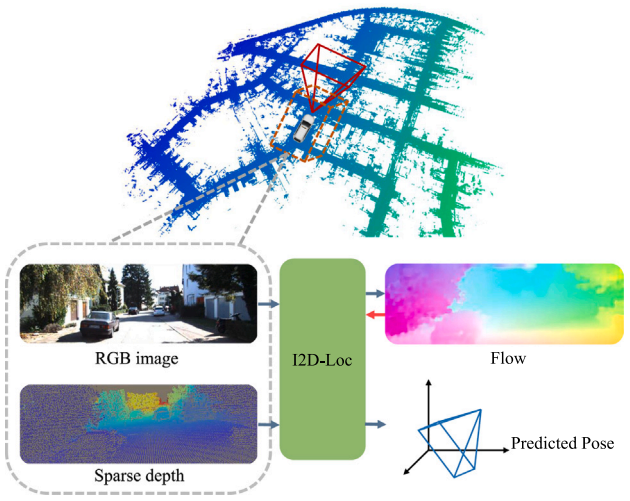


Fig. 1. Pose estimation with I2D-Loc. Given a LiDAR point cloud map, a reference image and a rough initial pose, I2D-Loc can estimate the dense 2D pixel to 3D point correspondences, so as to estimate the accurate camera pose in the LiDAR map.

outlier matches. (2) Description difference. Point clouds are 3D data with geometric information while images are 2D data with texture information. This internal difference makes it challenging to establish a high-dimensional descriptor space for measuring the similarities. To overcome the modality gaps, prior works usually project 3D data into 2D space (Wolcott and Eustice, 2014; Stewart and Newman, 2012) or reconstruct 3D point clouds from multi-view images (Caselitz et al., 2016; Engel et al., 2014) to match in the same domain. In these ways, they alleviate modal gaps and appearance differences to some extent. Compared to matching in 3D space, we argue that matching in 2D space will give more stable dense correspondences and low CPU computation burden since it involves the simple 3D–2D projection, and later cross-modal correspondences can be estimated with GPU acceleration.

Recent works (Cattaneo et al., 2019, 2020; Chang et al., 2021) utilize the learning-based flow estimation strategy to obtain dense 2D–3D correspondences and have achieved remarkable performance. The key insight is to extend the general RGB–RGB optical flow network (e.g. PWC-Net Sun et al., 2018) to the cross-modal RGB-depth flow estimation, which can establish dense 2D–3D point correspondences thus maintaining stable and accurate localization results on continuing frames. However, these approaches cannot be generalized to different intrinsic camera parameters without decoupling the correspondence and pose estimation, which makes the ground truth poses cannot directly supervise the training process of the neural network. Inspired by these works, we propose the I2D-Loc: a scene-agnostic and end-to-end trainable neural network that estimates a 6-DoF pose from an RGB image and an existing LiDAR map. The high-level overview is shown in Fig. 1. In our pipeline, we assume the initial camera pose in the LiDAR map is known, which is usually obtained via image retrieval, GPS, or the pose from the previous frame. With a rough pose, the proposed I2D-Loc can achieve 2D–3D pose tracking without accumulated drift problem. Specifically, we first project the 3D LiDAR map to obtain a sparse depth according to the initial pose and occlusion removal. Considering the sparsity of LiDAR point clouds, we design a simple but effective depth representation that uses a depth completion algorithm to obtain dense depth and a confidence map to weight the features extracted from the depth. Then we develop the cross-modal flow network to predict the correspondences between RGB images and LiDAR depth images. Finally, we connect the flow map with the BPnP module (Chen et al., 2020) to estimate the camera pose in an end-to-end manner with robustness to the changes of intrinsic camera parameters.

The main contributions of this paper are listed as follow:

- We propose a simple but effective network for camera localization based on the local image-LiDAR depth registration with a pose initialization, which predicts the flow between an image and LiDAR depth to obtain 2D–3D correspondences. A BPnP module is used for calculating the gradients of the back-end pose estimation process, enabling our model to be trained end-to-end.
- We design a novel LiDAR depth representation that uses a simple depth completion algorithm to acquire dense depth and utilizes a confidence map to weight the features extracted from the depth to enhance the depth features. It is a novel data enhancement strategy to alleviate the appearance difference between the RGB and depth images.
- To evaluate the performance, we conduct experiments on several public datasets such as KITTI, Argoverse, and Lyft5, and the experiments demonstrate that I2D-Loc can achieve centimeter camera localization without pose drift. More importantly, the experimental results on mixed datasets demonstrate the generalization of I2D-Loc on new cameras.

2. Related work

To establish 2D–3D feature correspondences for camera localization, the common practice is to use intermediary steps of transferring 2D to 3D or projecting 3D to 2D, then the matching can be estimated in the same domain (Sattler et al., 2016; Caselitz et al., 2016). Besides, direct cross-domain 2D–3D localization is more and more popular with the development of cross-modal feature learning (Feng et al., 2019; Li and Lee, 2021; Shi et al., 2022). In this section, we will mainly review the domain-transfer based and direct cross-domain 2D–3D localization methods. Since we mainly focus on the cross-domain 2D–3D localization topic in this paper, the general visual localization and LiDAR localization methods are not detailed.

2.1. Localization with domain transfer

Domain-transfer based localization alleviates the modal gaps between 2D images and 3D point clouds by transferring 2D to 3D or projecting 3D to 2D. We will review the related domain transfer strategies and the matching techniques in this section.

The first kind of domain-transfer based camera localization in LiDAR maps is matching geometry in 3D space after reconstructing 3D point clouds from 2D images by multiple view geometry. Caselitz (Caselitz et al., 2016) utilized Structure-from-Motion (SfM) to reconstruct sparse point clouds from video sequences and then match them to the existing LiDAR maps, which solves the scale estimation problem of the monocular visual odometry (VO) system and achieves online estimation of 6-DoF camera poses. To directly reconstruct true scale point clouds, stereo camera VO or VIO (visual-inertial odometry) (Kim et al., 2018; Zuo et al., 2019) are used to directly match the reconstructed stereo point clouds to LiDAR maps. Then the matching results are loosely or tightly coupled into the VO and VIO system to optimize camera poses. However, SfM-based localization methods highly depend on the accuracy of 3D reconstruction. When encountering dynamic objects, the reconstruction tends to have defects, resulting in a drop of localization accuracy. Besides, monocular SfM suffers from scale problems and the reconstructed sparse points may not have correspondence in LiDAR maps. Additionally, stereo reconstruction gives dense local point clouds but mostly is time-consuming and does not scale well for long-range depth estimation.

The second kind of domain-transfer-based camera localization is matching in 2D image space. Most visual simultaneous localization and mapping (SLAM) methods (Mur-Artal and Tardós, 2017; Qin et al., 2018) reconstruct sparse point clouds using visual descriptors and then match these descriptors to conduct the loop closure for maintaining pose drifts. Some image-based localization methods (Sattler et al.,

2016; Feng et al., 2015) reconstruct 3D point cloud maps using visual SfM, and match these features with the query image descriptors to acquire 2D–3D correspondences for camera localization. However, appearance and visual features are sensitive to changes in illumination, weather and seasons (Campbell et al., 2018), which makes the correspondences unstable for long-term camera localization. Additionally, attaching appearance descriptors to a large-scale LiDAR map has a huge workload and difficulties (multi-view issues) in engineering practice. In contrast, getting a pure LiDAR map without appearance is relatively easy for running a LiDAR SLAM (Zhang and Singh, 2014). Therefore, we focus on the camera localization in the LiDAR map without using appearance descriptors. To handle this characteristic, some methods generate LiDAR appearance synthetic images by projecting LiDAR reflection intensity information to 2D image space by the camera projection model and then use normalized mutual information (NMI) (Wolcott and Eustice, 2014) or normalized information distance (NID) (Stewart and Newman, 2012) to match these synthetic images with the camera images. Recently, CMRNet (Cattaneo et al., 2019) is proposed to firstly generate synthetic depth images by projecting LiDAR point clouds to 2D space with a rough initial pose, and then leverage a trainable optical flow network PWC-Net (Sun et al., 2018) to regress the camera poses. To make this pipeline robust to different intrinsic camera parameters, CMRNet++ (Cattaneo et al., 2020) uses the pixel deformation maps from rough poses to the GT poses to supervise the flow training. With the predicted deformation map, the camera poses can be estimated by RANSAC and PnP solver. These two methods transfer the 2D–3D matching problem to be a visual-depth flow estimation task, enabling the CNN network to focus on correspondence estimation. Following the same scheme, HyperMap (Chang et al., 2021) uses offline 3D map feature computation instead of the online depth map feature extraction to faster the map projection and feature extraction, which greatly reduces map size while maintaining comparable accuracy. However, these methods ignore the sparsity characteristic of the LiDAR projected depth images, which causes challenges for the dense optical flow network module and reduces the accuracy of 2D–3D matching estimation. Besides, these methods cannot achieve end-to-end training as well as remove the influence of intrinsic camera parameters.

