Real-Time Dense Monocular SLAM With Online Adapted Depth Prediction Network

Hongcheng Luo, Yang Gao, Yuhao Wu, Chunyuan Liao, Xin Yang, *Member, IEEE*, and Kwang-Ting Cheng, *Fellow, IEEE*

***Abstract*—Considerable advances have been achieved in estimating the depth map from a single image via convolutional neural networks (CNNs) during the past few years. Combining depth prediction from CNNs with conventional monocular simultaneous localization and mapping (SLAM) is promising for accurate and dense monocular reconstruction, in particular addressing the two long-standing challenges in conventional monocular SLAM: low map completeness and scale ambiguity. However, depth estimated by pretrained CNNs usually fails to achieve sufficient accuracy for environments of different types from the training data, which are common for certain applications such as obstacle avoidance of drones in unknown scenes. Additionally, inaccurate depth prediction of CNN could yield large tracking errors in monocular SLAM. In this paper, we present a real-time dense monocular SLAM system, which effectively fuses direct monocular SLAM with an online-adapted depth prediction network for achieving accurate depth prediction of scenes of different types from the training data and providing absolute scale information for tracking and mapping. Specifically, on one hand, tracking pose (i.e., translation and rotation) from direct SLAM is used for selecting a small set of highly effective and reliable training images, which acts as ground truth for tuning the depth prediction network on-the-fly toward better generalization ability for scenes of different types. A stage-wise Stochastic Gradient Descent algorithm with a selective update strategy is introduced for efficient convergence of the tuning process. On the other hand, the dense map produced by the adapted network is applied to address scale ambiguity of direct monocular SLAM which in turn improves the accuracy of both tracking and overall reconstruction. The system with assistance of both CPUs and GPUs, can achieve real-time performance with progressively improved reconstruction accuracy. Experimental results on public datasets and live application to obstacle avoidance of drones demonstrate that our method outperforms the state-of-the-art methods with greater map completeness and accuracy, and a smaller tracking error.**

***Index Terms*—Monocular SLAM, dense mapping, convolutional neural network, fusion, online tuning.**

Manuscript received December 16, 2017; revised May 12, 2018 and June 19, 2018; accepted July 5, 2018. Date of publication July 24, 2018; date of current version January 24, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61502188, in part by the Wuhan Science and Technology Bureau under Award 2017010201010111, and in part by the Program for HUST Acadamic Frontier Youth Team. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Cha Zhang. *(Corresponding author: Xin Yang.)*

H. Luo, Y. Gao, Y. Wu, and X. Yang are with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: hongcheng@hust.edu.cn; 503913389@qq.com; 1628422844@qq.com; xinyang2014@hust.edu.cn).

C. Liao is with HiScene Information Technology Co., Ltd, Pudong 201210, China (e-mail: liaocy@hiscene.com).

K.-T. Cheng is with the School of Engineering, Hong Kong University of Science and Technology Hong Kong (e-mail: timcheng@ust.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2018.2859034

I. Introduction

SIMULTANEOUS Localization and Mapping (SLAM) is a critical task for many robotics and computer vision applications, e.g., obstacle avoidance of drones, augmented reality (AR), etc. A SLAM system relying on a monocular camera as the only input sensor, known as monocular SLAM, is an attractive solution due to its simplicity to use, low cost, easy deployment in personal electronic devices and its superiority over alternative solutions in terms of size, weight, and power characteristics.

Significant advances have been achieved in real-time monocular SLAM during the past decade. These approaches assume that the camera translates in space over time and hence pairs of nearby frames with motion parallex can perform stereo matching for depth estimation. Based on the type of visual cues used for motion stereo matching, i.e., keypoints or raw intensity, existing methods can be categorized into two groups: 1) feature-based methods, such as PTAM [1] and ORB-SLAM [2], which track camera pose and estimate depth by keypoint matching and triangulation of matched keypoints; and 2) direct methods, represented by LSD-SLAM [3], which perform tracking and mapping based on raw color consistency. Despite these advances, existing monocular SLAM methods suffer from two major issues which limit their practical deployment. First, most methods can only provide a sparse or semi-dense map of a scene, in which depths of only high-gradient pixels are calculated. A sparse or semi-dense map is sufficient for localization while not adequate for some applications such as autonomous flight which needs a dense depth map in front of a drone for obstacle avoidance or AR for which virtual objects usually need to interact with the physical scene. Several dense mapping solutions have been proposed to alleviate this problem. However, these approaches either require geometric priors [4]–[6] or a piecewise planar assumption [7]–[9], which might not be feasible in unknown, unstructured environments. Second, scale ambiguity and drifting is another known problem common for monocular SLAM approaches. Some approaches [10] address this issue by detecting the presence of pre-defined 3D objects/landmarks with known depths so as to recover the initial scale based on the detected objects/landmarks. However, 3D object detection itself

1520-9210 © 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

is a challenging task and the absence of known objects in the scene yields failure of scale calibration.

Compared to conventional monocular SLAM, recent advances in depth estimation using weakly-supervised CNNs [11], [12] have achieved better completeness and also offer absolute scales of depth estimation. The key idea behind weakly-supervised depth prediction networks is to learn the correlations between visual patterns/cues and absolute depths from a large amount of training image pairs. However, the generalization ability of most pre-trained networks is usually poor due to the limited amount and variability of training data, yielding significant performance degradation for environments different from those present in the training data.

In this paper, we present a real-time monocular SLAM system which effectively integrates an online-adapted CNN depth prediction network into an existing direct monocular SLAM framework for progressively improving the overall depth prediction accuracy and robustness for scenes of different types and addressing the scale ambiguity problem. Specifically, a weakly-supervised depth prediction CNN [12] pre-trained using pairs of stereo images is employed. Tracking poses from the direct monocular SLAM [3] are applied to select pairs of motion stereo images which are used to tune the CNN model on-the-fly in order to improve its accuracy and generalization ability progressively for environments of different types from the training data. On the other hand, results from the adapted CNN are in turn employed to regress a more accurate absolute scale of depth prediction for each keyframe than CNN-SLAM [13] to alleviate the scale ambiguity problem of the monocular SLAM. The two processes mutually benefit each other to progressively improve the overall performance. Online CNN adaptation is usually considered infeasible for real-time SLAM applications due to three main challenges: 1) training samples collected on-the-fly could contain label noises (i.e., inaccurate tracking poses) arising from errors in monocular SLAM which could make the CNN model overfit to noisy instances; 2) the amount and diversity of training data collected on-the-fly are limited, restricting the generalization ability of the CNN model; 3) slow speed for model update could prevent the CNN model from being a practical depth predictor. To address these challenges, we first design a robust and strict training sample selection mechanism to generate reliable pairs of motion stereo images as training samples for each mini-batch. Additionally, we propose a stage-wise Stochastic Gradient Decent (SGD) method which partitions all model parameters into multiple stages, each stage consists of a stack of layers. The stage-wise SGD updates parameters stage-by-stage rather than updating all parameters at once in each mini-batch iteration. Experimental results show that the stage-wise SGD can better avoid overfitting at the presence of limited amount of training data and a faster convergence speed than the conventional SGD. Furthermore, to minimize the computational cost we design a selective yet effective updating scheme which invokes the adaptation module only when necessary (i.e., where there exists significant appearance difference between the current scene and scenes previously seen by the CNN) and terminates it soon sufficient consistency is observed. To address the scale ambiguity problem, depth estimation from the adapted CNN is employed to regress an absolute scale for each keyframe which is further refined using pixel matching and triangulation between every frame and its nearest keyframe. The depth prediction results from both adapted CNN and the refined semi-dense map of monocular SLAM with a regressed scale are fused to form final reconstruction.

