Computational Visual Media
https://doi.org/10.1007/s41095-022-0319-z

Research Article

Multi-task learning and joint refinement between camera localization and object detection

Junyi Wang[1,2], **Yue Qi**[1,2,3] (✉)

Abstract Visual localization and object detection both play important roles in various tasks. In many indoor application scenarios where some detected objects have fixed positions, the two techniques work closely together. However, few researchers consider these two tasks simultaneously, because of a lack of datasets and the little attention paid to such environments. In this paper, we explore multi-task network design and joint refinement of detection and localization. To address the dataset problem, we construct a medium indoor scene of an aviation exhibition hall through a semi-automatic process. The dataset provides localization and detection information, and is publicly available at `https://drive.google.com/drive/folders/1U28zkON4_IOdbzkqyIAKlAl5k9oUKOjI?usp=sharing` for benchmarking localization and object detection tasks. Targeting this dataset, we have designed a multi-task network, JLDNet, based on YOLO v3, that outputs a target point cloud and object bounding boxes. For dynamic environments, the detection branch also promotes the perception of dynamics. JLDNet includes image feature learning, point feature learning, feature fusion, detection construction, and point cloud regression. Moreover, object-level bundle adjustment is used to further improve localization and detection accuracy. To test JLDNet and compare it to other methods, we have conducted experiments on 7 static scenes, our constructed dataset, and the dynamic TUM RGB-D and Bonn datasets. Our results show state-of-the-art accuracy for both tasks, and the benefit of jointly working on both tasks is demonstrated.

Keywords visual localization; object detection; joint optimization; multi-task learning

1 Introduction

Regarded as key research problems in computer vision, visual localization and object detection both play indispensable roles in augmented reality (AR), mixed reality (MR), and robotics [1–6]. A localization algorithm determines the 6-DoF (degrees of freedom) camera pose within the target environment, which is fundamental to rendering virtual elements within the real environment. An object detection framework classifies target objects, and provides a bounding box for each.

In some application settings, such as museums, shopping malls, scenic spots, and so on, some detected objects have a unique character while always staying at fixed positions. In such scenes, localization and detection include the following characteristics and difficulties. In terms of localization, various virtual information, such as product introduction, animation, and so on, is always closely related to the pre-established 3D model. Thus, the relocalization problem is particularly important. Furthermore, certain specific circumstances (of lighting or occlusion) may cause an AR library such as ARCore to fail. In addition, medium to large scenes require a long processing time, which may lead to accumulation of errors in simultaneous localization and mapping (SLAM) systems. In terms of detection, the peculiarity of detected objects leads to failures through deep networks trained on

1 State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing 100191, China. E-mail: J. Wang, wangjy0524@163.com; Y. Qi, qy@buaa.edu.cn (✉).
2 Peng Cheng Laboratory, Shenzhen 518052, China.
3 Qingdao Research Institute of Beihang University, Qingdao 266104, China.
∗ Junyi Wang's present address: School of Computer Science and Technology, Shandong University, Qingdao, China.

Manuscript received: 2022-07-03; accepted: 2022-10-03

a universal dataset. Manual annotation of detection results typically takes a long time and much human effort. Furthermore, in such situations, detected objects with fixed positions result in close relations between detection and localization tasks, yet these relations are not exploited by current methods. In dynamic environments, detection of dynamics also contributes to the relocalization process.

To address the above problems, we consider multi-task learning and a joint refinement design for camera localization and object detection. Current localization approaches may be divided into methods based on hand-crafted features, direct learning and indirect learning. Hand-crafted feature based approaches exploit feature matching to calibrate camera pose and graph optimization to reduce the accumulated error. Various SLAM systems use a pipeline [7–9]. Furthermore, semantic information learned by neural networks may be embedded into the original SLAM process to enhance localization accuracy and robustness [10–12], and this seems to be a good solution for these tasks. However, the problems of accumulated errors and image labeling still exist.

As an alternative, deep networks have achieved great success in various computer vision tasks, such as segmentation, detection, and classification. PoseNet [13] and subsequent works regressed the 6-DoF camera pose by training deep networks [14–18]. Moreover, the multi-task network learns the inter-task relationships and shares contextual information between related tasks.

By using deep networks for localization, PoseNet and subsequent works directly regressed the 6-DoF camera pose. Going further, VLocNet++ [19] learned inter-task relationships and shared contextual information to determine 6-DoF global pose, odometry, and semantics, and achieves state-of-the-art results for a direct learning approach. Compared to hand-crafted feature based methods, its strengths are reflected in computational effectiveness, and lack of scale drift and accumulated errors, while its weakness is the imprecise localization result. Instead of direct localization by learning features, indirect learning based methods utilize a deep network to establish dense coordinate correspondences. Then the camera pose is evaluated by applying the Kabsch or PnP algorithm with RANSAC to the correspondences, aiming to reduce uncertainties in the learning process [20–25]. For dynamic environments, Dong et al. [26] bridged deep learning and a regression forest through a novel outlier-aware neural tree. Currently, this kind of approach has attracted much research interest, achieving state-of-the-art results [27–29].

In object detection, both accuracy and running speed are critical considerations. YOLO and its extensions [30–34] have superior calculation speed without significant loss of accuracy. In this paper, we design a joint learning network for camera relocalization and object detection by combining point learning, feature fusion, and a point cloud decoder based on YOLO v3 [31]. On the one hand, the detected objects having fixed positions are closely related to the camera pose. On the other hand, object detection also contributes to localization in dynamic environments. We know of no previous work designing a unified multi-task deep network for both camera localization and object detection tasks.

As Fig. 1 shows, we investigate multi-task network

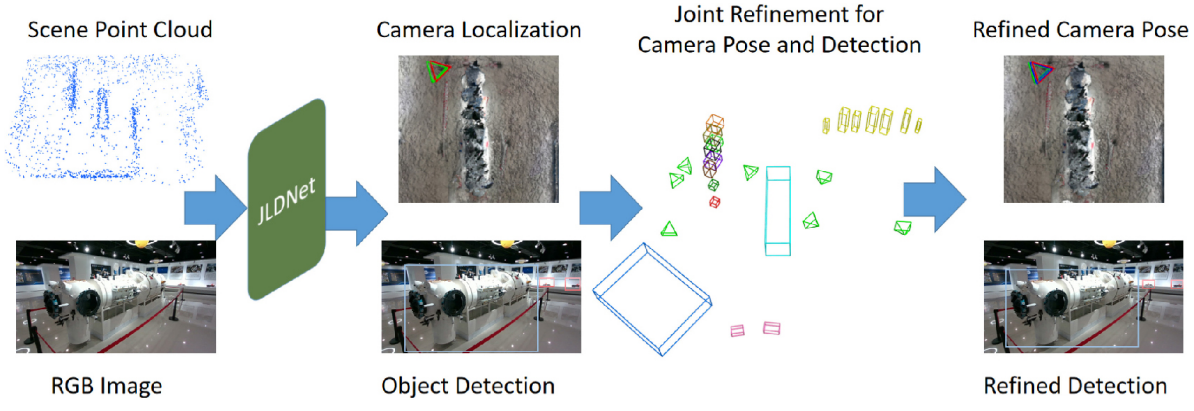

Fig. 1 Joint localization and detection. In the localization subfigure, green indicates the ground truth pose, red indicates the initial pose result, and blue shows the refined pose.

design and joint refinement of object detection and camera relocalization. Firstly, we used a semi-automated process to collect and construct a detection and localization dataset showing an aviation exhibition hall. Secondly, we have designed a multi-task deep network, JLDNet (joint localization and detection network), to regress both the target point cloud and detection boxes. An object-level refinement process operates on the results to further improve accuracy.

In summary, our main technical contributions are as follows.

(1) A semi-automated process to construct a detection and localization dataset, using structure from motion (SFM), multi-view stereo (MVS), 3D manual annotation, and 2D label generation; it takes less than an hour to label a whole scene with thousands of images. The procedure was used to gather and construct a medium-sized indoor scene of an aviation exhibition hall with 6-DoF camera poses and object labels. The dataset is publicly available at `https://drive.google.com/drive/folders/1U28zkON4_I0dbzkqyIAKlAl5k9oUK0jI?usp=sharing` as a benchmark for joint localization and detection.

(2) A multi-task deep network, JLDNet, based on YOLO v3, for joint relocalization and detection, and a suitable mixed loss function. It outputs target point clouds and object bounding boxes by combining 2D image and 3D point features.

(3) A joint object-level bundle adjustment (BA) process which operates on the detection and localization results from JLDNet. Going beyond traditional BA, object constraints are used to leverage high-level semantic information, further improving localization and detection accuracy.

To validate the approach and assess our method's capabilities, as well as comparing it to other methods, we conducted experiments on the 7 Scenes [24], our constructed aviation exhibition hall, TUM RGB-D [35], and Bonn [36], for both both static and dynamic scenes. The experiments demonstrate competitive results on 7 Scenes and state-of-the-art performance on our dataset, TUM RGB-D, and Bonn. Furthermore, we show the positive effects of the refinement stage and mutual promotion between the two tasks.

In the following, related works are summarized in Section 2. Section 3 considers construction of an aviation exhibition hall scene with camera poses and bounding boxes. Section 4 details the architecture and loss function of JLDNet, as well as the joint object-level refinement procedure. Section 5 reports our experiments and Section 6 draws conclusions.

2 Related work

2.1 Relocalization

Current research covers three kinds of relocalization algorithm: hand-crafted feature based methods, direct learning and indirect learning based approaches. Traditional hand-crafted feature based processes include various SLAM systems that exploit hand-crafted feature matching to obtain an initial pose, and global optimization to reduce accumulated errors [7–9, 37, 38]. Moreover, to improve results in dynamic environments and obtain a semantic map, semantic information is embedded in the original process [12, 39, 40]. Using hand-crafted features can achieve very precise results via complex optimization procedures (like SFM). However, they are slow and low-quality images are discarded. In this paper, we exploit the SFM pipeline to establish a dataset for an aviation exhibition hall. Alternatively, SLAM systems provide real-time localization and are widely applied in actual applications. In the relocalization process, hand-crafted features are often not robust to changing environmental conditions such as illumination or weather. Furthermore, accuracy is usually lower than for indirect learning based approaches.