2.2. Direct cross-modal localization

Cross-modal localization is directly matching features between 3D point clouds and 2D camera images to estimate camera poses. Compared with the modal-transfer methods, it does not rely on 3D reconstruction or synthetic image generation. Since the estimation of 2D–3D correspondences and camera poses are regarded as a chicken-and-egg conundrum, the conventional methods usually maximize the 2D–3D correspondences to estimate the camera poses (Campbell et al., 2018). SoftPosit (David et al., 2003) and BlindPnP (Moreno-Noguer et al., 2008) utilize existing 2D and 3D point features to maximize the 2D–3D correspondences and estimate camera poses simultaneously. The Accurate pose can be obtained by iteratively updating the pose initialization and correspondences. GOPAC (Campbell et al., 2018) and GOSMA (Campbell et al., 2019) uses branch-and-bound strategy to globally maximize the number of 2D–3D matching inliers without a pose initial guess. However, these methods have a strong assumption that there exists a large proportion of inliers among the 2D and 3D features. This assumption is not trivial for current 2D and 3D feature extraction methods. In Feng et al. (2019), SIFT (Lowe, 1999) keypoints are extracted from images and ISS (Zhong, 2009) keypoints are detected from point clouds for correspondence estimation. Then cross-modality descriptors are learned by feeding patches around the key points into the neural network. However, experimental results shows that it is still hard to maintain a stable inlier rate of correspondences. Liu et al. (1990), Li et al. (2021) and Yu et al. (2020) use geometric co-occurrence to find 2D–3D line correspondences because 2D images share some geometric consistent features with point clouds. However, these approaches only work well on the scenarios with rich geometric information and require accurate initialization, e.g., from a SLAM system. Li and Lee (2021) avoids the difficult challenge to learn cross-modality feature descriptors for registration but refers to the problem as a classification task. Extensive experiments verify its feasibility.

Among the above domain-transfer based and direct cross-domain 2D–3D localization, the LiDAR depth to RGB flow (Cattaneo et al., 2019, 2020; Chang et al., 2021) can be efficient and robust for localization of video sequences in large scale LiDAR maps. Inspired by this strategy, we develop the I2D-Loc, a scene-agnostic and end-to-end trainable neural network which establishes dense correspondences between LiDAR depth and RGB image. Unlike these methods, we creatively introduce a weighted depth-completion module, which effectively improves the degraded accuracy of RGB-depth flow estimation due to the sparsity of the LiDAR depth map (>80% pixels are invalid). By generating dense depth representation, I2D-Loc alleviates the appearance differences between 2D images and sparse LiDAR projections. More importantly, the end-to-end camera localization pipeline is robust to the changes of intrinsic camera parameters. As a result, the pre-trained I2D-Loc model on KITTI and Argoverse can be generalized to the Lyft5 dataset.

3. Proposed method

Problem formulation: Given an existing LiDAR point cloud map, a camera image with known intrinsic parameters, and a rough initial camera pose from image retrieval, GPS, or pose of the previous frame. We first project the 3D point cloud to a 2D plane to obtain the depth images according to the initial pose. Our purpose is to match the synthetic depth images with the given camera images, then predict the accurate 6-DoF camera pose according to the obtained 2D–3D correspondences.

Overview: We denote an RGB image as $I \in \mathbb{R}^{3\times W\times H}$, where W and H are the image width and height, respectively. A point cloud map is denoted as $P \in \mathbb{R}^{3\times N}$. According to the intrinsic camera matrix $\mathbf{K} \in \mathbb{R}^{3\times 3}$ and the rough initial pose $\mathbf{T}^{init} = \left[\mathbf{R}^{init}, \mathbf{t}^{init}\right]$ ($\mathbf{R}^{init} \in SO(3)$ is the 3×3 rotation matrix, $\mathbf{t} \in \mathbb{R}^{3\times 1}$ is the translation vector), the point cloud P can be projected to 2D plane to obtain the sparse depth imagery $D^s \in \mathbb{R}^{1\times W\times H}$. Then we use a simple depth completion algorithm to get dense depth imagery D^d. Meanwhile, a confidence map layer $M \in \mathbb{R}^{1\times W/2\times H/2}$ is used to weight the features extracted from D^d based on the distance from its coordinate to the nearest valid pixel. To obtain dense correspondences, we develop a network architecture to estimate the dense flow field $\mathbf{F} \in \mathbb{R}^{2\times W\times H}$ from the given RGB image I to the dense depth image D^d. Finally, we design a pose estimation algorithm to directly predict the accurate 6-DoF camera pose $\mathbf{T}^{refine}$. With the LiDAR map projection and depth completion, we can estimate the 2D–3D correspondences and camera poses in an end-to-end manner with the supervision of ground truth pixel deformations and poses. The detailed pipeline is shown in Fig. 2. Using the predicted camera poses with the I2D-Loc, the projected LiDAR depth image overlaps well with the corresponding RGB image.

3.1. Depth representation

3.1.1. Sparse depth generation

To transfer the 2D–3D correspondence estimation in 2D space, we first generate a sparse depth image from point clouds. It mainly consists of two steps: map projection and occlusion removal. The process of projecting a 3D LiDAR map to a 2D space can be described by the perspective projection model. We implement this process based on the pinhole camera model. Its definition is

$$[u, v, 1]^{\top} \triangleq \mathbf{K} \cdot \mathbf{T} \cdot \left[x^{\text{map}}, y^{\text{map}}, z^{\text{map}}, 1\right]^{\top}, \tag{1}$$

where $[x^{map}, y^{map}, z^{map}, 1]$ is the coordinate of a 3D point in the LiDAR map, $[u, v, 1]$ is the coordinate of projection on 2D image plane. $\mathbf{K}$ is the

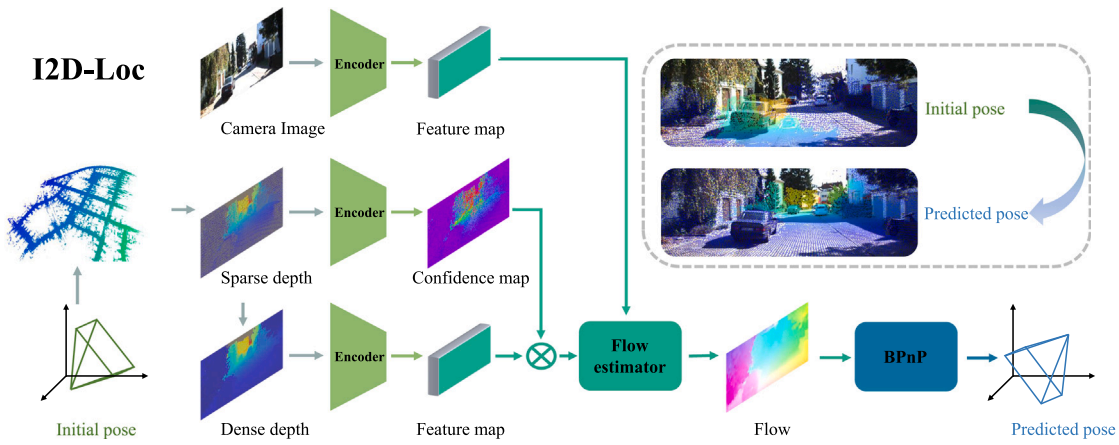


Fig. 2. Pipeline of our proposed I2D-Loc. With an initial pose, the point cloud is projected to a 2D plane to obtain a sparse depth image, which is then densified by a depth completion algorithm. The sparse depth image, dense depth image, and the query RGB image are fed into the model which consists of a flow estimator and a pose estimator called BPnP. In this process, the sparse depth image is used to estimate a confidence map to weight the feature map of the dense depth image. The two images in the upper right represent the projection of the point cloud according to the initial pose and the predicted pose using I2D-Loc, respectively. Given a correct pose, the projection will be perfectly aligned with the RGB image.

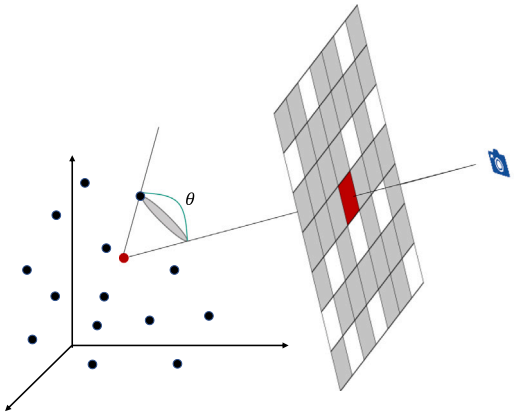


Fig. 3. Visibility. A point with a small aperture is invisible. If the maximal angle θ is bigger than a fixed threshold, the point is marked as visible.

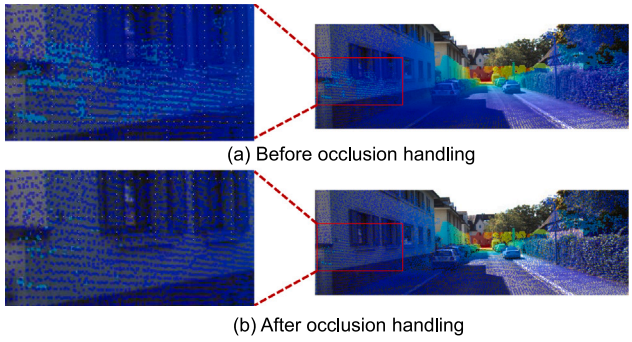


Fig. 4. Visualization of the occlusion handling. Observing the area marked by a red box in (a), we can find some points that should be invisible because of the sparsity of the point cloud. After the occlusion removal, occluded points are filtered out and the sparse depth projection is visualized in (b). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3×3 intrinsic camera parameters. $\mathbf{T}$ is the transformation matrix from the map coordinate system to the camera coordinate system, which is the inverse of camera pose $(\mathbf{R}, \mathbf{t}) \in \mathbf{SE}(3)$ in the map $\mathbf{T} = [\mathbf{R}^\top, -\mathbf{R}^\top \mathbf{t}]$. Thus, every 3D point can be projected to the image plane and only the points in the camera frustum are reserved for depth image generation.