Our experiments on recently published benchmarks over 16 videos demonstrate that our method outperforms the state-of-the-art methods [3], [12], [13] in terms of mapping completeness, mapping and tracking accuracy. The system achieves real-time performance by utilizing both CPUs and GPUs. We incorporate our system into a drone to evaluate its effectiveness for obstacle avoidance, and to demonstrate its potential and efficiency for practicable applications in real scenarios.

To summarize, our main contributions include:

- We propose a real-time dense monocular SLAM system based on an online adapted CNN. To the best of our knowledge, this is the first time an online-updated CNN is integrated into a monocular SLAM framework for progressively improving the accuracy and completeness of monocular mapping.
- We design several effective mechanisms for online training sample selection. The proposed mechanisms are computationally efficient and can choose relevant and reliable training data from the noisy outputs of the monocular SLAM module, facilitating a robust online adaptation of the CNN model.
- We design a novel stage-wise SGD training method with a selective update scheme. The proposed training method and scheme are key enablers for efficiently adapting the CNN on-the-fly and ensuring satisfactory generalization of the CNN model for scenes of different types with a very limited amount of data.
- We design an absolute scale regression method based on the adapted CNN which can greatly improve the accuracy of depth estimation for high-gradient pixels and in turn improve the overall tracking and reconstruction accuracy.
- We conduct extensive experiments on public datasets, as well as testing the obstacle avoidance capability of a drone in practical environments, to evaluate the performance of our method and the value of each proposed technique. Our method achieves the best results for public benchmarks and meanwhile demonstrates the feasibility and real-time performance for practical applications.

II. Related Work

There exists a vast body of literature on SLAM [2], [3], [14]–[17], depth estimation [12], [18]–[21], and 3D reconstruction [22], [23]. In this section we focus on reviewing only the work on two subjects that are most relevant to our study: monocular SLAM and single-view depth prediction.

A. Monocular SLAM

Several monocular SLAM methods for accurate, robust, efficient tracking and mapping have been developed during the past decade [1]–[3], [8], [14], [15]. Arguably, ORB-SLAM [2], which utilizes sparse ORB features for tracking, relocalization, loop closure and mapping, is the state-of-the-art in terms of pose estimation accuracy. However, ORB-SLAM can only provide very sparse depth reconstruction of the scene which is sufficient for camera pose tracking while the map is of low value for other applications. To provide denser map reconstruction, direct methods, such as LSD-SLAM [3] which relies on raw intensity information of all high-gradient pixels rather than sparse local features, were introduced. However, direct methods can only provide depth estimation for high-gradient pixels, which remains inadequate for obstacle avoidance and AR interaction. To achieve dense mapping, DTAM [14] utilized a regularization term which enforces smoothness of estimated depth for pixel-wise depth estimation. However, DTAM requires rich texture and large-parallax camera motions for accurate depth estimation and performs poorly in textureless regions. Some methods integrate geometric priors with monocular SLAM for dense mapping. For these methods, the geometric priors could be either obtained via 3D object detection and recognition [4]–[6] or based on a piecewise planarity assumption [7]–[9]. However, failures of 3D object detection/recognition and/or contradiction to the piecewise planarity assumption in complex unstructured scenarios could conversely degrade the dense mapping performance. In [24], the authors utilized active learning based on a random forest predictor to achieve 3D dense mapping in real time with low power consumption.

B. Single-View Depth Prediction Network

Conventional methods for single-view depth prediction mainly depend on handcrafted features and probabilistic graphical models [25], [26] to derive regularized depth maps. With the recent advances in deep learning, we have witnessed increasing popularity of using CNN-based methods for single-view depth prediction. Based on the types of training labels, existing CNN-based methods can be categorized into either supervised or weakly-supervised methods. For supervised methods, such as [18]–[20], absolute depth maps acquired by active depth sensors are required as training labels. Depth prediction is then formulated as a regression problem by training a CNN model which minimizes differences between CNN-predicted depths and the ground truth. A major weakness of supervised methods is the need to train the network using a vast amount of data with absolute depths as labels which typically requires expensive hardware, and careful acquisition. To alleviate such demands, several weakly-supervised methods have been proposed with the goal of easing the training label acquisition. Specifically, weakly-supervised methods [11], [12] generally train a CNN using stereo image pairs, i.e., source and target, with small, fixed, and known camera motion between the two images of each pair. The CNN takes the source image of a stereo pair as input and predicts a disparity map with respect to the target image, based on which an inverse warp of the target image can be generated to reconstruct the source image. The training process is to minimize reconstruction error between the reconstructed and the original source images over the training set. During the testing phase, only a single image is required as an input and the CNN outputs its disparity map, based on which a depth map can be derived.

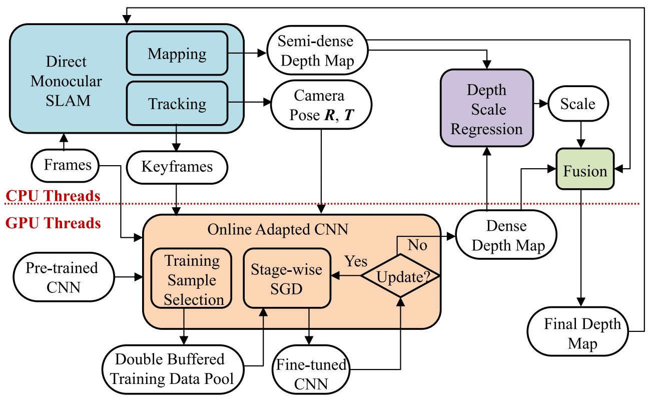


Fig. 1. An overview of our dense monocular SLAM system which consists of four major modules: direct monocular SLAM, online adapted CNN, depth scale regression and fusion.