Direct learning feature based methods take advantage of the deep network to preserve relations between the image and camera pose. After the first work, PoseNet, various researchers have focused on the loss function and network architecture to improve localization results [15, 19, 41, 42]. Absolute and relative pose approaches exist, sharing a similar process based on feature extraction and camera pose regression. Absolute pose approaches directly output the absolute camera pose [13–18, 43]. VLocNet [41] utilized auxiliary learning to leverage relative pose information during training. Based on it, VLocNet++ achieves state-of-the-art results by learning the relationship between semantics, 6-DoF global pose

and odometry. In contrast, relative pose methods predict the relative transformation matrix between the test image and one or more training images, rather than the absolute pose [42, 44]. RelocaNet [42] exploits camera frustum overlaps between pairs of images to optimize the network. Because of the insufficient learning linking image features with thousands of dimensions and the 7D camera pose, results are rather less precise than for the other two kinds of approaches.

Indirect learning feature based techniques focus on coordinate transformations instead of regressing camera pose directly [26, 28, 45, 46]. The Kabsch or PnP algorithm with an RANSAC pipeline is then applied to the known source coordinate and estimated target coordinate to obtain the final pose. Currently, this kind of approach achieves state-of-the-art localization performance and possesses strong robustness in static and dynamic environments [26, 27, 29]. To construct the target coordinates, regression forests or deep networks may be used. Using a regression forest, Cavallari et al. [20] applied a pre-trained forest to a new scene on the fly, overcoming the limitation of offline training. Cavallari et al. [29] extended this work to obtain more accurate relocalization performance in real time. By using deep learning, DSAC [47], DSAC++ [45], and DSAC* [27] achieved end-to-end training through a differentiable RANSAC pipeline. KFNet [28] considered in addition the time domain by means of Kalman filtering to establish precise 2D–3D correspondences. Aiming at dynamic environments, Dong et al. bridged deep learning and regression forests through a novel outlier-aware neural tree, obtaining state-of-the-art results. Instead of coordinate regression, Wang et al. [46] designed a novel pipeline composed of point cloud generation and registration. In JLDNet, we also adopt this pipeline with an object detection extension.

2.2 Object detection

The object detection task determines bounding boxes and corresponding classes of single or multiple objects; it is considered to be a much more challenging task than image classification [48–50]. The Region-based CNN (R-CNN) [48] provides a scalable detection algorithm that achieved significant improvements over previous approaches. Based on R-CNN, Fast R-CNN [49] focused on training and testing speed with increased detection accuracy, being more than 200 times faster at testing time. Faster R-CNN [51] proposed a region proposal network to enable faster region proposal, working at 5 frames/s on a GPU while also achieving state-of-the-art object detection accuracy. Going beyond bounding boxes for the detected objects, Mask R-CNN [52] predicts a high-quality object mask by means of an additional segmentation branch.

Compared to region-based methods, an alternative approach exploits a one-stage detector, extracting the bounding box and classifying an object in a unified network. In the single-shot multi-box detector (SSD) [53], the output space of boundary boxes is discretized into a set of default boxes; each feature map has different aspect ratios and scales. YOLO [30] and its variants [31, 33, 34, 54] regard object detection as a regression problem to generate spatially separate bounding boxes and associated class probabilities. YOLO's main benefit is its high speed without significant loss of accuracy, which is also a critical factor for relocalization.

In this paper, we employ the pipeline proposed by Wang et al. Based on the YOLO v3 structure, we learn additional 3D point features and regress the target point cloud. The results later demonstrate the mutual benefit of jointly performing these two tasks.

3 Dataset construction

Current popular indoor localization datasets include 7 Scenes, TUM RGB-D, and ScanNet [55]. For one thing, 7 Scenes and TUM RGB-D do not contain detection information. For another, all camera trajectories in three datasets are limited to a small area.

Thus, for the purposes of deep relocalization and detection tasks for medium indoor scenes, a dataset construction process is presented. Based on the procedure, a dataset of an aviation exhibition hall is gathered and generated, in total containing 9000 slices. In particular, to reduce the time needed for labelling the objects, each 2D bounding box is automatically marked by the camera pose and then manually place 3D labels.

As Fig. 2 shows, the whole process takes the collected images as the input and exports corresponding camera poses and bounding boxes. Firstly, an SFM algorithm is exploited to calibrate the camera pose via suitable parameters. To obtain occlusion

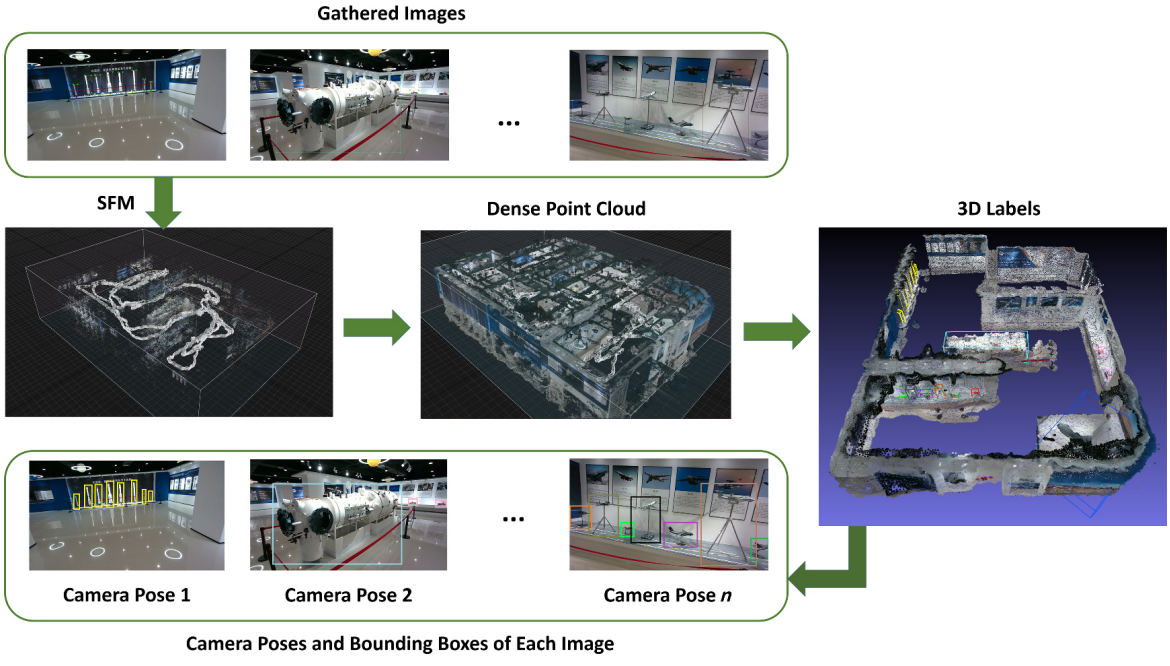

Fig. 2 Dataset construction pipeline. The process involves image collection, SFM, MVS, manual 3D object marking, and 2D label projection.

relationships between different detected objects, we need to construct a dense point cloud and mesh via MVS and a surface reconstruction algorithm. To remove outliers, the original SFM pipeline can detect samples that do not have sufficient feature matches to the global model. After dataset generation, we further check each sample with feature matching distance and coordinate distance. For each sample, we extract the feature and transform it into world space based on predicted pose and depth. Then the feature distance is calculated between the feature point and its corresponding point (given by SFM) in the dense point cloud. Meanwhile, the coordinate distance is also measured. After calculating each feature point, we obtain mean feature and coordinate distances. According to both metrics, we sort all frames in the dataset (more than 12,000 images), and then select the best 9000 samples to construct the final dataset.

After this, we label the 3D bounding box of all detected objects manually in the constructed scene. Then, using the 3D bounding boxes, camera poses, and mesh, the 2D bounding box of each detected object is determined in the source image, completing generation of the dataset. The outcome is visualized in Fig. 2. It is clear that the labelled images are quite accurate, verifying the correctness of camera poses.

It should be noted that our process leverages the projection of marked 3D bounding boxes to construct 2D detections instead of labelling the image directly. Obviously, manually labelled images are more accurate than if using 3D box projection. However, manual labelling takes about 1 min per image and thus 9000 min for all images. By contrast, labelling a 3D scene only needs about 30 min, improving efficiency by a large margin.

Table 1 gives detailed information about our dataset, comparing it to 7 Scenes. Our dataset has the following main characteristics.

- *Medium extent.* The extent of the aviation hall (about 1000 m^3) is much larger (about 100×) than the scenes in 7 Scenes. The large extent obviously increases localization difficulty.
- *Occlusion.* As can be seen, occlusion often occurs between scene objects and walls. This occlusion sometimes leads to large differences between images with little localization changes, causing difficulties for the two tasks.
- *Strong lighting.* The floor of the aviation exhibition hall is tiled, with strong reflections, which may lead to failure of the plane detection algorithm. Various scenes, such as malls, museums, and office buildings, have a similar appearance. The commonly used AR library ARCore fails for such scenes.

Table 1 Comparison of our constructed dataset to 7 Scenes

	Spatial extent	Training / testing frames	Method	Scene character
Our dataset				
Aviation hall	1000 m^3	4000 / 5000 2000 / 7000	SFM + MVS	1. Medium extent 2. Occlusion 3. Strong light
7 Scenes				
Chess	6 m^3	4000 / 2000		
Fire	2.5 m^3	2000 / 2000		
Heads	1 m^3	1000 / 1000		
Office	7.5 m^3	6000 / 4000	Kinect fusion	
Pumpkin	5 m^3	4000 / 2000		
Kitchen	18 m^3	7000 / 5000		
Stairs	7.5 m^3	2000 / 1000		

4 JLDNet

4.1 Localization process

Our framework is based on YOLO v3, which regresses bounding boxes from input images. Wang et al. proposed a localization process comprising a coarse stage with learned features and a refinement stage with hand-crafted features. In this paper, we exploit the idea of the coarse stage as the localization framework.