Although the camera frustum constraint can filter all the points out of camera FoV, the generated depth images still have artifacts because of point occlusions. Due to the sparsity of LiDAR maps, occluded 3D points might be visible on the projected depth, as shown in Fig. 3. Therefore, the generated depth image might have inconsistent depth, as shown in the red rectangle area of Fig. 4a. To check the occlusions and remove occluded points, we follow the scheme proposed by Pintus et al. (2011). For each pixel with a valid depth value (non-zero) in the synthetic depth images, we calculate the angles θ between the vector from the 3D point to the camera center point and the vectors from all other valid points in the neighbor of $N \times N$ to itself. If the maximal angle θ_{max} is bigger than a fixed threshold, the point is marked as visible. After occlusion removal, we can obtain consistent depth images. An example is shown in Fig. 4b, there are no ghost depth intensities in the red rectangle.

3.1.2. Depth completion

The depth image generated by direct projection is sparse, and more than 80% pixels do not have valid depth information. As shown in Fig. 5, we use a simple depth completion procedure similar to Ku et al. (2018) as a data enhancement strategy. Considering the generalization and simplicity, we circumvent using neural networks to conduct depth completion. Notably, the motivation is to reduce the appearance difference between depth images and RGB images caused by sparsity rather than propose a better depth completion method. The whole process can be described as five steps.

(1) **Invert.** The principle of our depth completion algorithm is to traverse the whole image using a convolution kernel. The shape of the convolution kernel is either square or circle. During the traversal, the pixel value at the center of the convolution kernel is replaced with the maximum pixel value in the superposition region, which results in larger depth overwriting smaller depth. However, it is universally acknowledged that objects close to the camera will block objects far away. To solve this problem, we invert the pixel values which are non-zero according to $D_{\text{inverted}} = 100.0 - D_{\text{input}}$ and leave the pixel values which are zero unchanged.

(2) **Dilate.** According to the geometric structure of 3D environments, the adjacent pixels of the LiDAR projected depth image have similar depth values in a local area. Specifically, we use a diamond kernel to traverse the sparse depth image to fill the empty pixels with their nearest valid pixels.

(3) **Close.** After dilation operation, there still exist many empty pixels due to the sparsity of the LiDAR maps. To solve this problem,

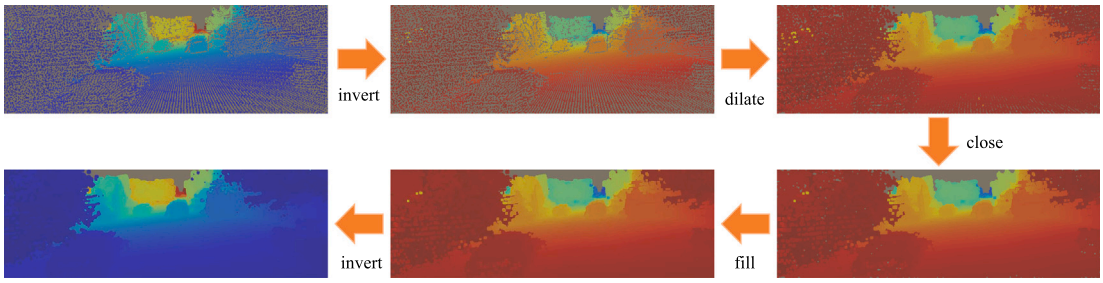


Fig. 5. Depth Completion. Inversion, dilation, close operation, filling, and inversion are conducted in the arrow order. The sparse depth image is getting denser without changing the original geometry.

a morphology close operation is applied. It can fuse pixel blocks that have tiny connections between them via erosion operation after dilation operation. The morphology close operation can fill in small cracks in the foreground object without changing the overall position and shape.

(4) **Fill.** Some medium-sized empty patches still exist after dilation and close. To fill these empty pixels, we first make a mask of all valid pixels, then conduct dilation operation again using a full kernel. Finally, we use the mask to restore the previous non-zero pixels.

(5) **Invert.** Finally, we revert back to the initial depth encoding from inverted depth values handled by previous operations according to $D_{\text{output}} = 100.0 - D_{\text{inverted}}$.

3.2. Localization network

After getting the paired depth and RGB images, we want to estimate the dense pixel correspondences between the two cross-modal images. Since the initial poses used for depth image generation are not accurate, we also need to obtain the true pixel deformation based on the initial poses and ground truth poses.

3.2.1. Flow estimation

To obtain the dense pixel matching between depth and RGB images, we design the image-to-LiDAR depth flow estimation network. Unlike traditional flow estimation networks, it does not utilize a coarse-to-fine pyramid structure to make the estimation process conform to the global smoothing hypothesis. The basic network structure is modified from the RAFT (Teed and Deng, 2020) network. It regards depth-RGB correspondence estimation as an optimization task that estimates the flow at the full resolution and updates it iteratively. Considering the inherent difference between LiDAR depth images and RGB images, the encoders have the same architecture but do not share the weight parameters. They use a modified ResNet (Szegedy et al., 2017) structure to map the depth images and RGB images to feature maps.

Confidence map. As mentioned above, we utilize a morphological algorithm to densify the sparse depth images. However, there exist some differences between the obtained dense depth images and the real depth images. For example, the gaps between the branches are filled with the same depth values as the trunk itself, and the boundary of the objects in the depth images is blurred. To alleviate the problem, a confidence map is utilized to weight the feature maps extracted from the depth images. The confidence maps are generated by mapping the original sparse depth images to 1/2 resolution with a 7×7 convolution kernel. Then the confidence map is used to multiply the feature maps extracted from the dense depth images during encoding.

Ground truth (GT) deformation generation. We obtain ground truth pixel deformation between paired RGB and depth images according to the following formulation:

$$[\Delta u, \Delta v] = \left[u^{gt} - u^{init}, v^{gt} - v^{init}\right], \tag{2}$$

where $[\Delta u, \Delta v]$ is the ground truth pixel deformation, $[u^{gt}, v^{gt}]$ and $[u^{init}, v^{init}]$ are the coordinates of a point projected to 2D plane based on the ground truth pose and the rough initial pose, respectively. Because some points may leave the camera frustum, only the points within the camera frustums are retained.

As shown in Fig. 2, we utilize the confidence map to weight the features extracted from the dense depth. Then the weighted depth features and the features extracted from the camera images are fed into the flow estimator network to predict correspondences between camera images and synthetic depth images, which is supervised by the ground truth deformation map. It should be noticed that GT deformation flows are sparse with valid LiDAR projections. We use this sparse deformation flow to predict the general dense correspondences.

3.2.2. Pose estimation

After estimating the dense correspondences between depth images and RGB images, 2D–3D correspondences between 2D pixel positions in the camera images and 3D points in the LiDAR maps can be inferred according to intrinsic camera parameters $\mathbf{K}$. Then a traditional PnP solver can be used to estimate the accurate camera pose in a RANSAC loop, which usually achieves the highest accuracy due to its robustness to outliers. However, such a formulation does not participate in the deep learning process because hypothesis selection in RANSAC is hard to be differentiable.

In this work, we use a BPnP module (Chen et al., 2020) to estimate the camera pose from the 2D–3D correspondences. It can calculate the gradients of the pose estimation procedure based on the implicit differentiation, and thus optimize the parameters of the front-end network by back propagation.

3.3. Loss function

For the network supervision, we first intuitively consider the front-end RGB-depth flow error and pose estimation error. Then we further account for the local point cloud structural consistency after projection using GT and predicted poses, which means the normal of the projected depth images should be the same. The first one is the average endpoint error(EPE), which is often used as the loss function of optical flow estimation networks (Ilg et al., 2017; Dosovitskiy et al., 2015; Sun et al., 2018; Teed and Deng, 2020). Here we use it to calculates the pixel matching error of images and LiDAR depths. Due to the sparsity of the point cloud, we design it as follows.

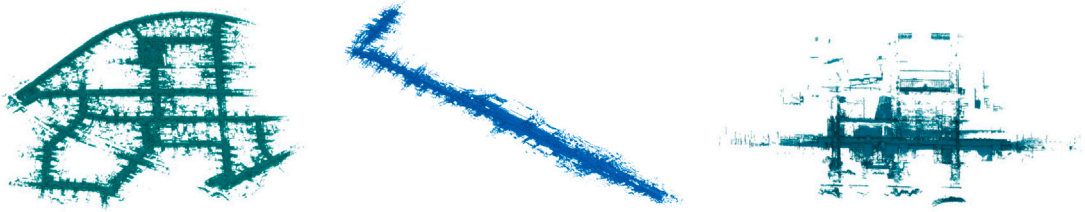
$$\mathcal{L}_{epe} = \frac{\sum \left\| \mathbf{F}_{gt}(u,v) - \mathbf{F}_{pre}(u,v) \right\|_2 \times \text{mask}_{uv}}{\sum \text{mask}_{uv}} \tag{3}$$

$$\text{mask}_{uv} = \begin{cases} 1, \mathbf{F}_{gt}(u,v) != \mathbf{0} \\ 0, otherwise \end{cases} \tag{4}$$

where $\mathbf{F}_{gt}$ represents the ground truth flow, $\mathbf{F}_{pre}$ is the predicted flow, $mask_{uv}$ records which pixels have valid depth values and thus have ground truth flows.