Despite successes, two major drawbacks of CNN-based methods limit their application for SLAM: 1) significant performance degradation for environments different from those present in the training data due to limited generalization ability of CNNs and the great variety of scenes in practical scenerios; and 2) the inherently error-prone characteristics for high-gradient pixels which could yield large tracking errors.

C. Fusion of Single-View Depth Prediction and Monocular SLAM

To address the above two limitations of depth CNNs, methods have been proposed to integrate a depth CNN into a monocular SLAM for improving both map density and accuracy. For instance, in [27] Bloesch et al proposed CodeSLAM to improve the compact and dense representation of scene geometry from a depth CNN inside a SLAM loop. In [13], the authors proposed CNN-SLAM in which the initial depth values for keyframes of a monocular SLAM is predicted by a depth CNN (i.e., Laina [20]) and the depth values are further revised via high-gradient pixel matching and triangulation, and graph optimization in a SLAM loop. Despite performance improvement of CNN-SLAM, the problems are far from solved. First, only high-gradient pixels depths are refined in CNN-SLAM while those of low-gradient pixels remain intact, yielding overall unsatisfactory reconstruction in particular for scenes different from those present in the training data. In addition, incorrect scale predictions from CNN for high-gradient pixels result in inadequate initialization and in turn yield non-trivial mapping and tracking errors. In contrast, we propose an online adapting CNN which is integrated into a monocular SLAM system. The proposed method could progressively improve the depth prediction accuracy for all pixels even for scene types which are quite different from those in the training data. In addition, we propose a new absolute scale regression method for reducing tracking errors and achieving more accurate dense mapping.

III. Method

Fig. 1 shows the overview of our system which consists of four major modules: direct monocular SLAM, online-adapted

CNN for depth prediction, depth scale regression and fusion. The direct monocular SLAM runs two parallel threads on CPU: 1) a tracking thread which estimates camera pose of each frame and selects keyframes, and 2) a mapping thread which calculates a semi-dense depth map of each keyframe. Camera poses and corresponding keyframes are selected and buffered in a training sample pool. Initially, pixel-wise depth is predicted based on a pre-trained model. Once the accuracy of the model becomes inadequate and the training sample pool is full, the system starts tuning the CNN model using the buffered training samples. Before the convergence of the tuning process, the depth of keyframes will still be predicted using the previous CNN model to ensure uninterrupted dense map prediction. Once the fine-tuned model is ready, it replaces the old one. The dense map prediction together with the semi-dense map of a keyframe from monocular SLAM are employed to regress an absolute depth scale for the semi-dense map. Finally, the refined semi-dense map with an absolute scale and the dense map are fused to generate the final result. In the following, we describe each module in more details.

A. Direct Monocular SLAM

1) Camera Pose Tracking: Similar to [3] the direct tracking method for camera pose tracking is implemented by searching for an optimized camera pose T_k^n between current frame n and nearest keyframe k that can minimize the photometric error between the frame n and keyframe k. Since homogeneous regions could lead to ambiguous pixel matching between frames (i.e., similar photometric errors for different T_k^n), thus to achieve high tracking robustness and reduce time cost for optimization, the photometric error $\mathbf{r}$ is typically calculated based on only high-gradient pixels $\{p\}$ in keyframe k as:

$$\mathbf{r} = \sum_{\{p\}} \mathbf{r}_p = \sum_{\{p\}} \left| I_k(p) - I_n\left(\pi\left(T_k^n \cdot \pi^{-1}\left(p, \mathcal{D}(p)\right)\right)\right)\right|^2 \tag{1}$$

where $\mathcal{D}(p)$ represents the depth value of a high-gradient pixel p, π is the projection model which maps a 3D point P^c in camera coordinates to a pixel p on a 2D image plane. π is determined by the 3×3 intrinsic camera parameters $\mathbf{K}$ which consists of the focal length and principal point offset of the camera. The parameters of $\mathbf{K}$ are obtained via the standard camera calibration process. Similarly, π^{-1} is the inverse projection model which maps $\mathbb{R}^2$ to $\mathbb{R}^3$. An optimized T_k^n is then solved via minimization of photometric error $\mathbf{r}$ for all high gradient pixels:

$$T_k^n = \arg\min_{T_k^n} \sum_{\{p\}} w_p \mathbf{r}_p \tag{2}$$

where w_p is the weight of pixel p proposed in [3] for increasing robustness and minimizing the impact from outliers. Eq. 2 can be solved by a standard Gauss-Newton optimization method.

2) Semi-Dense Depth Estimation: Similar to [28], the mapping thread of the direct SLAM estimates depths of high-gradient pixels through small-baseline stereo comparisons, i.e., pixel matching followed by triangulation. Specifically, once a new keyframe k is created, its depth map $\mathcal{D}_{pri}$ and depth estimation uncertainty $\mathcal{U}_{pri}$ are first initialized by projecting depths $\mathcal{D}_{k-1}$ and uncertainty $\mathcal{U}_{k-1}$ from the previous keyframe $k-1$ into it as:

$$\mathcal{D}_{pri} = \mathcal{D}_{k-1} - t_z \tag{3}$$

$$\mathcal{U}_{pri} = \left(\frac{\mathcal{D}_{k-1}}{\mathcal{D}_{pri}}\right)^4 \mathcal{U}_{k-1} + \sigma^2 \tag{4}$$

where t_z is the camera translation along the optical axis and σ^2 is the standard deviation of the initialization noise. The initialized depth map is refined using the subsequently tracked frames by first searching for a matching pixel for each high-gradient pixel p on the keyframe k along the epipolar line on the current frame. The search range along the epipolar line is determined according to the depth uncertainty of p, i.e., $\mathcal{U}_{pri}$. Once the matching pixel is obtained, the depth of p can be solved via triangular. We represent the pixel matching and triangulation process using a function $\mathcal{F}$, accordingly the observed depth $\mathcal{D}_{obs}$ based on $\mathcal{F}$ is denoted as:

$$D_{obs} = \mathcal{F}(D_{pri}, I_k, I_{cur}, T_{cur}^{ki}, \mathbf{K}) \tag{5}$$

where I_k and I_{cur} are keyframe k and a current frame respectively, T_{cur}^{ki} and $\mathbf{K}$ are camera motion from keyframe k to a current frame and camera intrinsic matrix respectively. The uncertainty of $\mathcal{D}_{obs}$, i.e., $\mathcal{U}_{obs}$, arises from noise in pixel matching between I_k, and I_{cur}, and noises in camera motion estimation T_{cur}^{ki}. It can be computed based on the method described in [28]. The refined depth and its uncertainty is then a fusion of initialized depth map and observed depths, which conforms to a distribution:

$$\mathcal{N}\left(\frac{\mathcal{U}_{pri}\mathcal{D}_{obs} + \mathcal{U}_{obs}\mathcal{D}_{pri}}{\mathcal{U}_{pri} + \mathcal{U}_{obs}}, \frac{\mathcal{U}_{pri}\mathcal{U}_{obs}}{\mathcal{U}_{pri} + \mathcal{U}_{obs}}\right) \tag{6}$$