As Fig. 3(above) shows, we predict the camera pose (as a 4×4 matrix R_{c2w} representing rotation and translation) by applying the point cloud registration algorithm SUPER 4PCS to the source and target point clouds. In detail, the source point cloud P_s in camera space is constructed by uniform sampling

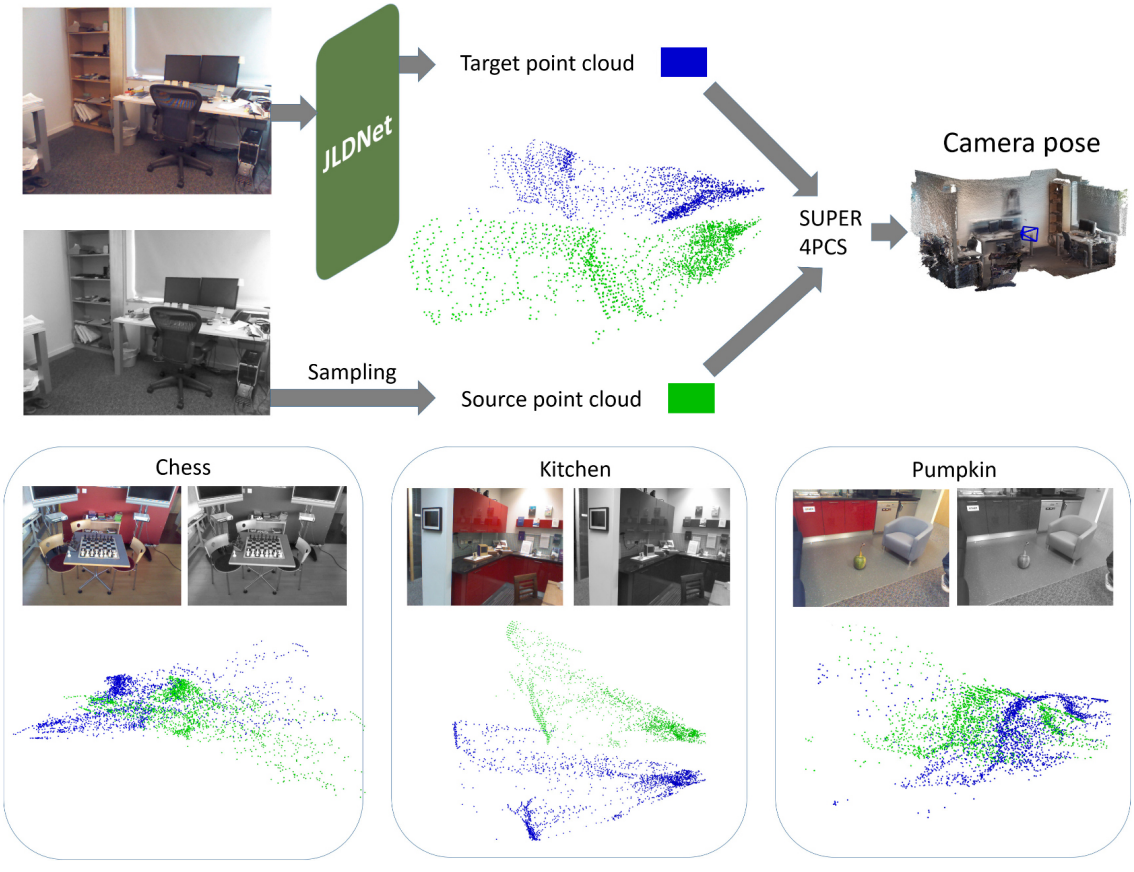

Fig. 3 Localization process and example point clouds. Green: source point cloud. Blue: target.

of the input color image which has known camera intrinsics. Then the ground truth target P_t is obtained by transforming the source into the world space: $P_t = R_{c2w}P_s$. In the localization process, the coordinates in world space of the target point cloud are learned by JLDNet. In particular, the source and target have the same shape, but differ in coordinate space by the transformation R_{c2w}, as shown in Fig. 3(below).

The detailed generation process for the source target point clouds is as follows. With only RGB input, Wang et al. sampled a set number of points from the constructed world point cloud. Then, in the training stage, the ground truth target point cloud is obtained by transforming the sampled point cloud, which results in the same output structure with different input images. In our opinion, this leads to few similarities between consecutive frames. Meanwhile, the source point cloud is unknown to the network. The above two points increase the learning difficulty. Therefore, we adopt the grey value as the depth to construct the target point cloud. In this case, the source point cloud is directly generated from the input image.

After obtaining the source and target point clouds, we use SUPER 4PCS to obtain the camera pose. SUPER 4PCS takes source and target point clouds as inputs, and outputs the transformation (R_{c2w}) from source to target using an optimal linear time output-sensitive global alignment algorithm.

4.2 Architecture

The inputs to JLDNet include the color image and outline scene point cloud. Based on YOLO v3, our overall architecture consists of input processing, feature extraction, feature fusion, point cloud decoder, and object regressor. In particular, the novelty of JLDNet mainly lies in the multi-level feature fusion and loss function design. Its detailed structure is shown in Fig. 4. Black solid lines indicate the original YOLO v3 structure, while dashed lines represent our additions.

We firstly introduce the input process. The scene point cloud is constructed by uniform sampling of the sparse point cloud. Target point cloud generation is typically performed by transforming the depth image from camera space to world space. However, most mobile devices do not provide depth information. Instead, we exploit the normalized grey value to replace the depth value.

Corresponding to the inputs, feature extraction also has three parts. As in YOLO v3, the image feature is learned by DarkNet53, while the scene point cloud is learned by PointNet [56] architecture. JLDNet outputs three image features ($W_1 \times H_1 \times C_1, W_2 \times H_2 \times C_2, W_3 \times H_3 \times C_3$), and a scene point cloud feature (C_4).

After extraction, the feature fusion block combines multi-level image features and point cloud features. In the detection branch, the point cloud feature is transformed to the same size as the corresponding image feature through a tiling operation ($C_4 \to W_1 \times H_1 \times C_4$, $C_4 \to W_2 \times H_2 \times C_4$, $C_4 \to W_3 \times H_3 \times C_4$). The final detection features are then obtained by concatenating the point features and image features.

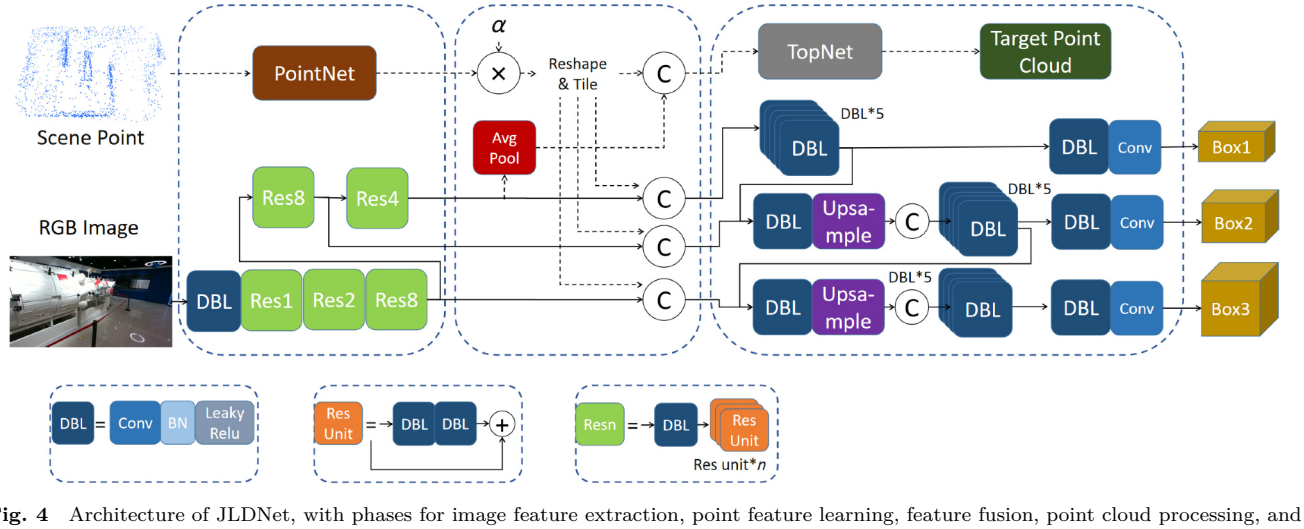

Fig. 4 Architecture of JLDNet, with phases for image feature extraction, point feature learning, feature fusion, point cloud processing, and detection decoder. Solid black lines represent the original YOLO v3 detection branch; the dashed black lines represent our additional branches.

In the point cloud regression branch, the image feature $W_3 \times H_3 \times C_3$ is converted to shape C_3 by average pooling and concatenated with the point features, thus obtaining the final localization features $(C_3 + C_4)$. Because the image and point features are learned by different networks, we use a weight α to balance them; α is also determined by the training process.

In the localization part, we use the pipeline proposed by Wang et al., which comprises point cloud generation and registration. The decoder adopts TopNet [57], which exploits a hierarchical rooted tree structure to generate point clouds; this achieves considerable performance.

In terms of network architecture, Wang et al. used DenseNet to extract image features and LSTM layers to learn pose features. In our paper, we obtain the encoder feature by combining the image feature learned by YOLO v3 with the point feature from PointNet. The feature is transformed to the point cloud decoder to regress the target point cloud.

4.3 Loss function

As explained, JLDNet outputs both detection and localization results, necessitating the design of an appropriate loss function. For detection loss, we adopt the commonly used loss that combines bounding box loss, classification loss, and object confidence loss:

$$\mathrm{Loss}_{\mathrm{box}} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{i,j}^{\mathrm{obj}} (2 - w_i - h_i) \mathrm{IOU}(b_i, \bar{b}_i) \quad (1)$$

$$\mathrm{Loss}_{\mathrm{cls}} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{i,j}^{\mathrm{obj}} \sum_{k=0}^{C} p_i(k) \log(\bar{p}_i(k)) \quad (2)$$

$$\mathrm{Loss}_{\mathrm{obj}} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} (\bar{\alpha} * \|c_i - \bar{c}_i\|^{\bar{\gamma}}) \big[c_i \log(\bar{c}_i) + (1 - c_i) \log(1 - \bar{c}_i)\big] \quad (3)$$

$$\mathrm{Loss}_{\mathrm{detect}} = \mathrm{Loss}_{\mathrm{box}} + \mathrm{Loss}_{\mathrm{cls}} + \mathrm{Loss}_{\mathrm{obj}} \quad (4)$$

where w_i and h_i denote the ground truth weight and height of the bounding box (in $[0, 1]$) respectively, and b_i and $\bar{b}_i$ represent the ground truth and predicted bounding box respectively. $p_i(k)$ and $\bar{p}_i(k)$ express the probability of class k respectively, and c_i and $\bar{c}_i$ express confidence respectively.