The second part of the loss function is based on the predicted camera poses. When BPnP is used to estimate camera poses, we calculate the reprojection error $\mathcal{L}_{reproj}$ according to the following formulation:

$$\mathcal{L}_{reproj} = \sum_{i}^{N} \left\| p_i - \mathbf{K}\mathbf{T}P_i \right\|_2 \tag{5}$$



(a) KITTI (b) Lyft5 (c) Argoverse

Fig. 6. Visualization of the LiDAR maps.

(a) Sunny (b) Cloudy

(c) Strong light (d) Night

Fig. 7. Visualization of image conditions in KITTI, Argoverse and Lyft5 datasets. The variety of imaging lighting and weather conditions is used to improve and validate the robustness of our I2D-Loc.

where p_i represents the 2D coordinate of a 3D point projection according to the GT camera pose, P_i represents the coordinate of a 3D point in the LiDAR maps, $\mathbf{T}$ is the camera pose predicted by the I2D-Loc.

Moreover, we design a loss function based on surface normal vectors to introduce local geometry constraints to supervise the model learning. Firstly, we obtain the synthetic depth images D^S_{gt} and D^S_{input} according to the ground truth pose and the predicted pose, respectively. Then the surface normal vectors $\mathbf{n}_{gt}$ and $\mathbf{n}_{input}$ are calculated by randomly sampling 3 points from the common points with valid depth values in the D^S_{gt} and D^S_{input}, respectively. We repeat the procedure N times and calculate the L2 distance between the normal vectors $\mathbf{n}^i_{gt}$ and $\mathbf{n}^i_{input}$ during each time. Finally, the loss function is calculated according to the following formulation:

$$\mathcal{L}_{normal} = \frac{1}{N}\sum \left\| \mathbf{n}^i_{gt} - \mathbf{n}^i_{pre} \right\|_2 \tag{6}$$

The final loss function is composed of the weighted sum of the above three loss functions, as

$$\mathcal{L} = \mathcal{L}_{epe} + \alpha \times \mathcal{L}_{reproj} + \beta \times \mathcal{L}_{normal} \tag{7}$$

where α is 10 and β is 100 in our experiments.

In summary, we describe two different back-end pose estimation methods with the estimated flow between RGB and LiDAR depth images: the traditional PnP solver and the BPnP module. The latter focuses more on back-propagating the pose errors to supervise the flow estimation. It can establish end-to-end training and enable the I2D-Loc to predict the camera pose directly with the input query image.

4. Experiments

In this section, we conduct extensive experiments on several datasets to evaluate the performance of our model and compare it with the state-of-the-art methods.

4.1. Datasets and evaluation metrics

4.1.1. Datasets

We conduct experiments on KITTI, Argoverse, and Lyft5 datasets. Portions of the LiDAR maps in these datasets are visualized in Fig. 6. KITTI (Geiger et al., 2013) is one of the datasets commonly used in the field of autonomous driving and is collected by a variety of sensors mounted on a car driving in Karlsruhe, Germany. It consists of 22 sequences, each containing images taken by two color cameras and point cloud data collected by LiDAR. We use the odometry sequences 03, 05, 06, 07, 08, and 09 as the training set and the sequences 00 as the validation set.

Argoverse (Chang et al., 2019) is a dataset published by Argo AI, Carnegie Mellon University, and Georgia Institute of Technology to support 3D tracking and motion detection research of autonomous vehicles. It includes two parts: Argoverse 3D Tracking and Argoverse Perception. We use the sequences train1, train2, train3 from Argoverse Perception as the training set, and the sequences train4 as the validation set.

Lyft5 (Kesten et al., 2019) is released by Lyft, including the Prediction Dataset for detection and the Perception Dataset for Perception. We use the Perception Dataset to assess the generalization ability of our approach. Specifically, we first train our model on KITTI and Argoverse datasets, then evaluate it on the Lyft5 dataset without training for generalization evaluation.

The three datasets are all for autonomous driving but with various scenarios from urban to the countryside, different imaging conditions (illumination, weather, season, etc.), and camera intrinsics. As shown in Fig. 7, weather in KITTI is basically sunny, while the weather in Argoverse includes cloudy, night, and strong light besides sunny. In addition, KITTI and Argoverse datasets are collected primarily in an urban street environment, while the Lyft5 dataset includes some suburban scenarios. The diversity of training and test datasets can guarantee the generalization and robustness of the proposed method.

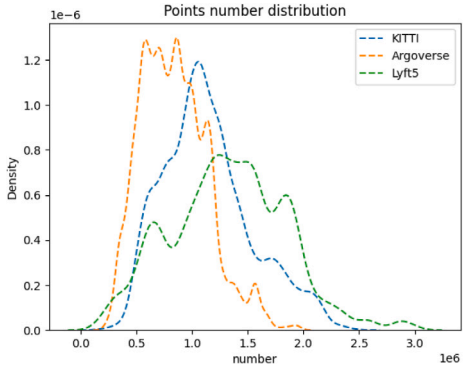


Fig. 8. Quantity distribution of points contained by each point cloud in the three datasets, including KITTI, Argoverse, and Lyft5.

4.1.2. Map generation

Since the three datasets do not provide the global map, we need to register all LiDAR scans to generate a LiDAR map. A single LiDAR scan only contains partial point clouds of direct line-of-sight on the environment at a location. To obtain the whole LiDAR maps, we aggregate all scans at their ground truth positions. Then, we down-sample the LiDAR maps at a resolution of 0.1 m. Additionally, due to the velocity uncertainty of moving objects, LiDAR points will shift in LiDAR scans. Consequently, the point clouds of these moving objects contain a lot of noise, which is detrimental to accurate depth projection. Therefore, we remove all cars and pedestrians in the LiDAR map according to the 3D bounding boxes provided by the dataset, which makes the depth image more accurate. It should be noticed that the KITTI dataset does not provide those objects' bounding boxes, so we do not post-process the KITTI LiDAR maps. Each point cloud subset corresponding to an RGB image is obtained by cutting from the large-scale LiDAR map. The coverage range of each point cloud is centered on the corresponding ground truth camera pose of the RGB image, extending 100 m forward, 10 m backward, and 25 m left and right, respectively. The number of points contained by each cropped point cloud varies with the scene, and its number distribution in each dataset is visualized in Fig. 8. With the point cloud and GT poses, we can generate accurate LiDAR depth images, which can align well with the corresponding RGB images. To obtain rough initial poses with known pose error, we randomly sample the translation error in [−2 m, 2 m], and rotation error in [−10°, +10°] regarding the GT poses. Then the deformed depth image can be generated by projecting the LiDAR maps using the initial poses. Meanwhile, by comparing the two depth images projected using GT poses and initial poses, we can obtain the GT flow for the front-end model training and evaluation.

4.1.3. Evaluation metrics

To evaluate the performance of our model, we quantitatively use the mean and median errors of the predicted poses as our evaluation metrics. The position errors are measured in centimeters and the rotation errors are quantified in degrees. We assume the position errors larger than four meters as a *failure*. Additionally, the average Endpoint Error (EPE), commonly used in flow estimation tasks, is used to evaluate the front-end flow estimation network. For qualitative comparison with competitors, we further plot the overlapped images of depth and RGB information, the rotation and translation error distributions.

4.2. Experimental setup

Accounting for the network training, we scale and resize images to 960 × 320, and change the intrinsic camera parameters accordingly for different datasets. Note although the input image resolutions are the same, they have different intrinsic parameters for the three datasets. In addition, we randomly change the contrast, saturation, brightness, and mirror flip the images to enhance the diversity of the dataset. For the network, the learning rate, weight decay, and training batch size are set to 4×10^{-5}, 1×10^{-4}, 2, respectively. We use *OneCycleLR* as the learning rate scheduler and train our network using Adam optimizer (Kingma and Ba, 2014). The network is trained for 100 epochs, and the BPnP module participates in training after 30 epochs. All experiments are conducted on a single NVIDIA GTX 3090 GPU. For the deployment of the I2D-Loc, it can be divided into preliminary computations and a real-time onboard component. The training process is part of the preliminary computation, while the trained model is used to conduct the efficient onboard inference.

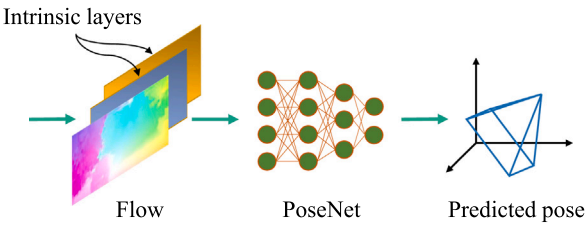


Fig. 9. Pose regression network structure. It directly regresses the translation and rotation of camera poses with the predicted flows and intrinsic layers.

4.3. Ablation study

Generally, the pipeline of I2D-Loc consists of a front-end depth-RGB flow estimation module and a back-end pose estimation module. To verify the effectiveness of I2D-Loc, we discuss the influence of our designed strategies, i.e., depth completion operation, confidence map, normal loss, and BPnP, on the flow and pose estimation results. For the back-end, we compare three pose estimation methods, i.e., traditional RANSAC+PnP, BPnP, and Flow2PoseNet.