B. Weakly-Supervised CNN for Depth Prediction

Every time a new keyframe is created, a dense depth map is calculated for it via CNN. In this work, we employ the state-of-the-art approach [12] which can be trained in a weakly-supervised manner for single-view depth prediction. Fig. 2 shows the network architecture. In particular, the first part of the architecture is based on the convolutional layers (ConvLayers), pooling layers and batch normalization layers of ResNet-50 [29]. Note that we do not include the fully connected (FC) layers of ResNet-50 in our network architecture. We then apply a ConvLayer using a $1 \times 1 \times 2048 \times 1$ kernel to convert 2048 feature maps into a single 8×6 feature map, followed by two FCN blocks [30] with skip connections from the input of the previous pooling layers and three deconvolutional layers to increase the size of the feature map to the original input size.

Training the depth prediction CNN requires pairs of rectified stereo images which are captured by stereo cameras with a fixed baseline $B_{pre-train}$ and camera focal length $f_{pre-train}$ as training data. Consequentially, each pair of training images are coplanar and have pixel disparities along only x-axis and little disparities along y-axis. By explicitly generating an inverse image warp of the target image (i.e., a reconstructed source image) based on the disparity map $Dis(p)\{p = \{u, v\}\}$ predicted from the CNN, the photometric error between the original source and

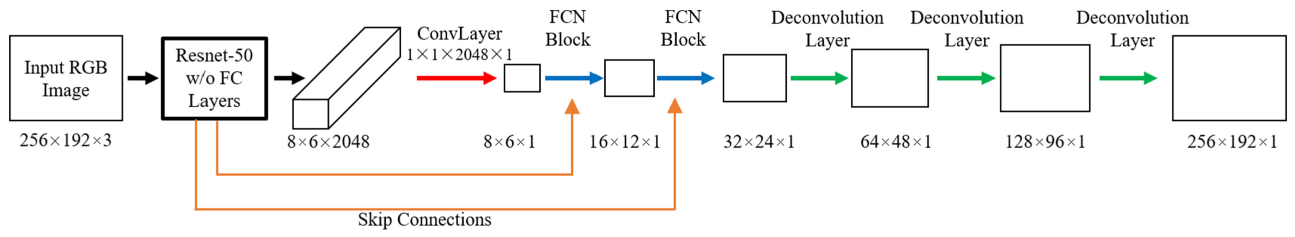


Fig. 2. Network architecture. The black, red, blue, green arrows correspond to ResNet-50 w/o FC, convolution, FCN, deconvolution layers respectively. The orange arrows denote skip connections. The two FCN blocks upsample the predictions and combine them with the input of the previous pooling layers respectively.

the reconstructed source can be derived as:

$$E_{photometric} = \sum_{\{p\}} \|I_{warp}(p) - I_{source}(p)\|^2 = \sum_{\{p\}} \|I_{target}(u + Dis(p), v) - I_{source}(u, v)\|^2 \quad (7)$$

where $\{p\} \in \mathbb{R}^2$. Note that the photometric error is not informative in homogeneous areas. A smoothing term based on the derivative of the predicted disparity is introduced:

$$E_{smooth} = \|\nabla Dis(p)\|^2 \quad (8)$$

The back propagation error of the network is then defined as the weighted summation of the two terms:

$$E = E_{photometric} + \gamma E_{smooth} \quad (9)$$

where γ adjusts the impact of the smoothing term and in this work we set it to 0.01 as suggested in [12]. The training process aims at minimizing the back-propagation error E over the training set.

During the testing phase, the CNN takes a single image as input and outputs its disparity map $Dis(p)$ based on which, the predicted depth for each pixel $\mathcal{D}(p)$ is:

$$\mathcal{D}(p) = \frac{f_{pre-train} \times B_{pre-train}}{Dis(p)} \quad (10)$$

C. Online CNN Adaptation

A pre-trained CNN for depth prediction could well learn the relationships between visual structures and depths for similar scenes. However, one major limitation is that it does not generalize well outside their immediate types of scenes which was also noted in [12], [19]. For example, a network that is trained using indoors scenes like Beijing quadrangles do not work well for indoor scenes of French palaces nor street scenes. Additionally, the network could inherently encode intrinsic parameters and baseline of the stereo camera used for acquiring training images into the network. For images captured using cameras of different parameters, the accuracy of depth prediction could degrade significantly.

In the following, we present a novel method for tuning the CNN on-the-fly once unknown scenes that are sufficiently different from those previously seen by the current CNN model are encountered. A desired method for online CNN adaptation should achieve the following two goals: 1) the generalization ability of the CNN model can be improved progressively during online adaptation; and 2) each round of the adaptation process can be completed within a very short and constrained period of time. To this end, we design a robust mechanism for selecting and accumulating temporal training samples from new frames captured by the SLAM system. Once the training pool is full and the CNN model demands refinement, the tuning process based on the stage-wise SGD method is triggered. After completion of the tuning process, the system discards the training images and starts the collection of a new set of training images for the next round of adaptation. We detail each step in the following.

1) Online Tuning Data Selection: The depth prediction network requires pairs of rectified stereo images with a fixed baseline $B_{pre-train}$ as input data for training. To tune the CNN model on-the-fly, we collect pairs of images captured at two distinct moments by a monocular camera during its movement (i.e., motion stereo) to simulate stereo images. Typically stereo camera is almost coplanar to each other with relatively small rotations and translations along y and z axises. Therefore, after image rectification two images from the stereo camera can be easily aligned to share a common image plane to have only horizontal disparities with image distortions. However, for a monocular camera, there could exits nontrivial rotations and translations along y and z axises, yielding errors and severe distortions after rectification of a motion stereo image pair. Since each round of online CNN adaptation relies on only a small amount of training data, any errors and distortions in training data could have significant negative impact on the adapted CNN model, yielding a potential overfit to noisy instances. Therefore, it is critical to collect highly reliable training samples and in this work we design the following constraints to meet this objective.