To measure the distance between the target and evaluated point cloud, we adopt chamfer distance (CD) loss, producing a high-quality point cloud in practice [57, 58]. To balance the two types of losses, a learned weight loss function is used following Kendall and Cipolla [15]. Overall, the loss function for JLDNet is formulated as Eqs. (5) and (6):

$$\mathrm{Loss}_{\mathrm{loc}} = \sum_{x \in S_{\mathrm{p}}} \min_{y \in S_{\mathrm{g}}} \|x - y\|_2^2 + \sum_{x \in S_{\mathrm{g}}} \min_{y \in S_{\mathrm{p}}} \|x - y\|_2^2 \quad (5)$$

$$\mathrm{Loss} = \exp(-L)\mathrm{Loss}_{\mathrm{loc}} + L + \exp(-D)\mathrm{Loss}_{\mathrm{detect}} + D \quad (6)$$

where S_{p} and S_{g} denote the prediction and ground truth set of the target point cloud respectively, and L and D are variables learned by JLDNet.

4.4 Joint refinement for localization and detection

Various SLAM systems utilize a bundle adjustment (BA) process to reduce accumulated errors [7–9, 39]. By means of JLDNet, the initial detection and localization results are obtained. In this subsection, we design an object-level BA process to further promote relocalization and detection performance. An object-level optimization process has proved effective in many works [59–61]. Aiming at different targets, various strategies cover data representation, association, and optimization metric. Huang et al. [59] introduced a semantic SDF tracker to calibrate the camera pose by semantic TSDF representation and build a global pose graph to reduce drift for globally consistent 3D reconstruction. BuildingFusion [60] regarded rooms as basic elements for global LCD to reconstruct the whole building map based on detected loop closure constraints. ObjectFusion [61] performed joint optimization for object shape, object pose, and camera pose in a sliding keyframe window.

Compared to these works, our novelty lies in two aspects. Firstly, we pay more attention to the object-level information. We mainly use a metric for bounding box points as the optimization metric, which is related to the detection task. Secondly, since the model and 3D object are known, we can jointly optimize bounding boxes and pose.

We now detail the whole BA process. Feature points, camera poses, and objects are denoted $\boldsymbol{P} = P_i$, $\boldsymbol{C} = C_i$, $\boldsymbol{O} = O_i$ respectively. BA is expressed as the nonlinear optimization problem in Eq. (7):

$$C^*, P^*, O^* = \mathrm{argmin} \sum_{C_i, P_i, O_i} (\|E(P_i, C_i)\| + \|E(O_i, C_i)\|) \quad (7)$$

where $E(O_i, C_i)$ and $E(P_i, C_i)$ denote camera-object and camera-point errors, respectively.

The camera-point error has two parts. The first concerns 3D known feature points and the current image, while the second concerns the current and previous images. Both errors are assessed as 3D point projection error in ORB SLAM [9, 62, 63] with the form in Eq. (8):

$$E(P_i, C_i) = \pi(R_{\mathrm{w2c}} P_i) - \boldsymbol{d} \tag{8}$$

where R_{w2c} is the transformation matrix from world space to camera space ($R_{\mathrm{w2c}} = R_{\mathrm{c2w}}^{-1}$), π represents the matrix from camera space to pixel space, and $\boldsymbol{d}$ denotes the pixel coordinate of the 3D point.

In addition to the camera-point error, the camera-object error represents the relation between camera poses, 3D world objects, and 2D detected objects. Based on the obtained camera pose from JLDNet and the known 3D object, we calculate the 2D bounding box expressed as its top left and bottom right 2D pixel coordinates. Then object-camera error is calculated by comparing the projected 2D bounding box with the detected 2D bounding box from JLDNet:

$$\begin{aligned} E(O_i, C_i) = &\|\pi(R_{\mathrm{w2c}} O_i(\mathrm{tl})) - O_{(2\mathrm{d})}(\mathrm{tl})\| \\ &+ \|\pi(R_{\mathrm{w2c}} O_i(\mathrm{br})) - O_{(2\mathrm{d})}(\mathrm{br})\| \end{aligned} \tag{9}$$

where tl and br denote the top left and right bottom points respectively, and $O_{(2\mathrm{d})}$ represents the 2D bounding boxes predicted by JLDNet.

Through the above BA procedure, the camera pose is optimized. Furthermore, we use the refined pose and known 3D object, the optimized 2D bounding box is obtained by Eqs. (10) and (11):

$$O^*_{(2\mathrm{d})}(\mathrm{tl}) = \bar{c}_i O_{(2\mathrm{d})}(\mathrm{tl}) + (1 - \bar{c}_i)\pi(R'_{\mathrm{w2c}} O_i(\mathrm{tl})) \tag{10}$$

$$O^*_{(2\mathrm{d})}(\mathrm{br}) = \bar{c}_i O_{(2\mathrm{d})}(\mathrm{br}) + (1 - \bar{c}_i)\pi(R'_{\mathrm{w2c}} O_i(\mathrm{br})) \tag{11}$$

where R'_{w2c} denotes the refined camera pose, and $\bar{c}_i$ is the evaluated confidence value from JLDNet.

5 Experiments

5.1 Datasets

5.1.1 *Approach*

To evaluate the performance of JLDNet, we conducted experiments on 7 Scenes, and our constructed dataset for testing static performance, and on TUM RGB-D and Bonn for evaluating dynamic localization. We further labeled 3D objects and reprojected 2D bounding boxes for testing detection performance on 7 Scenes, TUM RGB-D, and our dataset.

5.1.2 *7 Scenes*

7 Scenes was gathered using a Kinect RGB-D camera; ground truth pose was obtained by KinectFusion [64]. It comprises seven indoor environments: Chess, Fire, Heads, Pumpkin, Office, Red Kitchen, and Stairs. Since the scenes do not contain detection information, we labelled several 3D objects in each scene and generated the 2D bounding boxes by projection.

5.1.3 *TUM RGB-D*

TUM RGB-D is a large dataset that contains RGB-D and ground-truth data as a benchmark for the evaluation of visual odometry and SLAM systems. In our experiment, we selected all 9 sequences (fr3w_*, fr3s_*, and fr2_desk_with_person) from the dynamic object category. Due to a lack of detection information in the TUM RGB-D dataset, we firstly labelled the source images. As Fig. 5 shows, the static parts that include table, chair, and screen were automatically generated from labelled 3D bounding boxes and known poses. For the dynamic factors (persons walking in the scene), we combined manual marking and a deep network. The 8 sequences (fr3w_* and fr3s_*) from the dynamic object category contain 7477 images, so editing a ground truth annotation for

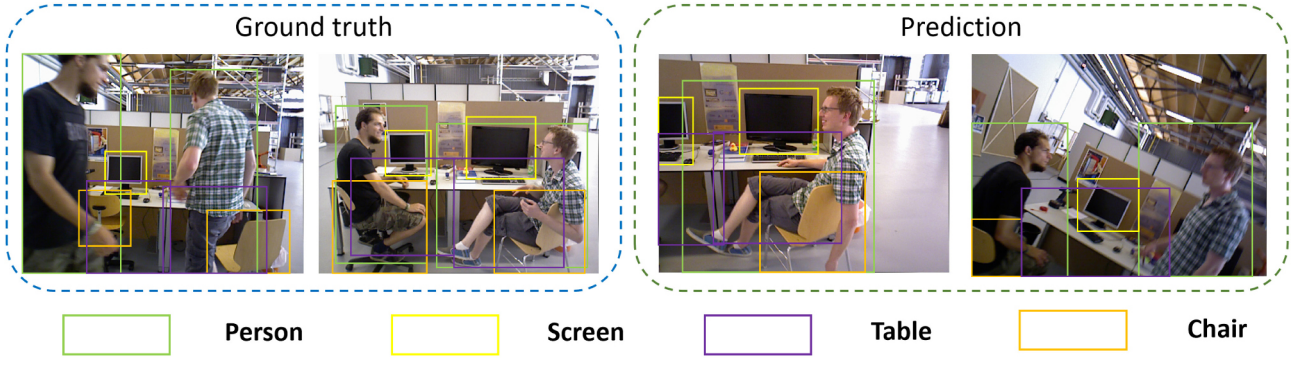

Fig. 5 Ground truth and predicted detection results for dynamic scenes from TUM RGB-D.

each image would be a time-consuming task. Instead, 400 images were selected from all the images through average sampling in the time domain, and labelled manually. Then we trained YOLO v3 with labelled images and tested on all images to obtain the results. Manual and predicted results are illustrated in Fig. 5. Because of small differences between frames, it is clear that the predicted results are quite accurate.

We performed training for other 7 sequences except for the target testing set in 8 sequences (fr3s_* and fr3s_*). When testing on fr2_desk_with_person, we used three scenes (fr2_desk, fr2_xyz, and fr2_rpy) for training. Learning difficulty increases with no dynamics in the training dataset.

5.1.4 Bonn

Bonn is a dataset for dynamic localization, containing highly dynamic sequences. For convenience of comparison to other work, we selected 20 sequences from all 24 dynamic sequences. Dataset labelling is similar to TUM RGB-D with manually labeled dynamics for one image in every 20 frames for convenience. We split the dataset into two parts. One part contains balloon, balloon2, crowd, crowd2, crowd3, moving-no-box, moving-no-box2, placing-no-box, and removing-no-box. When testing on one part, training is performed on the other part.

5.1.5 Our dataset

Our constructed dataset includes 9000 images in 9 sequences. To test localization and detection, we give two train–test splits. In split 1, the training set has 4000 images, while the other 5000 images constitute the testing set. Split 2 contains 2000 training samples and 7000 testing samples. It is obvious that split 2 is more challenging.