Flow2PoseNet, as shown in Fig. 9, is a pose regression network based on ResNet50 to directly regress the translation and rotation using the predicted flows and intrinsic layers, which is similar to Wang et al. (2020). It is independent of the changes of intrinsic parameters by adding intrinsic layers. We conduct extra experiments to verify its feasibility and compare it with the other two pose estimation methods.

For the influence of depth completion operation, confidence map, normal loss, and BPnP, the results are shown in Table 1. Firstly, we only input the sparse depth image and the corresponding RGB image to the model and use a PnP solver in a RANSAC loop to calculate the camera pose. The result is shown in Table 1(a). When the sparse input depth image is replaced with a dense depth image in Table 1(b), EPE, pose error and fail rate are all decreased with a large margin. Then when a confidence map is used to weight the dense depth image, Table 1(c) shows that the localization accuracy is further improved. The above experiments show that the proposed depth representation strategy is more effective than the original sparse depth representation. When the proposed surface normal loss is added to the loss function (Table 1(d)), the localization accuracy is slightly improved, which verifies the effectiveness of the proposed loss function. In Table 1(e), we replace the PnP solver in a RANSAC loop with the BPnP module to calculate the camera pose during training, then still use the PnP solver in a RANSAC loop to estimate the camera pose during testing. The experimental results show that the BPnP module can improve the overall accuracy of flow and camera pose. Finally, in Table 1(e), we observe that all the modules working together further improve the performance of the model and achieve the highest localization accuracy.

For the back-end, we explore the impact of the three different pose estimation modules and the results are shown in Table 2. In these experiments, we keep the same parameter settings and only change the pose estimation module. We first use the PnP solver in a RANSAC

Table 1
Performance comparison of different modules on the KITTI dataset. "D", "C", "N", "B" represent the depth completion operation, confidence map, normal loss, and BPnP module, respectively. The performance metrics on the KITTI dataset show comprehensive performance.

Case	Module				EPE	Mean error		Median error		Fail [%]
	D	C	N	B		Rot. [°]	Trans. [cm]	Rot. [°]	Trans. [cm]	
Initial pose					–	≈9.6726	≈182.8381	≈9.9033	≈187.3079	–
(a)					12.62	0.8861	27.1123	0.7194	19.5648	2.03
(b)	✓				11.52	0.8762	25.6747	0.7266	18.3091	1.63
(c)	✓	✓			11.50	**0.8459**	25.0510	0.7048	18.1515	1.72
(d)	✓		✓		11.46	0.8705	25.4964	0.7155	17.8745	1.74
(e)	✓			✓	10.94	0.8726	25.2878	0.7141	18.0915	**1.59**
(f)	✓	✓	✓	✓	10.75	0.8502	**24.7932**	**0.7039**	**17.7212**	1.61

Table 2
Performance comparison of different back-end pose estimation modules on the KITTI dataset. "Train Back-End" represents pose estimation module used in training. "Val Back-End" represents pose estimation module used in validation.

Case	Train Back-End	Val Back-End	Mean error		Median error		Fail [%]
			Rot. [°]	Trans. [cm]	Rot. [°]	Trans. [cm]	
Initial pose			≈9.6726	≈182.8381	≈9.9033	≈187.3079	–
(a)	PnP+RANSAC	PnP+RANSAC	**0.8459**	25.0510	0.7048	18.1515	1.72
(b)	BPnP	PnP+RANSAC	0.8502	**24.7932**	**0.7039**	**17.7212**	**1.61**
(c)	BPnP	BPnP	1.1217	35.3565	0.8335	22.8746	3.70
(d)	GTFlow2PoseNet	GTFlow2PoseNet	1.3507	58.4423	1.1992	52.0703	3.28
(e)	PreFlow2PoseNet	PreFlow2PoseNet	1.5079	94.2455	1.2729	95.3252	13.83

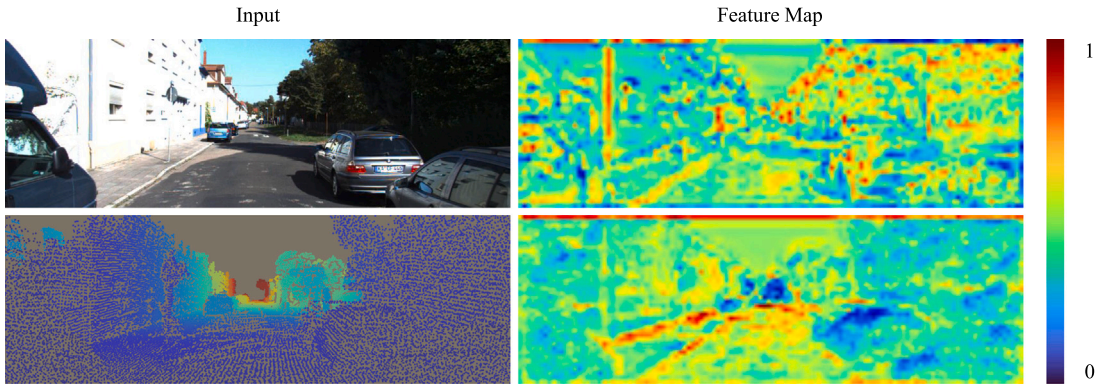


Fig. 10. Feature map visualization from RGB and depth images. I2D-Loc learns to focus on the geometry information like edges of the buildings or lane lines shared by modality.

loop to estimate the camera poses during training and testing. The rotation error is less than 1 degree and the mean translation error is less than 30cm. For the autonomous driving scenarios, this accuracy is fair for navigation and obstacle avoidance. In Table 2(b), we replace the PnP solver with a BPnP module during training, which can calculate the gradients of the pose estimation error implicitly and enable the model to be trained end-to-end. The role of the BPnP is to improve the performance of the front-end optical flow estimator by introducing pose for supervising. Comparing Table 2(a) and (b), we can observe that using BPnP for training and PnP+RANSAC for evaluation relatively improves the comprehensive performance, which bears out BPnP's effectiveness. In Table 2(c), using the BPnP module during testing leads to a lower localization accuracy. This is because the BPnP module does not contain the RANSAC operation and is less robust to outliers. In Table 2(d)–(e), we use the pose regression network Flow2PoseNet to estimate the camera poses directly. Since this pose regression model is independent of camera intrinsics, if the performance is comparable to the traditional ones, we argue this end-to-end network model will be much better. However, no matter we input the predicted flows from the trained front-end network or ground truth flows to the pose regression network, the results are all worse than the traditional PnP solver and BPnP module. The reason might be that the transformation and deformation are too large for pose regression, iterations are needed just as the CMRNet (Cattaneo et al., 2019).

To identify the matching mechanism behind appearance and geometry since they capture different types of information, the feature maps extracted from RGB and depth are visualized in Fig. 10. It is clear that some regular geometric structures, such as the edges of buildings, have higher feature values in the extracted feature maps. It implies that the neural network has learned to infer geometry information from the query image. These similar geometric structures constitute the consistency between RGB images and LiDAR depths, even though pixels in RGB images and LiDAR depths represent different types of information. This geometric consistency is the basis of matching between RGB images and depth maps.

4.4. Results analysis

To evaluate the performance of the proposed I2D-Loc, extensive experiments are conducted on three datasets. We train our model on KITTI and Argoverse datasets. It should be noticed that they have unbalanced training samples (11 426 and 36 347, respectively), which might lead the network to perform better on one dataset than the other. To alleviate this problem, we randomly select the same amount of data from the two datasets for training in each epoch. Because we sample the subset every epoch, the network eventually processes all data from KITTI and Argoverse datasets. For evaluations, all three datasets are used for testing, while Lyft5 is only used for testing the generalization of the proposed I2D-Loc.

The qualitative results are shown in Figs. 11–13 for the three datasets, respectively. The results of front-end flow can rectify the

Fig. 11. Qualitative results on the KITTI dataset. Point clouds are projected using the initial poses and overlapped with the RGB images (First column). After estimating the flow using I2D-Loc, initial depth images are warped to overlapped with RGB (Second column). The third column is the depth images projected based on the ground truth pose. The warped depth images have the best structural consistency with RGB images and are most similar to ground truth.

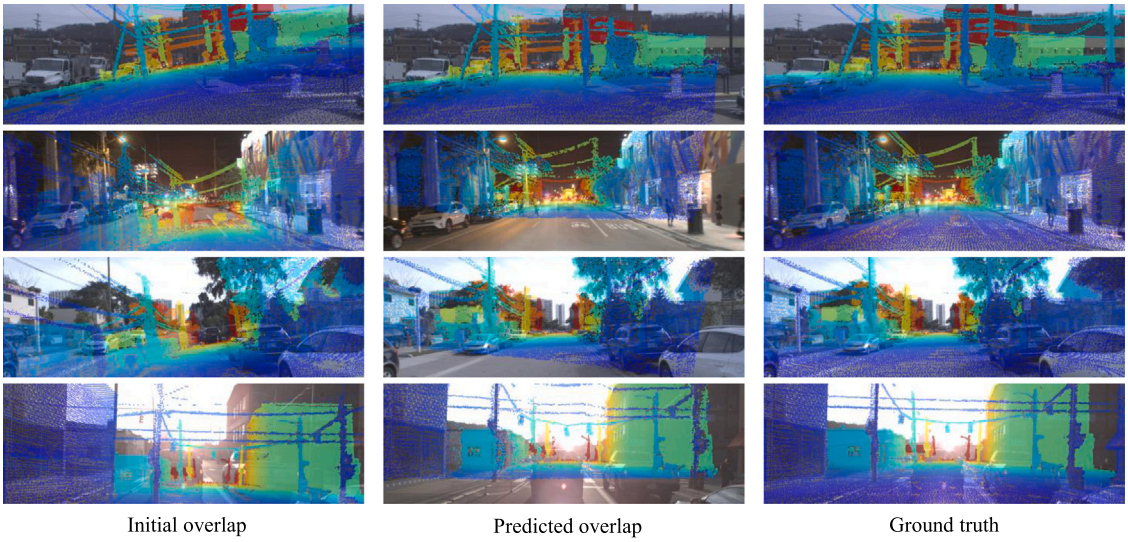


Fig. 12. Qualitative results on the Argoverse dataset. Compared with the KITTI dataset, it consists of more various weather conditions such as cloudy, night, sunny and strong light from the first to the fourth row. Our I2D-Locstill gives the best structural consistency on the predicted overlap between depth and RGB.