- **Camera motion constraints:** For each round of CNN model update, we require all training samples in a mini-batch to have a common baseline B. Thus, we collect image pairs captured by cameras with a relative translation $T = \sqrt{t_x^2 + t_y^2 + t_z^2}$ obtained from the tracking thread of the direct SLAM, and $\|T - B\| \leqslant \delta$ where δ is a very short distance, say 0.01 m, to ensure small differences in T among a mini-batch of training samples. Besides, to minimize errors/distortions in image rectification, we add another requirement that the translation along the y and z axes between two camera positions must be extremely small to keep disparity mainly horizontal. In our implementation, we require $|t_x| > 0.9 \times T$, where 0.9 is determined empirically for ensuring dominant disparity along only the x axis.

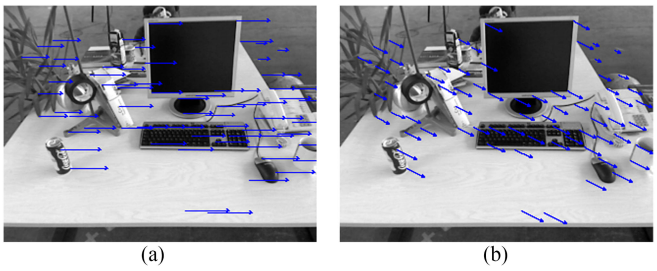


Fig. 3. Illustration of disparity constraint. A pair with non-horizontal whose disparity is almost horizontal (a) is selected as an candidate. A pair with non-horizontal disparity (b) is discarded as noise.

- **Disparity constraint:** Once pairs of motion stereo images are collected based on the motion constraint, image rectification based on [31] is applied to project the two images of a stereo pair to a common image plane. However, despite confining the camera translation of an image pair to be small in the y and z directions, it is still inevitable that 1) translations obtained from direct SLAM could have errors, 2) there exits rotations between two images, and 3) image rectification is imperfect, yielding noisy image pairs with non-zero disparities along y and z axes after rectification. To further eliminate these noisy data, we match corners between images of a stereo pair via the optical flow method [32] and calculate the average vertical disparity Dis_{avg} for all matching corners. Image pairs whose $Dis_{avg} < \varepsilon$ (ε is set to 0.5 in our experiment) are kept as candidate training samples, as shown in Fig. 3 (a). Otherwise, they are considered noisy pairs and are discarded, as shown in Fig. 3(b).
- **Diversity constraint:** To ensure sufficient diversity of training data in each mini-batch, we require that each selected pair of motion stereo images is associated with a unique keyframe. Note that in direct SLAM, a new keyframe is created whenever the difference between the current frame and the previous keyframe is sufficiently large. Therefore, having training samples selected from different keyframe intervals can ensure sufficient appearance differences among training data and can in turn reduce the risk of overfitting.
- **Training pool size constraint:** We apply a batch of training samples, more than just a single training image, to update CNN parameters in each adaptation round to ensure sufficient robustness of training and avoid the negative impact caused by noises due to retraining using a single sample. For details of the selection of batch size please refer to *Sec. IV-A*. We use a double-buffered training pool for storing online collected image pairs and for reducing the wait time for data collection. Once the first buffer is full, the second buffer is then used to continue the collection of training samples while its examined whether the CNN model should be updated or not. If yes, all samples in the first buffer are sent to the GPU for training. Otherwise, these samples are discarded. Once the first buffer is emptied, the two buffers are swapped.

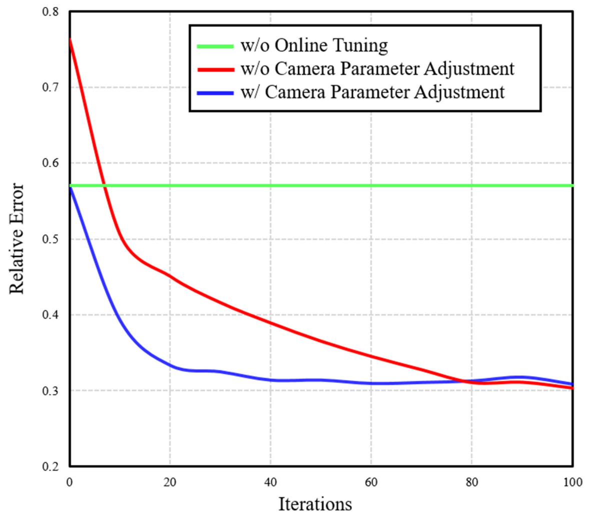


Fig. 4. Comparison of testing errors among a pre-trained CNN model, an online adapted CNN model w/ and w/o camera parameter adjustment.

2) Adjustment Based on Camera Parameters: It is noted that the focal length $f_{adapted}$ of the monocular camera used for online capturing the training data and the baseline $B_{adapted}$ between two images of a motion stereo pair could be different from those used for acquiring the original training images for the pre-trained model, i.e., $f_{pre-train}$ and $B_{pre-train}$. Since the relationship of camera parameters and scene depths have been encoded into the network implicitly, for an image using a different focal length the absolute scale of the 3D reconstruction will become inaccurate. We collected a new set of training images (from the Malaga dataset [33]) which are captured using a camera with quite different parameters from those used for capturing the original training images (from the KITTI dataset [34]). We fine-tuned the pre-trained CNN model trained from the KITTI dataset images using the new set of training images and evaluated the relative depth prediction error on the testing images (from the Malaga dataset). Fig. 4 shows the relative error with respect to the number of training iterations. The green line denotes the training loss based on the pre-trained CNN model and the red line indicates the result of an online adapted CNN without camera parameter adjustment. Clearly, the training loss of a new batch of training images is much greater than that of previous training data. Consequently, the entire network needs to be adjusted so as to accommodate the camera parameter differences, yielding a slow updating speed. To alleviate this problem, we adopted a similar strategy proposed in CNN-SLAM [13] which adjusts the output disparity map by multiplying the disparity of every pixel with a scale factor δ as:

$$\delta = \frac{f_{adapted} \times B_{adapted}}{f_{pre-train} \times B_{pre-train}} \tag{11}$$

The blue curve denotes the relative error of the online adapted CNN with camera parameter adjustment. Clearly, with camera parameter adjustment the model tuning process has a faster convergence speed and better prediction accuracy.

3) Stage-Wise SGD for Tuning: We use ResNet-50 for extracting different levels of features from an input image which are then encoded into a disparity map via a set of upsampling blocks. ResNet-50 is a very deep network consists of 49 convolutional layers. Tuning the entire network of such a deep and complex network architecture using a small amount of data could easily lead to overfitting of the CNN. As result, when a new batch of training data arrives, the limited generalization ability of the current CNN model might not be able to infer a correct relationship between currently captured features and the disparities. Thus, the entire network need to be adjusted so as to learn from the new image batch, yielding a slow updating speed.