5.2 Implementation details

The implementation details are as follows. The input color image is resized to 256×256 pixels with a range from -1.0 to 1.0. Due to the lack of depth information in our dataset, we use grey values to replace depth values. Moreover, to ensure that the grey value is consistent with the scene extent, we multiply the grey value by 8. The output point cloud has shape 2048×3. The image encoder exploits DarkNet53 as the base encoder architecture. After DarkNet53, the multi-size features with shapes $32 \times 32 \times 256$, $16 \times 16 \times 512$, and $8 \times 8 \times 1024$ are obtained. Meanwhile, the shape of 3D bounding and outline point features is 64 through PointNet. JLDNet is trained through the ADAM optimizer [65] with a learning rate of 1×10^{-4} and batch size 8. The whole training is on a Tesla P100 GPU.

The detailed implementation of SUPER 4PCS is as follows. Following Wang et al., we run the algorithm multiple times (10 times in the experiment) to determine appropriate parameters. To accelerate running time, the number of points in SUPER 4PCS is set to a fixed value, 300. In the joint refinement stage, based on the obtained camera pose, we establish correspondences between known world points and current images, covering both feature points and object bounding box points. Meanwhile, adjacent frames are selected from previous frames and training images. Then we perform object-level refinement to calibrate the final pose and object detection.

5.3 Static localization results

Table 2 provides relocalization results on the 7 Scenes dataset, with a comparison to other hand-crafted feature, direct learning and indirect learning based methods. Position and orientation errors are reported in meters and degrees. In detail, position error is measured by Euclidean distance between ground truth and prediction, while orientation error is expressed by angle of intersection. The comparative approaches cover Active Search [38], PoseNet [15], MapNet [14], Xue et al. [17], AtLoc [66], VLocNet, KFNet, InLoc [67], Wang et al., DSAC++, DSAC*, SANet [68], Tang et al. [69], and Li et al. [70]. On 7 Scenes, JLDNet obtains comparable accuracy to the state-of-the-art techniques, KFNet and DSAC*. In detail, we achieve the best position result except for the Head and Stairs scenes and the best orientation in Red Kitchen. Although we use the coarse stage of Wang et al., there are obvious accuracy improvements in comparison to the method of Wang et al., demonstrating the superiority of our framework.

On the aviation exhibition hall, our JLDNet achieves state-of-the-art performance for both mean and median metrics, as illustrated in Table 3. We provide the median and mean results under two training–testing splits. Split 1 contains 5000 training samples and 4000 testing images, while split 2 has only 2000 training slices. Obviously, split 2 is more challenging. Apart from our method, we can see that KFNet achieves the best results, by establishing

Table 2 Relocalization errors on 7 Scenes for our method and various state-of-the-art methods. Units of position and orientation are m and °, respectively. In each case, the most accurate result is in bold

	Chess	Fire	Heads	Office	Pumpkin	Kitchen	Stairs	Mean
Hand-crafted feature								
Active search	0.04/1.96	0.03/1.53	0.02/1.45	0.09/3.61	0.08/3.10	0.07/3.37	0.03/2.22	0.051/2.46
Direct learning								
PoseNet	0.32/8.12	0.47/14.4	0.29/12.0	0.48/7.68	0.47/8.42	0.59/8.64	0.47/13.8	0.44/10.44
Xue et al.	0.09/3.25	0.26/10.92	0.17/12.7	0.18/5.45	0.20/3.66	0.23/4.92	0.23/11.3	0.194/7.46
MapNet	0.08/3.25	0.27/11.7	0.18/13.3	0.17/5.15	0.22/4.02	0.23/4.93	0.30/12.1	0.207/7.78
Atloc	0.10/3.18	0.26/10.8	0.14/11.4	0.17/5.16	0.20/3.94	0.16/4.90	0.29/10.20	0.189/7.08
VLocNet	0.036/1.71	0.039/5.34	0.046/6.64	0.039/1.95	0.037/2.28	0.039/2.20	0.097/6.48	0.048/3.80
Indirect learning								
DSAC++	0.02/**0.5**	0.02/0.9	**0.01/0.8**	0.03/**0.7**	0.04/1.1	0.04/1.1	0.09/2.6	0.036/1.0
DSAC*	0.019/1.11	0.019/1.24	0.011/1.82	0.026/1.18	0.042/1.41	0.03/1.70	0.041/1.42	0.027/1.41
KFNet	0.018/0.65	0.023/0.90	0.014/0.82	0.025/0.69	0.037/1.02	0.038/1.16	0.033/0.94	0.027/**0.88**
InLoc	0.03/1.05	0.03/1.07	0.02/1.16	0.03/1.05	0.05/1.55	0.04/1.31	0.09/2.47	0.041/1.38
SANet	0.03/0.88	0.03/1.08	0.02/1.48	0.03/1.00	0.05/1.32	0.04/1.40	0.16/4.59	0.051/1.68
Tang et al.	0.02/0.71	0.02/**0.85**	**0.01**/0.85	0.03/0.84	0.04/1.16	0.04/1.17	0.05/1.33	0.03/0.99
Li et al.	0.02/0.7	0.02/0.9	**0.01**/0.9	0.03/0.8	0.04/**1.0**	0.04/1.2	**0.03/0.8**	0.03/0.9
Wang et al. (coarse stage)	0.021/0.85	0.028/1.42	0.028/1.45	0.043/1.37	0.045/1.17	0.042/1.65	0.048/1.21	0.036/1.30
Ours (no detection)	0.025/1.29	0.027/1.54	0.019/1.82	0.033/1.57	0.049/2.03	0.047/1.65	0.056/2.24	0.037/1.73
Ours (no fusion block)	0.016/0.87	**0.018**/1.22	0.013/1.42	0.024/1.06	0.039/1.32	0.036/1.01	0.037/1.44	0.026/1.19
Ours (no refinement)	0.017/0.91	0.020/1.38	0.014/1.67	0.029/1.46	0.045/1.73	0.041/1.45	0.046/1.79	0.030/1.48
Ours	**0.014**/0.76	**0.018**/1.05	0.012/1.17	**0.021**/0.91	**0.036**/1.11	**0.035/0.95**	0.032/1.25	**0.024**/1.03

Table 3 Median localization errors on our constructed dataset for our method and other state-of-the-art methods. Units of position and orientation are m and °, respectively. In each case, the most accurate result is in bold

	Split 1 (mean)	Split 1 (median)	Split 2 (mean)	Split 2 (median)
PoseNet	0.28/4.62	0.14/2.98	0.36/5.77	0.17/3.98
VLocNet	0.25/4.26	0.11/2.65	0.32/6.08	0.16/3.37
DSAC*	0.17/2.72	0.09/1.99	0.21/3.76	0.14/2.65
KFNet	0.16/2.42	0.08/1.69	0.19/3.67	0.14/2.68
Ours (no detection)	0.18/3.27	0.11/2.42	0.24/4.13	0.15/3.12
Ours (no fusion block)	0.13/2.15	0.07/1.28	0.15/2.55	0.11/1.94
Ours (no refinement)	0.15/2.35	0.08/1.59	0.17/3.05	0.13/2.26
Ours	**0.12/1.95**	**0.06/1.19**	**0.14/2.36**	**0.09/1.78**

accurate correspondences by Kalman filtering, which is a strong competitor. Compared to KFNet, the median results are improved by 0.02 m, 0.5° on split 1 and by 0.03 m, 0.9° on split 2, while mean performance improves by 0.04 m, 0.47° and 0.05 m, 1.31° respectively. Figure 6 shows some results for the testing sequence. It is clear that both position and orientation have small errors, suitable for AR applications.

In conclusion, the improvements over JLDNet come from multi-task learning and joint refinement. Because of the sharing of contextual information, detection and relocalization promote each other. Moreover, the object-level BA refinement combines both low-level point features and high-level object information, further improving accuracy.

For mobile applications, computational performance is an important consideration. Our JLDNet takes about 45 ms per image for the whole process on a Tesla P100 GPU. In actual applications, JLDNet runs on a server and provides real-time detection and relocalization services. Figure 7 demonstrates an AR guide program based on the results from JLDNet.

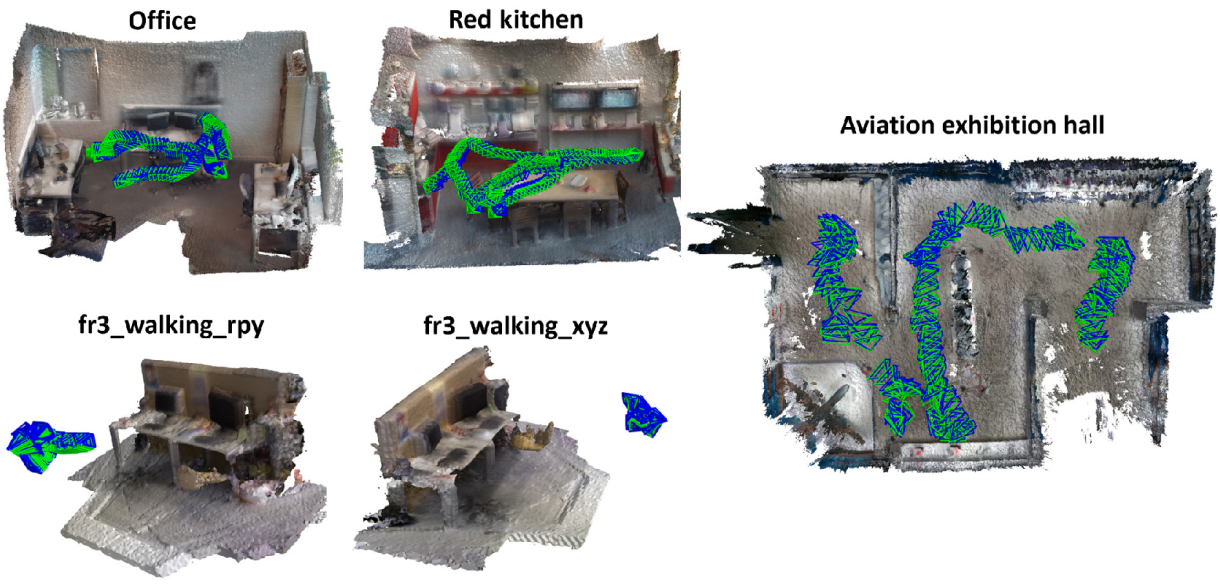

Fig. 6 Visualization of ground truth and prediction camera pose. Green and blue camera poses denote the ground truth and prediction respectively.