miss-alignment caused by wrong initial poses. Therefore, the overlap between RGB and depth images can directly show the performance of 2D–3D correspondences. Better geometry consistency between RGB and depth images can lead to better localization results. In each figure, the first column shows the depth images projected based on the initial poses overlapped on the RGB images, the second column shows the depth images warped from the initial depth images based on the predicted flow, and the third column shows the depth images projected based on the ground truth poses. We can observe that although the three datasets are with different illuminations, weather, and geometric structures, the overall predicted overlap between RGB and depth from the initial overlap shows better geometric consistency, which is similar to the overlap projected by the GT pose. These qualitative results show that the proposed I2D-Loccan estimate the correspondences between RGB and depth images and has good generalization for various scenes.

To further quantitatively evaluate the localization performance of the I2D-Loc, we plot the distribution of rotation and translation errors between the predicted poses and the ground truth poses. In Fig. 14, the red line represents the error distribution of the initial poses' error, which fits the normal distribution with a mean of (2 m, 10°). The purple line represents the error distribution of the predicted poses in CMRNet (Cattaneo et al., 2019). CMRNet directly regresses the camera poses with the flow on sparse LiDAR depth to RGB images, which cannot adapt to the changes of camera intrinsics. The green line represents the error distribution of the predicted poses in our model. The graphs in the first column show the validation results on KITTI. It can be observed that the curve of our model is steeper and closer to the y-axis, which means a smaller mean and smaller variance. Therefore, our model is significantly superior to CMRNet. In addition, We conduct the same experiments on Argoverse and Lyft5, the error distribution curves are shown in the second and third columns of Fig. 14, respectively. It should be noticed that the network of CMRNet tightly-couples the pose estimation with the intrinsic camera parameters, therefore we need to retrain the CMRNet on every single dataset and test only on the same dataset. From the error distribution maps, we can find that the mean rotation and translation errors for CMRNet are smaller than the initial poses, but much worse than our I2D-Loc even it is retrained on every dataset.

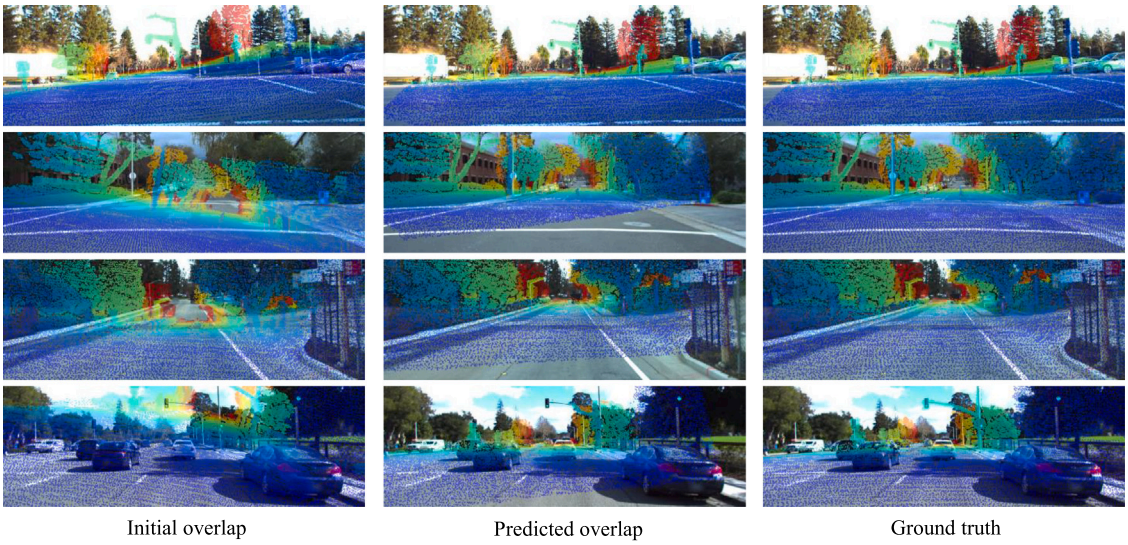


Fig. 13. Qualitative results on the Lyft5 dataset. Compared with the above datasets, it consists of some wide scenes outside the structured city. Our model has not been trained on this dataset but it still gives good results compared with ground truth, which demonstrates the generalization of the I2D-Loc.

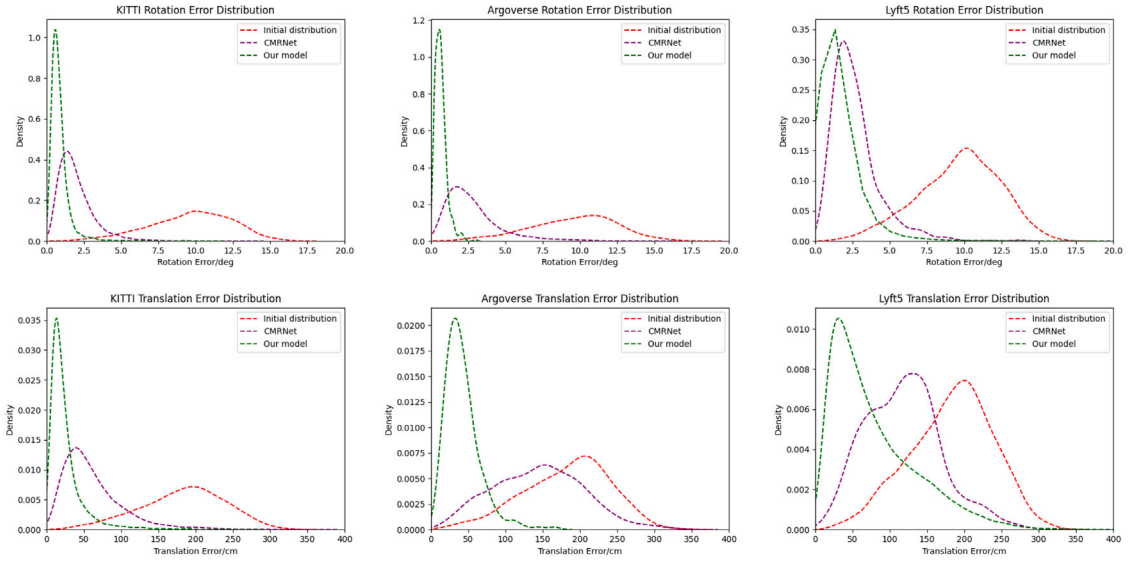


Fig. 14. Error distributions on three datasets (KITTI, Argoverse, and Lyft5 respectively). The initial pose error fits the normal distribution. For the validation results on the KITTI dataset, the mean and median of the Rotation error are 0.85° and 0.70°, and the mean and median of the Translation error are 24.79 cm and 17.72 cm. For the validation results on the Argoverse dataset, the mean and median of the Rotation error are 0.90° and 0.71°, and the mean and median of the Translation error are 55.74 cm and 47 cm. For the validation results on the Lyft5 dataset, the mean and median of the Rotation error are 1.71° and 1.18°, and the mean and median of the Translation error are 78.30 cm and 61 cm. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 3
Performance comparison on KITTI, Argoverse, and Lyft5 datasets.

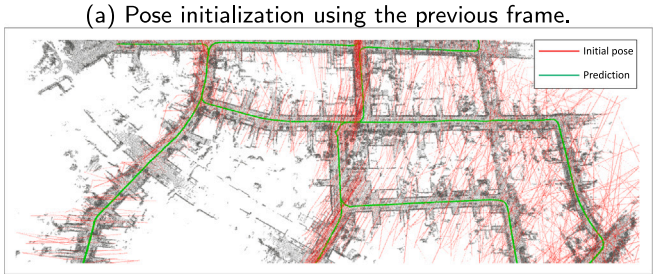
Case	KITTI localization error			Argoverse localization error			Lyft5 localization error		
	Transl. [cm]	Rot. [deg]	Fail [%]	Transl. [cm]	Rot. [deg]	Fail [%]	Transl. [cm]	Rot. [deg]	Fail [%]
CMRNet (Cattaneo et al., 2019)	46	1.60	–	145	2.32	–	118	2.33	–
CMRNet++ (Cattaneo et al., 2020)	55	1.46	2.18	80	1.55	**6.24**	132	2.13	**10.56**
HyperMap (Chang et al., 2021)	48	1.42	–	58	0.93	–	–	–	–
Ours	**18**	**0.70**	**1.61**	**47**	**0.71**	7.52	**61**	**1.18**	15.44

Finally, to demonstrate the SOTA performance of the proposed I2D-Loc, we compare the proposed I2D-Net with the latest learning-based camera localization competitors including CMRNet (Cattaneo et al., 2019), CMRNet++ (Cattaneo et al., 2020), and HyperMap (Chang et al., 2021). CMRNet++ modifies the back-end pose regression module of CMRNet with the traditional PnP solver, thus can be directly trained one time and tested on other datasets. HyperMap follows the pipeline of CMRNet with a pre-computed 3D feature map. Since the source code and pre-trained model of HyperMap are not publicly available, we directly use the results in their paper (Chang et al., 2021). The quantitative results are shown in Table 3. It can be observed that our model obtains the minimum translation and rotation errors among the three datasets. Since LiDAR maps are the aggregations of local scans, the moving cars will inevitably influence the flow estimation

Table 4
Localization accuracy of I2D-Loc by training 3 models under 3 different maximum error ranges of the initial camera poses. The maximum range of the three iterations are [±2 m, ±10°], [±1 m, ±2°] and [±0.6 m, ±1°], respectively.