To reduce the risk of overfitting given the limited amount of training data, we reduce the network complexity for each update. Specifically, we follow the same strategy as [29] to group the entire ConvLayers of ResNet-50 into 5 stages, namely conv1, conv2_x, conv3_x, conv4_x and conv5_x. Except conv1 which consists of a single ConvLayer, each of the other four stages consists of a number of stacked bottleneck building blocks (i.e., $1 \times 1, 3 \times 3, 1 \times 1$ convolutional layers with a shortcut connection). For each mini-batch for training, we only update parameters of a single stage $W_i^k (i = 1, 2, 3, 4, 5)$ in one iteration k and keep parameters of other stages intact. In the immediate next training iteration, we only update parameters of the next stage $W_j^k \; (j = (i+1)\,\%5)$. This process iterates until a stopping critera is met (i.e., either the number of iterations reaches a limit, or the training loss is lower than a pre-defined threshold). We have also experimented with different stage sampling schemes for iterative parameter update, e.g., random selection of stages. Experimental results indicated that a stage-wise SGD with a sequential selection scheme achieves the best performance and could greatly alleviate the overfitting problem.

It is indeed quite possible that the training based on a single batch could fall into a local minimum. But as we continuously provide newly collected batches on-the-fly, such a problem is alleviated. Once a new batch is applied to the network, the gradient is no longer zero which in turn updates network parameters and helps the network to get out of the local minimum.

4) Selective Update: Updating the CNN model every time when a new batch is ready could lead to unnecessarily excessive computational cost. As long as the current CNN model is capable for providing sufficiently accurate depth prediction of the current scenes, we maintain the current model until adaptation becomes a necessity. We therefore calculate the training loss of the CNN model for each image batch. Once the overall loss of the batch is greater than a predefined threshold L_{high}, we invoke the adaptation process. The adaptation process based on a single batch terminates until the training loss is reduced to below L_{low} or the number of iterations reaches a pre-defined limit. This strategy greatly reduces the computational cost while maintaining satisfactory accuracy.

The process of online CNN adaptation is illustrated in Algorithm 1.

Algorithm 1: Online CNN Adaptation

```
Input: Image pairs with known relative transform,
    Pre-trained CNN model $W$ separated into 5 stages
    ($W_1$,. . ., $W_5$); Threshold $L_{high}$ and $L_{low}$, Limit of
    iteration count $K$.
Output: Online tuned CNN model $W^*$
 1: while Direct SLAM thread is running do
 2:   if $1^{st}$ and $2^{nd}$ buffers are not full then
 3:     Select training data and Put into pool
 4:   else if $1^{st}$ buffer is full and $2^{nd}$ buffer is not full then
 5:     Transfer data form CPU memory to GPU
 6:     Empty $1^{st}$ buffer and swap $1^{st}$ and $2^{nd}$ buffers
 7:     Calculate Loss $L$ of current batch of training data
 8:     if $L < L_{high}$ then
 9:       Do not Update
10:     else
11:       for $k = 0$ to $K$ do
12:         Stage id $j = (k+1)\%5$
13:         Update $W_j$ using SGD
14:         Calculate loss $L$ of current batch of data
15:         if $L < L_{low}$ then
16:           Break
17:         end if
18:       end for
19:     end if
20:     Save $W$ to $W^*$
21:   end if
22: end while
```

D. Accurate Scale Regression Based on Adapted CNN

Accurate estimation of camera motions with absolute scale information, which acts as the camera motion constraint, is essential for selecting reliable training data for online CNN adaptation. However, camera motions estimated from the direct monocular SLAM typically lack the absolute scale, which could cause significant noises in the online collected training data and, in turn, CNN tuning. In this section, we present a simple yet effective method for absolute scale regression based on the depth predicted via the adapted CNN.

A key advantage of the CNN method is the ability of predicting absolute depths. Intuitively, directly using the CNN depth predictions for tracking could address the scale ambiguity problem for both tracking and mapping. However, as shown in Fig. 6, the depths predicted by the CNN could have large prediction errors for some high-gradient pixels, and in turn yielding inaccurate tracking as the tracking mainly relies on the depths of high-gradient pixels (as shown in Eqs. 1 and 2).

We plot the distribution of $\mathcal{D}_{sd}(p) - \mathcal{D}_{gt}(p)$ using blue points in Fig. 5 (a), where $D_{sd}(p)$ denotes semi-dense depth prediction from monocular SLAM for a particular high-gradient pixel p and $\mathcal{D}_{gt}(p)$ indicates the ground truth depth for pixel p. The ratio between $\mathcal{D}_{gt}(p)$ and $\mathcal{D}_{sd}(p)$, i.e., $S(p) = \frac{D_{gt}(p)}{\mathcal{D}_{sd}(p)}$, is the scale difference between the semi-dense estimation and ground truth for pixel p. From Fig. 5 (a) we observe that, points of $\mathcal{D}_{sd}(p) - \mathcal{D}_{gt}(p)$ align well along a line with a slope of 1.39, which implies that for most high-gradient pixels, their relative depth predictions are accurate and share a common scale. Based on this observation, we consider that if we could obtain a set of high-gradient pixels $\{\mathbf{P}\}$ whose absolute depth predictions

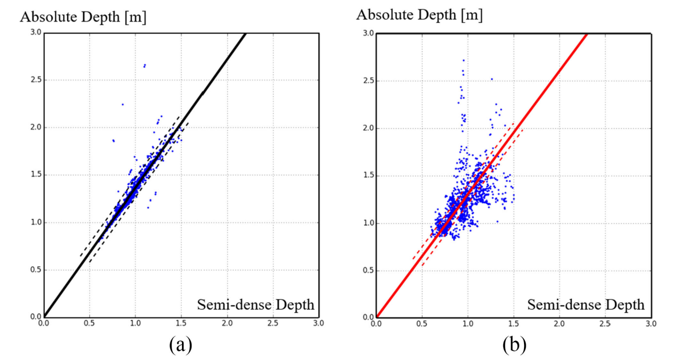


Fig. 5. Illustration of scale regression method. (a) RANSAC + Ground-truth slope = 1.39. (b) RANSAC + CNN Prediction slope = 1.30.

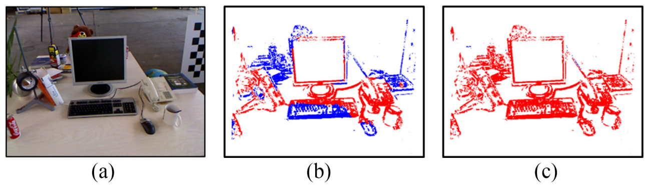


Fig. 6. Comparison between (b) CNN depth prediction and (c) direct SLAM depth estimation with regressed scaling factor. Red pixels denote correctly estimated depths whose errors are within 10% of the ground truth and blue pixels indicate incorrect depth estimations whose error with respect to the ground truth are greater than 10%. (a) RGB Image. (b) CNN Depth Prediction for High Gradient Pixels. (c) Direct SLAM Depth Estimation with Regressed Scaling Factors.