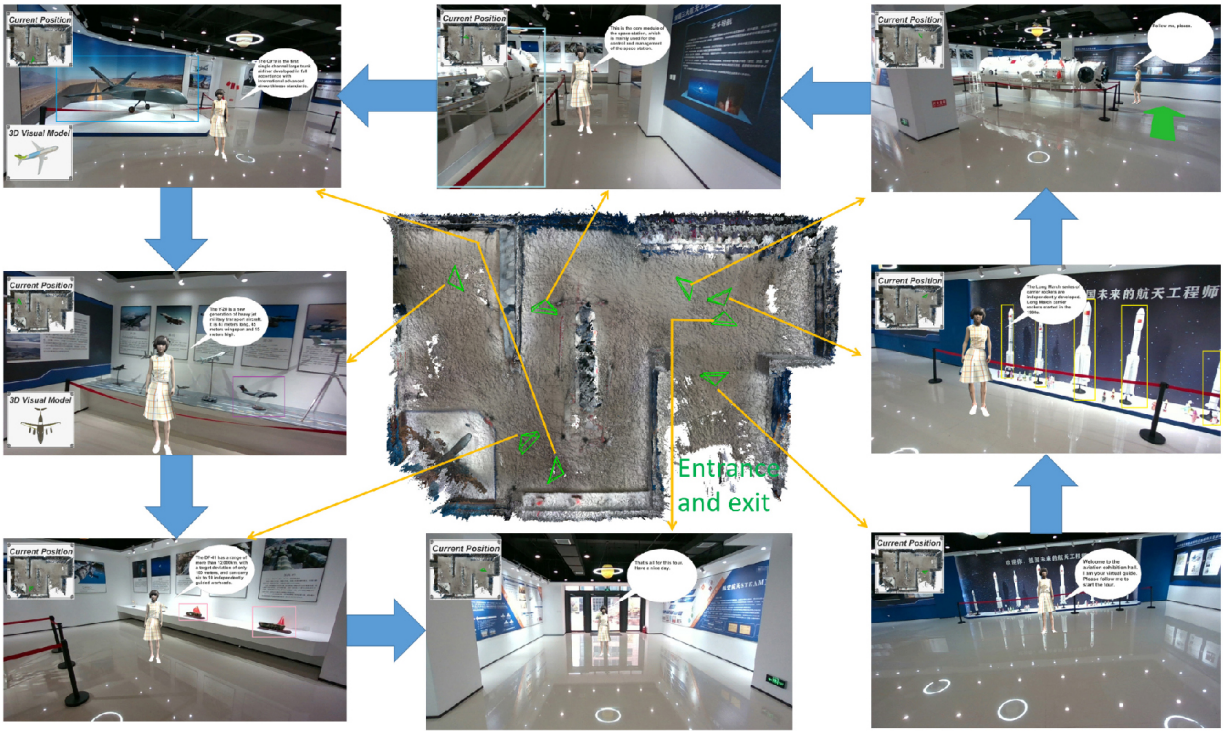

Fig. 7 An AR guide program based on JLDNet. The center shows the 3D scene and camera poses. Other eight images show the actual effect of the program. Each running result contains 3D position (top left), virtual guide, introduction UI, detected object, navigation information, and 3D virtual model (bottom left).

5.4 Dynamic localization results

Table 4 gives ATE RMSE and RPE RMSE results for all 9 TUM RGB-D dynamic scenes, making a comparison to ORB-SLAM2 [63], DS-SLAM [39], DynaSLAM [12], MaskFusion [40], and LC-CRF SLAM [71]. It is clear that our mean RMSE results outperform other SLAM systems. In detail, we achieve the best results in 7 Scenes on ATE and 7 Scenes on RPE, which shows state-of-the-art performance. When testing on fr2_desk_with_person, we use three scenes (fr2_desk, fr2_xyz, and fr2_rpy). In particular, learning difficulty is increased by

lack of dynamics in the training dataset. Again, we obtain the most accurate result for this scene. Moreover, the additional detection branch and joint refinement enhance the localization accuracy in dynamic environments, as demonstrated by the last two rows of Table 4.

In Table 5, we report experiments on the Bonn dataset and make a comparison to ReFusion [36], MaskFusion, and LC-CRF SLAM. We split the datasets into two parts. One part contains balloon, balloon2, crowd, crowd2, crowd3, moving-no-box, moving-no-box2, placing-no-box, and removing-no-

Table 4 ATE and RPE RMSE results compared to some SLAM systems for dynamic scenes in TUM RGB-D. In each scene, the top accurate result is marked bold

	fr3w_xyz	fr3w_static	fr3w_rpy	fr3w_half	fr3w_xyz	fr3s_static	fr3s_rpy	fr3s_half	fr2_desk_with_person
ATE (m)									
ORB-SLAM2 (RGB-D)	0.459	0.090	0.662	0.351	0.011	0.009	0.044	0.020	0.074
DynaSLAM	0.015	**0.006**	0.035	0.025	0.015	—	—	0.017	—
DS-SLAM	0.025	0.008	0.444	0.030	—	**0.007**	—	—	—
MaskFusion	0.104	0.035	—	0.106	0.031	0.021	—	0.052	
LC-CRF SLAM	0.016	0.011	0.046	0.028	**0.009**	—	—	—	0.069
Ours (no detection)	0.021	0.011	0.032	0.033	0.018	0.012	0.024	0.018	0.066
Ours	**0.013**	0.007	**0.018**	**0.019**	0.014	**0.007**	**0.019**	**0.013**	**0.057**
RPE (m/s)									
ORB-SLAM2 (RGB-D)	0.412	0.216	0.424	0.355	0.016	0.011	0.039	0.024	0.104
DS-SLAM	0.033	0.010	0.150	0.030	—	**0.008**	—	—	—
MaskFusion	0.097	0.039	—	0.093	0.046	0.017	—	0.041	
LC-CRF SLAM	0.021	0.014	0.050	0.035	**0.012**	—	—	—	0.086
Ours (no detection)	0.022	0.011	0.036	0.042	0.024	0.014	0.029	0.021	0.081
Ours	**0.016**	**0.009**	**0.027**	**0.026**	0.017	0.010	**0.023**	**0.017**	**0.069**

Table 5 ATE and RPE RMSE results compared to some SLAM systems on TUM RGB-D dynamic scenes. For each scene, the most accurate result is marked bold. RF = ReFusion, MF = MaskFusion

	ATE (m)				RPE (m/s)			
	RF	MF	LC-CRF	Ours	RF	MF	LC-CRF	Ours
balloon	0.175	0.165	0.027	**0.021**	0.576	0.509	0.612	**0.322**
balloon2	0.254	0.114	0.024	**0.022**	0.540	0.499	0.541	**0.439**
balloon-tracking	0.302	0.194	0.025	**0.019**	1.031	0.991	0.965	**0.856**
balloon-tracking2	0.322	0.238	0.045	**0.029**	1.059	0.937	0.935	**0.821**
crowd	0.204	0.473	**0.019**	0.022	0.198	0.633	**0.238**	0.326
crowd2	0.155	0.653	0.031	**0.030**	0.315	0.854	**0.199**	0.231
crowd3	0.137	0.341	0.023	**0.019**	0.223	0.503	**0.194**	0.258
kidnapping-box	0.148	0.200	0.023	**0.017**	0.886	0.840	1.001	**0.792**
kidnapping-box2	0.161	0.182	**0.020**	0.031	1.077	1.027	1.184	**0.943**
moving-no-box	0.071	0.120	**0.018**	0.024	0.939	0.947	0.936	**0.879**
moving-no-box2	0.179	0.193	0.038	**0.025**	1.287	1.252	1.399	**1.132**
moving-o-box	0.343	0.216	0.253	**0.195**	1.274	**0.847**	1.158	0.977
moving-o-box2	0.528	0.298	0.341	**0.285**	1.523	**0.576**	1.143	0.866
person-tracking	0.289	0.301	0.035	**0.034**	1.209	1.312	1.193	**1.023**
person-tracking2	0.463	0.220	**0.040**	0.046	1.165	1.267	1.297	**0.971**
placing-no-box	0.106	0.325	**0.014**	0.019	0.355	0.598	**0.333**	0.422
placing-no-box2	0.141	0.153	**0.016**	0.025	0.282	0.330	0.271	**0.215**
placing-no-box3	0.174	0.156	0.036	**0.029**	0.511	0.491	0.482	**0.422**
placing-o-box	0.571	0.424	0.320	**0.264**	1.180	0.791	**0.505**	0.740
removing-no-box	0.041	0.058	**0.013**	0.025	0.262	0.263	0.240	**0.216**
Mean	0.238	0.251	0.068	**0.058**	0.795	0.773	0.741	**0.643**

box. When testing on one part, training is performed on the other part. Our framework achieves the best mean results on RPE and ATE metrics. Meanwhile, we also obtain state-of-the-art results for most scenes. The reason for accuracy improvement is located in the joint learning and refinement between detection and localization, which promotes pose estimation.

5.5 Detection results

To use the dataset for evaluation, our method requires both localization and detection ground truth. Moreover, the testing set should be the same scene as the training set. There are few common detection datasets satisfying the above requirements. MS COCO does not contain localization information, while the training and testing scenes of ScanNet are different, so are not appropriate for our method.

To validate the detection performance of our framework, we conducted experiments on 7 Scenes, TUM RGB-D, and our aviation exhibition dataset. The labelling process and train–test split are discussed above. The results of mean average precision under IoU threshold 0.5 (mAP@0.5) are presented in Table 6. JLDNet achieves more accurate detection results in all scenes than YOLO v3. The mean improvements are 3.8% on 7 Scenes, 3.9% on TUM RGB-D, and 1.7 % on our dataset, demonstrating the state-of-the-art detection results provided by our method. Moreover, the positive roles of the fusion block and refinement stage are also illustrated. On TUM RGB-D, we can see that the mean results for the fr3s_* four scenes are more accurate than for the other four fr3w_* scenes. This is because the four highly dynamic fr3w_* scenes contain more dynamic persons than the less dynamic four fr3s_* scenes, which increases the learning difficulty.