	CMRNet			CMRNet++			Ours		
	Transl. [cm]	Rot. [deg]	Fail [%]	Transl. [cm]	Rot. [deg]	Fail [%]	Transl. [cm]	Rot. [deg]	Fail [%]
Iteration 1	51	1.39	0	55	1.46	2.18	**18**	**0.70**	**1.61**
Iteration 2	31	1.09	0	22	0.77	0	**11**	**0.39**	0
Iteration 3	27	1.07	0	14	0.43	0	**8**	**0.30**	0

(a) Pose initialization using the previous frame.



(b) Pose initialization using a random disturbance to the GT.

Fig. 15. The top view of the final trajectories of I2D-Loc using different pose initialization schemes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

performance. Therefore we remove all cars in Argoverse and Lyft5 LiDAR maps. This might be the reason why the failure rates of our I2D-Loc are slightly higher than CMRNet++ on Argoverse and Lyft5. This car removal can result in some failure cases when the projected depth images do not have enough geometric constraints to match with corresponding images.

In addition, like CMRNet (Cattaneo et al., 2019) and CMRNet++ (Cattaneo et al., 2020), we also train three models under different maximum error ranges of the initial camera poses to verify the effectiveness of using more iterations. The experimental results are shown in Table 4. It shows that the performance of the proposed I2D-Loc can also be improved significantly with more iterations, and the proposed I2D-Loc outperforms the state-of-the-art methods (Cattaneo et al., 2019, 2020) using the scheme of iteration.

4.5. Discussions

For the LiDAR depth image generation, the pipeline of I2D-Loc needs a rough pose initial guess. We assume it may be from GPS, image retrieval, or pose from the previous frame for continuous video sequences. Therefore the accuracy of the initial pose may affect the algorithm. If no initial pose is provided, the 2D–3D localization will become a different problem of global matching between a single image and large-scale point clouds. It is an NP-hard problem, which is time-consuming and can easily fall into a local optimum. Fig. 15 shows the localization results of the proposed I2D-Loc using different pose initialization schemes. Fig. 15(a) shows the predicted trajectory of 2D–3D pose tracking using pose initialization with the previous frame.

Table 5
Inference time comparison with different pose estimators. The inference time only includes the time spending on pose estimation.

	PnP+RANSAC	BPnP	PoseNet
Inference time [ms]	25	75	5

The corresponding video demo is also provided.[1] The red curve represents the ground truth trajectory, and the green curve represents the prediction. The figure shows that the two trajectories align well, indicating that the 2D–3D pose tracking error is small. This 2D–3D pose tracking application is an example for continuous videos, which is useful for small aerial robot localization and autonomous driving. Fig. 15(b) shows the trajectory with pose initialization using a random disturbance to the ground truth. The corresponding video demo is provided.[2] This is a simulation of camera localization with pose initialization using GPS or image retrieval. The red curve represents the initial rough pose, and the green curve represents the prediction. It is clear that using I2D-Loc, the green trajectory is smooth and aligns well with the large-scale LiDAR map.

In our experiments, we set the initial translation error between $[-2\text{ m}, 2\text{ m}]$ for each axis, therefore the maximal position error is about $2\sqrt{3}$ m, which is close to the localization accuracy of GPS in real life. For the initial rotation error, it is set within $[-10°, +10°]$ for each axis of Euler angles, which can be easily obtained using IMU sensors. If the initial error between the initial pose and the GT pose is too large, there will be only a small or even no overlap between RGB and LiDAR depth, which makes our model prone to failure. Besides, by analyzing the failure cases as shown in Fig. 16, our model lack robustness when there exists extreme degeneracy, e.g., homogeneous depth projections, extremely dark or bright RGB images, complexly textured RGB images.

For the choice of pose estimator, PnP + RANSAC can yield accurate pose estimations for the mathematical model of 3D–2D projection. However, it is non-differentiable due to the hypothesis selection procedure. Neural networks have shown powerful fitting abilities to offer potentials for 6-DoF pose regression (Cattaneo et al., 2019; Wang et al., 2020). Therefore, both BPnP and PoseNet are introduced to make the pose estimation end-to-end and independent of the camera intrinsics.

For the BPnP module, the gradients of the PnP solver can be back-propagated to guide parameter optimization of the flow estimator based on the implicit differentiation. Experimental results validate this back-end can be integrated into the end-to-end network and can keep the estimation accuracy.

Besides, we consider that using a neural network to directly regress the poses from the predicted flow is also promising. Using a neural network requires less inference time because it does not involve an iterative optimization process. As a result, we try to use the regression network PoseNet to estimate the camera poses directly. The inference time comparison is shown in Table 5, which indicates the potential of using neural networks to predict camera poses at higher efficiency and be independent of camera intrinsic parameters in an end-to-end manner.

[1] https://youtu.be/C_oF8Sl7pKM.
[2] https://youtu.be/dBjFTBT_xw0.

Fig. 16. Examples of failure cases. The degeneracy comes from homogeneous depths, extremely dark or bright RGB images, complexly textured RGB images, etc.

Table 6
Localization accuracy on the KITTI dataset.

Methods	RTE [m]	RRE [°]
Direct Regression (Li and Lee, 2021)	4.94 ± 2.87	21.98 ± 31.97
MonoDepth2 (Godard et al., 2019) + GT-ICP	2.9 ± 2.5	12.4 ± 10.3
DeepI2P (Li and Lee, 2021)	3.28 ± 3.09	7.56 ± 7.63
DeepI2P(retrained)	3.22 ± 1.99	6.55 ± 6.58
Ours	0.18 ± 0.17	0.46 ± 0.42

In addition to the methods mentioned in Table 3, we also compare our method with several camera localization methods based on 2D–3D registration. The registration is evaluated with two criteria (Li and Lee, 2021): average Relative Translation Error (RTE) and average Relative Rotation Error (RRE). The comparison results are shown in Table 6. These methods solve the image-to-point registration problem under different settings so they may have unfair advantages or disadvantages over ours. Direct regression (Li and Lee, 2021) encodes the point cloud and image respectively and regresses the relative poses directly. MonoDepth2 (Godard et al., 2019)+GT-ICP uses Monodepth2 to estimate a depth map from a single image and then obtain point-to-point correspondences based on Iterative Closest Point(ICP). DeepI2P (Li and Lee, 2021) converts the cross-modality registration problem into a classification and inverse camera projection optimization problem. We also retrained DeepI2P under the same setup as ours to give a fair comparison. The experimental results show that our method achieves the best performance.

5. Conclusions

In this paper, we propose an end-to-end camera localization network I2D-Loc to estimate the accurate 6-DoF camera pose in the existing LiDAR map. Considering the difference between LiDAR point clouds and RGB imagery, we propose a simple yet efficient LiDAR depth representation based on depth completion and the confidence map. The depth-RGB flow network can establish dense 2D–3D correspondences regardless of LiDAR depth sparsity and 3D occlusion. Comparing the traditional PnP solver, BPnP module, and pose regression network, we obtain the best localization accuracy and generalization with the BPnP module in an end-to-end manner. Experimental results on three public autonomous driving datasets demonstrate the effectiveness and generalization of I2D-Loc. Future work will focus on an efficient LiDAR feature encoding to handle large-scale LiDAR map projection.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was partially supported by the Fundamental Research Funds for the Central Universities under Grant 2042022kf1010, the National Natural Science Foundation of China under Grant 62271354, and the Natural Science Foundation of Hubei Province, China. The numerical calculations were conducted on the supercomputing system in the Supercomputing Center, Wuhan University. Sincere thanks to the editorial board and the anonymous reviewers for their comments and suggestions.

References

Acharya, D., Khoshelham, K., Winter, S., 2019. BIM-PoseNet: Indoor Camera Localisation using a 3D Indoor Model and Deep Learning from Synthetic Images. ISPRS J. Photogramm. Remote Sens. 150, 245–258.

Balntas, V., Li, S., Prisacariu, V., 2018. RelocNet: Continuous Metric Learning Relocalisation using Neural Nets. In: European Conference on Computer Vision. pp. 751–767.

Behley, J., Stachniss, C., 2018. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In: Robotics: Science and Systems, Vol. 2018. pp. 1–10.

Brachmann, E., Krull, A., Nowozin, S., Shotton, J., Michel, F., Gumhold, S., Rother, C., 2017. DSAC - Differentiable RANSAC for Camera Localization. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6684–6692.

Brachmann, E., Rother, C., 2018. Learning Less Is More - 6D Camera Localization via 3D Surface Regression. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4654–4662.