(i.e., $\mathcal{D}_{cnn}(p), p \in \{\mathbf{P}\}$) are accurate, we can calculate the scale $S(p) = \frac{\mathcal{D}_{cnn}(p)}{\mathcal{D}_{sd}(p)}$ for all pixels p, and regress a common scale for all semi-dense estimations. However, it is inevitable that depth predictions by the CNN are inaccurate for some pixels. As shown in Fig. 5(b), we plot distribution of $\mathcal{D}_{sd}(p) - \mathcal{D}_{cnn}(p)$ for all high-gradient pixels p using blue points, where $\mathcal{D}_{cnn}(p)$ is the depth of point p predicted by the CNN. The points scatter in the figure, implying that the CNN depth predictions are inaccurate for many pixels. To identify a subset of CNN depth predictions that are more accurate (i.e., inliers) based on which an absolute scaling factor is regressed, we employ the RANSAC algorithm [35], which is a popular method for regressing correct parameters with the presence of a nontrivial amount of outliers. The black and red lines denote the regressed slopes (i.e., scales) based on the ground truth and the CNN predictions via RANSAC. Obviously, RANSAC helps achieve an accurate regressed scale which is close to scale based on ground truth.

We display the semi-dense depth estimation multiplied with the regressed absolute scaling factor in Fig. 6 (c). Clearly, almost all high-gradient pixels can obtain correct depth estimations, facilitating more accurate pose tracking. Fig. 7 compares pose trajectory accuracy of our method with regressed scaling factor based on the CNN-predicted depth with that of LSD-SLAM and the ground truth on two exemplar sequences from the *TUM* dataset [36]. Results show that after multiplying a scaling factor regressed based on the CNN-predicted depth, the estimated trajectory from the monocular SLAM is much closer to the ground truth than that of the LSD-SLAM which lacks absolute scale information.

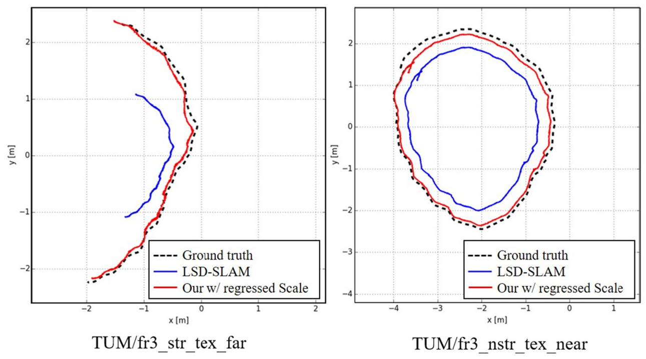


Fig. 7. Comparison of pose trajectory accuracy among ground truth trajectory, LSD-SLAM and our method with regressed scaling factors on two video sequences from the TUM dataset.

E. Result Fusion

Our monocular SLAM obtains two depth maps for each keyframe: a dense depth map $\mathcal{D}_{cnn}$ from the continuously adapted CNN and a refined semi-dense map $\mathcal{D}_{sd}$ with an absolute scale. To combine the two maps for a more accurate reconstruction, we fuse the results from two aspects.

1) Fusion of $\mathcal{D}_{cnn}$ and $\mathcal{D}_{sd}$: Recall that, a smoothing term, expressed in Eq. 8, is integrated into the back-propagation error of the depth prediction network, denoted in Eq. 9, so as to avoid incorrect depth prediction in the homogeneous regions. However, the smoothing term could also enforce depth continuity even at regions with drastic depth changes, and in turn yield large errors in such regions. The semi-dense map, on the other hand, is accurate in the high-gradient regions with respect to a relative scale factor, which could well complement the CNN depth prediction. Thus, we propose to combine the two maps using a voting strategy. Specifically, for each pixel p in keyframe i, we project it to its nearest keyframe $i-1$, denoted as $p^{'}_{cnn}$, according to the transformation $T^{k_{i-1}}_{k_i}$ and its depth $\mathcal{D}_{cnn}(p)$ predicted by the CNN. Similarly, we make another projection of p to keyframe $i-1$, denoted as $p^{'}_{sd}$, based on the depth $\mathcal{D}_{sp}(p)$ from semi-dense mapping with an absolute scale.

We choose a small patch centered around the projected points $p^{'}_{cnn}$ and $p^{'}_{sd}$ respectively in keyframe $i-1$ and calculate the normalized cross correlation NCC_{cnn} between patches $R(p)$ and $R_{cnn}(p^{'})$ and NCC_{sd} between patches $R(p)$ and $R_{sd}(p^{'})$. If NCC_{cnn} is smaller than NCC_{sd} which implies that the depth prediction from semi-dense mapping than CNN, we choose $\mathcal{D}_{sd}(p)$ as the final prediction for pixel p. Otherwise, we choose $\mathcal{D}_{cnn}(p)$. For pixels which only have a single prediction from $\mathcal{D}_{cnn}(p)$, the final map of pixel p will then be based on $\mathcal{D}_{cnn}(p)$.

2) Inter-Keyframe Fusion of Depth: Similar as CNN-SLAM [13], for each depth map $\mathcal{D}_{k_i}$ of keyframe k_i, we associate it to its uncertainty map U_{k_i} which is calculated as the element-wise square difference between k_i and that of the nearest keyframe $k_{i-1} (i > 1)$, warped according to the estimated transformation $T^{k_i}_{k_{i-1}}$ as:

$$\mathcal{U}_{k_i}(p) = (\mathcal{D}_{k_i}(p) - \mathcal{D}_{k_{i-1}}(p^{'}))^2 \tag{12}$$

where $p^{'}$ represents the pixel in keyframe k_{i-1} calculated by the reprojection $p^{'} = \pi(T_{k_i}^{k_{i-1}} \cdot \pi^{-1}(p, d_p))$.