Leading to the accuracy improvement, for one thing, the additional point cloud and feature fusion parts make JLDNet learn more features. For another, the localization branch that is also related to detection promotes object regression. Moreover, joint refinement of camera poses and object bounding boxes further improves both branches.

5.6 Detailed studies

The above experiments demonstrate state-of-the-art results on both static and dynamic environments. In our opinion, the improvement is due to multi-task learning of shared context information, the fusion block for balancing two branches, and joint refinement that combines point feature and object-level information.

In this subsection, we discuss the effects of these three parts, stated in last four columns of Tables 2 and 3 and last two rows of Table 4. Localization accuracy clearly decreases when we remove the fusion, refinement, or detection stage, which indicates the positive effect of these three parts for both 7 Scenes and the aviation exhibition hall. It is also noticeable that accuracy suffers the biggest degradation without the detection branch, as feature fusion and joint refinement cannot work without the detection part.

Table 7 provides localization and detection results using different YOLO networks, covering YOLO, YOLO v2 [34], YOLO v3, YOLO v4 [33], and YOLO v5 [54]. Compared to YOLO and YOLO v2, our

Table 6 mAP@0.5 results compared to YOLO v3 on 7 Scenes, TUM RGB-D, and our constructed dataset

7 Scenes									
	Chess	Fire	Heads	Office	Pumpkin	Kitchen	Stairs	Mean	
YOLO v3	87.3%	82.4%	86.1%	85.4%	71.2%	78.3%	74.7%	80.8%	
Ours	91.2%	86.9%	89.4%	90.4%	74.1%	82.3%	78.2%	84.6%	
TUM RGB-D									
	fr3w_xyz	fr3w_static	fr3w_rpy	fr3w_half	fr3s_xyz	fr3s_static	fr3s_rpy	fr3s_half	Mean
YOLO v3	81.3%	83.2%	80.1%	81.1%	88.7%	87.6%	83.3%	85.1%	83.8%
Ours	85.4%	87.9%	84.2%	85.6%	91.1%	90.3%	87.4%	89.3%	87.7%
Our dataset									
	Split 1	Split 2							
YOLO v3	90.6%	86.0%							
Ours (no fusion block)	91.9%	88.2%							
Ours (no refinement)	91.1%	86.8%							
Ours	92.3%	88.7%							

Table 7 Localization and detection results using different Yolo structures on 7 Scenes

	Localization	Detection (mAP)
YOLO	0.026/1.41	84.9%
YOLO v2	0.025/1.23	85.1%
YOLO v3	0.024/**1.03**	85.7%
YOLO v4	0.024/1.09	**85.9%**
YOLO v5	**0.023**/1.13	85.6%

method (YOLO v3) achieves a little improvement; results are comparable to those of YOLO v4 and YOLO v5. In our opinion, this is for two reasons. Firstly, the training and testing datasets always have many similar samples, which differ from those in common detection datasets. Secondly, the post refinement stage may reduce network errors.

5.7 Limitations and future work

The above experiments show superior results for both localization and detection tasks. However, the limitation of our framework lies in its scene dependence. Before performing localization and object detection, we need training samples with pose and bounding box labels for the target scene. In future, we intend to focus on scene-independent localization, by considering relative pose estimation and detection constraints between consecutive frames.

6 Conclusions

This paper addresses the challenges of joint detection and localization. We also investigate dataset construction, multi-task learning network design, and joint refinement. First of all, we exploit a pipeline based on SFM, MVS, 3D manual marking, and 2D label projection to construct a medium-sized indoor scene of an aviation exhibition hall. The dataset contains 9000 images in total. Each slice provides a color image, camera pose, and detected object information. Moreover, paying attention to multi-task learning of detection and localization, we propose a deep network called JLDNet, composed of image feature extraction, point feature learning, feature fusion, detection output, and point cloud regression. To optimize JLDNet, we design a loss function to balance the two branches with learned coefficients. Based on the results of JLDNet, an object-level refinement process further improves the accuracy of both tasks. To test JLDNet, we conducted experiments on 7 Scenes, our constructed dataset, TUM RGB-D and Bonn datasets, making comparisons to other methods, for both static and dynamic scenes. The localization and detection results demonstrate state-of-the-art accuracy and the promotion effects of the two tasks.

Acknowledgements

We would like to thank the anonymous reviewers for their constructive comments.

Funding

This paper was supported by the National Natural Science Foundation of China (No. 62072020), and Key-Area Research and the Leading Talents in Innovation and Entrepreneurship of Qingdao (No. 19-3-2-21-zhc).

Data availability statement

The datasets generated during the current study are available from `https://drive.google.com/drive/folders/1U28zkON4_IOdbzkqyIAKlAl5k9oUKOjI?usp=sharing`.

Author contributions

Conceptualization, J.W. and Y.Q.; methodology, J.W. and Y.Q.; software, J.W.; validation, J.W. and Y.Q.; formal analysis, J.W.; investigation, J.W.; resources, Y.Q.; data curation, Y.Q.; writing—original draft preparation, J.W.; writing—review and editing, Y.Q.; visualization, J.W.; supervision, Y.Q.; project administration, Y.Q.; funding acquisition, Y.Q. All authors have read and agreed to the published version of the manuscript.

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] Bao, W.; Wang, W.; Xu, Y. H.; Guo, Y. L.; Hong, S. Y.; Zhang, X. H. InStereo2K: A large real dataset for stereo matching in indoor scenes. *Science China Information Sciences* Vol. 63, No. 11, 212101, 2020.

[2] Yan, F. H.; Li, Z. X.; Zhou, Z. Robust and efficient edge-based visual odometry. *Computational Visual Media* Vol. 8, No. 3, 467–481, 2022.

[3] Huang, J. H.; Yang, S.; Zhao, Z. S.; Lai, Y. K.; Hu, S. M. ClusterSLAM: A SLAM backend for simultaneous rigid body clustering and motion estimation. *Computational Visual Media* Vol. 7, No. 1, 87–101, 2021.

[4] Wang, C.; Guo, X. H. Feature-based RGB-D camera pose optimization for real-time 3D reconstruction. *Computational Visual Media* Vol. 3, No. 2, 95–106, 2017.

[5] Nakajima, Y.; Saito, H. Robust camera pose estimation by viewpoint classification using deep learning. *Computational Visual Media* Vol. 3, No. 2, 189–198, 2017.

[6] Liu, S.; Zhang, Y. Q.; Yang, X. S.; Shi, D. M.; Zhang, J. J. Robust facial landmark detection and tracking across poses and expressions for in-the-wild monocular video. *Computational Visual Media* Vol. 3, No. 1, 33–47, 2017.

[7] Qin, T.; Li, P. L.; Shen, S. J. VINS-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics* Vol. 34, No. 4, 1004–1020, 2018.

[8] Rublee, E.; Rabaud, V.; Konolige, K.; Bradski, G. ORB: An efficient alternative to SIFT or SURF. In: Proceedings of the International Conference on Computer Vision, 2564–2571, 2011.

[9] Campos, C.; Elvira, R.; Rodriguez, J. J. G.; Montiel, J. M. M.; Tardos, J. D. ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM. *IEEE Transactions on Robotics* Vol. 37, No. 6, 1874–1890, 2021.

[10] Xu, B. B.; Li, W. B.; Tzoumanikas, D.; Bloesch, M.; Davison, A.; Leutenegger, S. MID-fusion: Octree-based object-level multi-instance dynamic SLAM. In: Proceedings of the International Conference on Robotics and Automation, 5231–5237, 2019.

[11] Yang, S. C.; Scherer, S. CubeSLAM: Monocular 3-D object SLAM. *IEEE Transactions on Robotics* Vol. 35, No. 4, 925–938, 2019.

[12] Bescos, B.; Facil, J. M.; Civera, J.; Neira, J. DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters* Vol. 3, No. 4, 4076–4083, 2018.

[13] Kendall, A.; Grimes, M.; Cipolla, R. PoseNet: A convolutional network for real-time 6-DOF camera relocalization. In: Proceedings of the IEEE International Conference on Computer Vision, 2938–2946, 2015.

[14] Brahmbhatt, S.; Gu, J. W.; Kim, K.; Hays, J.; Kautz, J. Geometry-aware learning of maps for camera localization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2616–2625, 2018.

[15] Kendall, A.; Cipolla, R. Geometric loss functions for camera pose regression with deep learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 6555–6564, 2017.

[16] Walch, F.; Hazirbas, C.; Leal-Taixe, L.; Sattler, T.; Hilsenbeck, S.; Cremers, D. Image-based localization using LSTMs for structured feature correlation. In: Proceedings of the IEEE International Conference on Computer Vision, 627–637, 2017.

[17] Xue, F.; Wang, X.; Yan, Z. K.; Wang, Q. Y.; Wang, J. Q.; Zha, H. B. Local supports global: Deep camera relocalization with sequence enhancement. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2841–2850, 2019.

[18] Kendall, A.; Cipolla, R. Modelling uncertainty in deep learning for camera relocalization. In: Proceedings of the IEEE International Conference on Robotics and Automation, 4762–4769, 2016.

[19] Radwan, N.; Valada, A.; Burgard, W. VLocNet++: Deep multitask learning for semantic visual localization and odometry. *IEEE Robotics and Automation Letters* Vol. 3, No. 4, 4407–4414, 2018.

[20] Cavallari, T.; Golodetz, S.; Lord, N. A.; Valentin, J.; Di Stefano, L.; Torr, P. H. S. On-the-fly adaptation of regression forests for online camera relocalisation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 218–227, 2017.

[21] Schmidt, T.; Newcombe, R.; Fox, D. Self-supervised visual descriptor learning for dense correspondence. *IEEE Robotics and Automation Letters* Vol. 2, No. 2, 420–427, 2017.

[22] Brachmann, E.; Michel, F.; Krull, A.; Yang, M. Y.; Gumhold, S.; Rother, C. Uncertainty-driven 6D pose estimation of objects and scenes from a single RGB image. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 3364–3372, 2016.