Brachmann, E., Rother, C., 2019. Expert Sample Consensus Applied to Camera Re-Localization. In: IEEE International Conference on Computer Vision. pp. 7525–7534.

Brachmann, E., Rother, C., 2021. Visual Camera Re-Localization from RGB and RGB-D Images using DSAC. IEEE Trans. Pattern Anal. Mach. Intell. 1–10.

Campbell, D., Petersson, L., Kneip, L., Li, H., 2018. Globally-Optimal Inlier Set Maximisation for Camera Pose and Correspondence Estimation. IEEE Trans. Pattern Anal. Mach. Intell. 42 (2), 328–342.

Campbell, D., Petersson, L., Kneip, L., Li, H., Gould, S., 2019. The Alignment of the Spheres: Globally-Optimal Spherical Mixture Alignment for Camera Pose Estimation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11796–11806.

Caselitz, T., Steder, B., Ruhnke, M., Burgard, W., 2016. Monocular Camera Localization in 3D LiDAR Maps. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 1926–1931.

Cattaneo, D., Sorrenti, D.G., Valada, A., 2020. CMRNet++: Map and Camera Agnostic Monocular Visual Localization in LiDAR Maps. In: IEEE ICRA 2020 Workshop on Emerging Learning and Algorithmic Methods for Data Association in Robotics.

Cattaneo, D., Vaghi, M., Ballardini, A.L., Fontana, S., Sorrenti, D.G., Burgard, W., 2019. CMRNet: Camera to LiDAR-Map Registration. In: IEEE Intelligent Transportation Systems Conference. pp. 1283–1289.

Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., et al., 2019. Argoverse: 3D Tracking and Forecasting With Rich Maps. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8748–8757.

Chang, M.-F., Mangelson, J., Kaess, M., Lucey, S., 2021. HyperMap: Compressed 3D Map for Monocular Camera Registration. In: IEEE International Conference on Robotics and Automation. pp. 11739–11745.

Chen, B., Parra, A., Cao, J., Li, N., Chin, T.-J., 2020. End-to-End Learnable Geometric Vision by Backpropagating PnP Optimization. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8100–8109.

David, P., DeMenthon, D., Duraiswami, R., Samet, H., 2003. Simultaneous Pose and Correspondence Determination using Line Features. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vol. 2. pp. 1–10.

Deng, H., Birdal, T., Ilic, S., 2018. PPF-FoldNet: Unsupervised Learning of Rotation Invariant 3D Local Descriptors. In: European Conference on Computer Vision. pp. 602–618.

Ding, M., Wang, Z., Sun, J., Shi, J., Luo, P., 2019. CamNet: Coarse-to-Fine Retrieval for Camera Re-Localization. In: IEEE International Conference on Computer Vision. pp. 2871–2880.

Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., Brox, T., 2015. FlowNet: Learning Optical Flow With Convolutional Networks. In: IEEE International Conference on Computer Vision. pp. 2758–2766.

Engel, J., Schöps, T., Cremers, D., 2014. LSD-SLAM: Large-Scale Direct Monocular SLAM. In: European Conference on Computer Vision. pp. 834–849.

Feng, Y., Fan, L., Wu, Y., 2015. Fast Localization in Large-Scale Environments Using Supervised Indexing of Binary Features. IEEE Trans. Image Process. 25 (1), 343–358.

Feng, M., Hu, S., Ang, M.H., Lee, G.H., 2019. 2D3D-Matchnet: Learning To Match Keypoints Across 2D Image And 3D Point Cloud. In: IEEE International Conference on Robotics and Automation. pp. 4790–4796.

Geiger, A., Lenz, P., Stiller, C., Urtasun, R., 2013. Vision meets robotics: The KITTI dataset. Int. J. Robot. Res. 32 (11), 1231–1237.

Godard, C., Mac Aodha, O., Firman, M., Brostow, G.J., 2019. Digging into Self-supervised Monocular Depth Estimation. In: IEEE International Conference on Computer Vision. pp. 3828–3838.

Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T., 2017. FlowNet 2.0: Evolution of Optical Flow Estimation With Deep Networks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2462–2470.

Kendall, A., Cipolla, R., 2017. Geometric Loss Functions for Camera Pose Regression With Deep Learning. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5974–5983.

Kesten, R., Usman, M., Houston, J., Pandya, T., Nadhamuni, K., Ferreira, A., Yuan, M., Low, B., Jain, A., Ondruska, P., et al., 2019. Lyft level 5 AV dataset. https://level5.lyft.com/dataset.

Kim, Y., Jeong, J., Kim, A., 2018. Stereo Camera Localization in 3D LiDAR Maps. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 1–9.

Kingma, D.P., Ba, J., 2014. Adam: A Method for Stochastic Optimization. In: International Conference on Learning Representations. pp. 1–10.

Ku, J., Harakeh, A., Waslander, S.L., 2018. In defense of classical image processing: Fast depth completion on the cpu. In: 2018 15th Conference on Computer and Robot Vision. CRV, IEEE, pp. 16–22.

Lepetit, V., Moreno-Noguer, F., Fua, P., 2009. EPnP: An Accurate O(n) Solution to the PnP Problem. Int. J. Comput. Vis. 81 (2), 155–166.

Li, J., Lee, G.H., 2019. USIP: Unsupervised Stable Interest Point Detection From 3D Point Clouds. In: IEEE International Conference on Computer Vision. pp. 361–370.

Li, J., Lee, G.H., 2021. DeepI2P: Image-to-Point Cloud Registration via Deep Classification. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15960–15969.

Li, H., Yu, H., Wang, J., Yang, W., Yu, L., Scherer, S., 2021. ULSD: Unified Line Segment Detection Across Pinhole, Fisheye, and Spherical Cameras. ISPRS J. Photogramm. Remote Sens. 178, 187–202.

Liu, Y., Huang, T.S., Faugeras, O.D., 1990. Determination of Camera Location From 2-D to 3-D Line and Point Correspondences. IEEE Trans. Pattern Anal. Mach. Intell. 12 (1), 28–37.

Liu, K., Li, Q., Qiu, G., 2020. PoseGAN: A Pose-to-Image Translation Framework for Camera Localization. ISPRS J. Photogramm. Remote Sens. 166, 308–315.

Lowe, D.G., 1999. Object Recognition from Local Scale-invariant Features. In: IEEE International Conference on Computer Vision, Vol. 2. pp. 1150–1157.

Moreno-Noguer, F., Lepetit, V., Fua, P., 2008. Pose Priors for Simultaneously Solving Alignment and Correspondence. In: European Conference on Computer Vision. pp. 405–418.

Mur-Artal, R., Tardós, J.D., 2017. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. IEEE Trans. Robot. 33 (5), 1255–1262.

Pintus, R., Gobbetti, E., Agus, M., 2011. Real-time Rendering of Massive Unstructured Raw Point Clouds using Screen-Space Operators. In: International Conference on Virtual Reality, Archaeology and Cultural Heritage. pp. 105–112.

Qin, T., Li, P., Shen, S., 2018. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. IEEE Trans. Robot. 34 (4), 1004–1020.

Sattler, T., Leibe, B., Kobbelt, L., 2016. Efficient & Effective Prioritized Matching for Large-Scale Image-Based Localization. IEEE Trans. Pattern Anal. Mach. Intell. 39 (9), 1744–1756.

Shi, C., Li, J., Gong, J., Yang, B., Zhang, G., 2022. An Improved Lightweight Deep Neural Network with Knowledge Distillation for Local Feature Extraction and Visual Localization using Images and LiDAR Point Clouds. ISPRS J. Photogramm. Remote Sens. 184, 177–188.

Stewart, A.D., Newman, P., 2012. LAPS - Localisation using Appearance of Prior Structure: 6-DoF Monocular Camera Localisation using Prior Pointclouds. In: IEEE International Conference on Robotics and Automation. pp. 2625–2632.

Sun, D., Yang, X., Liu, M.-Y., Kautz, J., 2018. PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8934–8943.

Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A., 2017. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In: Thirty-First AAAI Conference on Artificial Intelligence. pp. 4278–4284.

Teed, Z., Deng, J., 2020. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In: European Conference on Computer Vision. pp. 402–419.

Wang, W., Hu, Y., Scherer, S., 2020. Tartanvo: A generalizable learning-based VO. In: The Conference on Robot Learning.

Wolcott, R.W., Eustice, R.M., 2014. Visual Localization within LIDAR Maps for Automated Urban Driving. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 176–183.

Yew, Z.J., Lee, G.H., 2018. 3DFeat-Net: Weakly Supervised Local 3D Features for Point Cloud Registration. In: European Conference on Computer Vision. pp. 607–623.

Yu, H., Zhen, W., Yang, W., Scherer, S., 2020. Line-Based 2-D–3-D Registration and Camera Localization in Structured Environments. IEEE Trans. Instrum. Meas. 69 (11), 8962–8972.

Zhang, J., Singh, S., 2014. LOAM: Lidar odometry and mapping in real-time. In: Robotics: Science and Systems, Vol. 2. pp. 1–9.

Zhong, Y., 2009. Intrinsic Shape Signatures: A Shape Descriptor for 3D Object Recognition. In: IEEE International Conference on Computer Vision Workshops. pp. 689–696.

Zuo, X., Geneva, P., Yang, Y., Ye, W., Liu, Y., Huang, G., 2019. Visual-Inertial Localization With Prior LiDAR Map Constraints. IEEE Robot. Autom. Lett. 4 (4), 3394–3401.