[23] Guzman-Rivera, A.; Kohli, P.; Glocker, B.; Shotton, J.; Sharp, T.; Fitzgibbon, A.; Izadi, S. Multi-output learning for camera relocalization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1114–1121, 2014.

[24] Shotton, J.; Glocker, B.; Zach, C.; Izadi, S.; Criminisi, A.; Fitzgibbon, A. Scene coordinate regression forests for camera relocalization in RGB-D images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2930–2937, 2013.

[25] Valentin, J.; Niebner, M.; Shotton, J.; Fitzgibbon,

A.; Izadi, S.; Torr, P. Exploiting uncertainty in regression forests for accurate camera relocalization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4400–4408, 2015.

[26] Dong, S. Y.; Fan, Q. N.; Wang, H.; Shi, J.; Yi, L.; Funkhouser, T.; Chen, B. Q.; Guibas, L. Robust neural routing through space partitions for camera relocalization in dynamic indoor environments. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 8540–8550, 2021.

[27] Brachmann, E.; Rother, C. Visual camera relocalization from RGB and RGB-D images using DSAC. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 44, No. 9, 5847–5865, 2022.

[28] Zhou, L.; Luo, Z. X.; Shen, T. W.; Zhang, J. H.; Zhen, M. M.; Yao, Y.; Fang, T.; Quan, L. KFNet: Learning temporal camera relocalization using Kalman filtering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 4918–4927, 2020.

[29] Cavallari, T.; Golodetz, S.; Lord, N. A.; Valentin, J.; Prisacariu, V. A.; Stefano, L. D.; Torr, P. H. S. Real-time RGB-D camera pose estimation in novel scenes using a relocalisation cascade. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 42, No. 10, 2465–2477, 2020.

[30] Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 779–788, 2016.

[31] Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv preprint* arXiv:1804.02767, 2018.

[32] Kim, S.; Park, S.; Na, B.; Yoon, S. Spiking-YOLO: Spiking neural network for energy-efficient object detection. *Proceedings of the AAAI Conference on Artificial Intelligence* Vol. 34, No. 7, 11270–11277, 2020.

[33] Bochkovskiy, A.; Wang, C. Y.; Liao, H. Y. M. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint* arXiv:2004.10934, 2020.

[34] Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 6517–6525, 2017.

[35] Sturm, J.; Engelhard, N.; Endres, F.; Burgard, W.; Cremers, D. A benchmark for the evaluation of RGB-D SLAM systems. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 573–580, 2012.

[36] Palazzolo, E.; Behley, J.; Lottes, P.; Giguère, P.; Stachniss, C. ReFusion: 3D reconstruction in dynamic environments for RGB-D cameras exploiting residuals. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 7855–7862, 2019.

[37] Liu, L.; Li, H. D.; Dai, Y. C. Efficient global 2D–3D matching for camera localization in a large-scale 3D map. In: Proceedings of the IEEE International Conference on Computer Vision, 2391–2400, 2017.

[38] Sattler, T.; Leibe, B.; Kobbelt, L. Efficient & effective prioritized matching for large-scale image-based localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 39, No. 9, 1744–1756, 2017.

[39] Yu, C.; Liu, Z. X.; Liu, X. J.; Xie, F. G.; Yang, Y.; Wei, Q.; Fei, Q. DS-SLAM: A semantic visual SLAM towards dynamic environments. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 1168–1174, 2018.

[40] Runz, M.; Buffier, M.; Agapito, L. MaskFusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In: Proceedings of the IEEE International Symposium on Mixed and Augmented Reality, 10–20, 2018.

[41] Valada, A.; Radwan, N.; Burgard, W. Deep auxiliary learning for visual localization and odometry. In: Proceedings of the IEEE International Conference on Robotics and Automation, 6939–6946, 2018.

[42] Balntas, V.; Li, S. D.; Prisacariu, V. RelocNet: Continuous metric learning relocalisation using neural nets. In: *Computer Vision – ECCV 2018. Lecture Notes in Computer Science, Vol. 11218*. Ferrari, V.; Hebert, M.; Sminchisescu, C.; Weiss, Y. Eds. Springer Cham, 782–799, 2018.

[43] Melekhov, I.; Ylioinas, J.; Kannala, J.; Rahtu, E. Image-based localization using hourglass networks. In: Proceedings of the IEEE International Conference on Computer Vision Workshops, 870–877, 2017.

[44] Laskar, Z.; Melekhov, I.; Kalia, S.; Kannala, J. Camera relocalization by computing pairwise relative poses using convolutional neural network. In: Proceedings of the IEEE International Conference on Computer Vision Workshops, 920–929, 2017.

[45] Brachmann, E.; Rother, C. Learning less is more—6D camera localization via 3D surface regression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 4654–4662, 2018.

[46] Wang, J. Y.; Qi, Y. Camera relocalization using deep point cloud generation and hand-crafted feature

refinement. In: Proceedings of the IEEE International Conference on Robotics and Automation, 5891–5897, 2021.

[47] Brachmann, E.; Krull, A.; Nowozin, S.; Shotton, J.; Michel, F.; Gumhold, S.; Rother, C. DSAC—Differentiable RANSAC for camera localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2492–2500, 2017.

[48] Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 580–587, 2014.

[49] Girshick, R. Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, 1440–1448, 2015.

[50] Lan, Y. Q.; Duan, Y.; Liu, C. Y.; Zhu, C. Y.; Xiong, Y. S.; Huang, H.; Xu, K. ARM3D: Attention-based relation module for indoor 3D object detection. *Computational Visual Media* Vol. 8, No. 3, 395–414, 2022.

[51] Ren, S. Q.; He, K. M.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 39, No. 6, 1137–1149, 2017.

[52] He, K. M.; Gkioxari, G.; Dollar, P.; Girshick, R. Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, 2980–2988, 2017.

[53] Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C. Y.; Berg, A. C. SSD: Single shot MultiBox detector. In: *Computer Vision – ECCV 2016. Lecture Notes in Computer Science, Vol. 9905.* Leibe, B.; Matas, J.; Sebe, N.; Welling, M. Eds. Springer Cham, 21–37, 2016.

[54] Jocher, G. Yolo v5. 2020. Available at https://github.com/ultralytics/yolov5

[55] Dai, A.; Chang, A. X.; Savva, M.; Halber, M.; Funkhouser, T.; Niessner, M. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2432–2443, 2017.

[56] Charles, R. Q.; Hao, S.; Mo, K. C.; Guibas, L. J. PointNet: Deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 77–85, 2017.

[57] Tchapmi, L. P.; Kosaraju, V.; Rezatofighi, H.; Reid, I.; Savarese, S. TopNet: Structural point cloud decoder. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 383–392, 2019.

[58] Fan, H. Q.; Su, H.; Guibas, L. A point set generation network for 3D object reconstruction from a single image. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2463–2471, 2017.

[59] Huang, S. S.; Chen, H. X.; Huang, J. H.; Fu, H. B.; Hu, S. M. Real-time globally consistent 3D reconstruction with semantic priors. *IEEE Transactions on Visualization and Computer Graphics* Vol. 29, No. 4, 1977–1991, 2023.

[60] Zheng, T.; Zhang, G. Q.; Han, L.; Xu, L.; Fang, L. Building fusion: Semantic-aware structural building-scale 3D reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 44, No. 5, 2328–2345, 2022.

[61] Zou, Z. X.; Huang, S. S.; Mu, T. J.; Wang, Y. P. ObjectFusion: Accurate object-level SLAM with neural object priors. *Graphical Models* Vol. 123, 101165, 2022.

[62] Mur-Artal, R.; Montiel, J. M. M.; Tardos, J. D. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics* Vol. 31, No. 5, 1147–1163, 2015.

[63] Mur-Artal, R.; Tardós, J. D. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics* Vol. 33, No. 5, 1255–1262, 2017.

[64] Newcombe, R. A.; Izadi, S.; Hilliges, O.; Molyneaux, D.; Kim, D.; Davison, A. J.; Kohli, P.; Shotton, J.; Hodges, S.; Fitzgibbon, A. KinectFusion: Real-time dense surface mapping and tracking. In: Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality, 127–136, 2011.

[65] Kingma, D. P.; Ba, J. Adam: A method for stochastic optimization. *arXiv preprint* arXiv:1412.6980, 2014.

[66] Wang, B.; Chen, C. H.; Xiaoxuan Lu, C.; Zhao, P. J.; Trigoni, N.; Markham, A. AtLoc: Attention guided camera localization. *Proceedings of the AAAI Conference on Artificial Intelligence* Vol. 34, No. 6, 10393–10401, 2020.

[67] Taira, H.; Okutomi, M.; Sattler, T.; Cimpoi, M.; Pollefeys, M.; Sivic, J.; Pajdla, T.; Torii, A. InLoc: Indoor visual localization with dense matching and view synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 43, No. 4, 1293–1307, 2021.

[68] Yang, L. W.; Bai, Z. Q.; Tang, C. Z.; Li, H. H.; Furukawa, Y.; Tan, P. SANet: Scene agnostic network for camera localization. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 42–51, 2019.

[69] Tang, S. T.; Tang, C. Z.; Huang, R.; Zhu, S. Y.; Tan, P. Learning camera localization via dense scene matching. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 1831–1841, 2021.

[70] Li, X. T.; Wang, S. Z.; Zhao, Y.; Verbeek, J.; Kannala, J. Hierarchical scene coordinate classification and regression for visual localization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 11980–11989, 2020.

[71] Du, Z. J.; Huang, S. S.; Mu, T. J.; Zhao, Q. H.; Martin, R. R.; Xu, K. Accurate dynamic SLAM using CRF-based long-term consistency. *IEEE Transactions on Visualization and Computer Graphics* Vol. 28, No. 4, 1745–1757, 2022.

Junyi Wang is a Ph.D. candidate at State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, working under the supervision of Prof. Yue Qi. His research is in camera localization, deep object pose estimation, and 3D reconstruction.

Yue Qi is a professor at State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University. He was a senior visiting scholar at Harvard University. His research covers virtual reality, augmented reality algorithms and applications, human–computer interaction, computer graphics theory and methods, computer vision algorithms and applications.