Each element of the uncertainty map measures how incoherent each predicted depth value is across different keyframes. To further improve the accuracy of each newly predicted keyframe k_i, we fuse its depth map $\mathcal{D}_{k_i}$ and uncertainty map $\mathcal{U}_{k_i}$ with those propagated from the nearest keyframe $k_{i-1}(i > 1)$ as:

$$\tilde{\mathcal{U}}_{k_{i-1}}(p^{'}) = \frac{\mathcal{D}_{k_{i-1}}(p^{'})}{\mathcal{D}_{k_i}(p)} \mathcal{U}_{k_{i-1}}(p^{'}) + \sigma^2 \tag{13}$$

where σ^2 represents white noise variance and:

$$\mathcal{D}_{k_i}(p) = \frac{\tilde{\mathcal{U}}_{k_{i-1}}(p^{'}) \cdot \mathcal{D}_{k_i}(p) + \mathcal{U}_{k_i}(p) \cdot \mathcal{D}_{k_{i-1}}(p^{'})}{\mathcal{U}_{k_i}(p) + \tilde{\mathcal{U}}_{k_{i-1}}(p^{'})} \tag{14}$$

$$\mathcal{U}_{k_i}(p) = \frac{\tilde{\mathcal{U}}_{k_{i-1}}(p^{'}) \cdot \mathcal{U}_{k_i}(p)}{\mathcal{U}_{k_i}(p) + \tilde{\mathcal{U}}_{k_{i-1}}(p^{'})} \tag{15}$$

IV. Experiments

We experimentally evaluate the performance of our method in terms of tracking and reconstruction accuracy on two public benchmark datasets: the *ICL-NUIM* dataset [37] and the *TUM RGB-D SLAM* dataset [36]. Both datasets include video sequences of indoor environment, and provide ground truth depth maps and camera trajectories. Each video in the *ICL-NUIM* dataset is about 30∼40 seconds long and the length of each video in the *TUM RGB-D SLAM* dataset is about 30∼150 seconds. In all our experiments, we used the pre-trained CNN model which was originally trained using 15,407 pairs of stereo images in which 6,510 pairs are from the *Wean Hall* dataset [38] captured using a common stereo camera with a fixed baseline 0.12 m and 8,897 pairs are from our own images acquired using a ZED stereo camera. Our self collected images are captured in a conference room. Like [12], we take randomly applied conventional methods (e.g., random scale, translation, rotation, left-right flip) to augment the dataset to 42,098 pairs. The amount of origin training data used for CNN-SLAM [13] is about 12,000. After data augmentation, the total number of training images for CNN-SLAM [13] is about 96,000, which would be twice of the total number of training samples used in our method.

The evaluation is performed on a desktop PC with an Intel Xeon CPU at 3.6 GHz with 32 GB of RAM running Ubuntu operating system, and a Nvidia GM200 GPU with 12 GB of VRAM. We use the source code provided by authors of [3] for the implementation of LSD-SLAM,[1] and the source code provided by authors of [12] for the implementation of CNN raw prediction.[2] Since the authors of CNN-SLAM did not provide open source implementation for the method, we use the results reported in [13] for the evaluation.

In addition to quantitatively assessment on public datasets, we have also incorporated our system into a drone for obstacle avoidance, and examine its effectiveness for this application. In the following, we first conduct experiments for selecting batch size, which is a critical parameter of our system (*Sec. IV-A*). Then we present comparison results in terms of mapping accuracy (*Sec. IV-B*), tracking accuracy (*Sec. IV-C*), runtime efficiency (*Sec. IV-D*), and its application to obstacle avoidance (*Sec. IV-E*).

[1]https://github.com/tum-vision/lsd_slam
[2]https://github.com/Ravi-Garg/Unsupervised_Depth_Estimation

TABLE I
Runtime of Online Training for Each Update Iteration

Batch size	2	4	8	16
Time per iteration	58ms	86ms	127ms	203ms

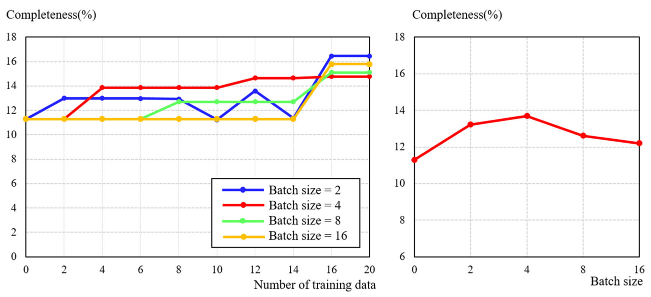


Fig. 8. Completeness of different batch size.

A. Batch Size Selection

A smaller batch size implies faster collection of the online data and quicker convergence of the training process, but it also makes the online updated model more sensitive to noise, yielding performance fluctuation for different online updates. On the other hand, a larger batch size demands a longer period of time for online data collection and a slower training convergence, but the online updated model is more robust to noises.

We have conducted experiments to evaluate the depth estimation performance and runtime for different batch sizes (Table I). To evaluate the performance of dense mapping, we use the same metric (i.e., completeness) as used in [8], [9], [13], which calculates the percentage of depth predictions whose difference with respect to the corresponding ground truth depth is less than 10%, i.e., $\frac{|D - D_{gt}|}{D_{gt}} < 10\%$. The evaluation was conducted on a validation dataset whose video sequences is different from the testing video sequences. The validation dataset is captured in an office room and lasts about 120 seconds.

Fig. 8 shows the experimental results. With a batch size of 2 (denoted by the blue line in the left figure of Fig. 4), the first online update becomes available after two training instances collected, but when noisy instances were selected in the 5th and 7th online updates the performance degrades. With a batch size from 4 to 16, the online update process is relatively stable due to the smoothing effect among training samples of each batch. However, a large batch size increases the wait time for online data collection and runtime for online training, potentially reducing the benefits of online update and yielding a lower average completeness. As shown in the right figure of Fig. 8, for a batch size of 4 we could achieve the highest average completeness on the validation set. Table IV shows the runtime of online training for each update iteration, which indicates the monotonic increase of the online training time with the batch size increases from 2 to 16. Based on the results of completeness and training

TABLE II
COMPARISON OF PERCENTAGE OF CORRECTLY PREDICTED DEPTH ON ICL-NUIM AND TUM DATASETS

	Perc. Correct Depth (%)						**Online Adaptation Profile**		
	LSD SLAM [3]	CNN SLAM [13]	Ours w/o online &fusion [12]	Ours w/ normal SGD	Ours w/ stage-wise SGD	Ours Final	Total Batches	Used Batches	Tuning Iterations
ICL/office0	0.153	19.410	19.117	18.880	22.206	**22.940**	7	3	33
ICL/office1	0	29.150	28.086	30.045	31.289	**34.653**	3	1	20
ICL/office2	0	**37.226**	21.695	21.695	21.695	22.059	6	0	0
ICL/office3	0	-	20.095	**21.269**	20.302	20.607	1	1	7
ICL/living0	0.019	12.840	18.680	15.189	**23.278**	22.698	4	3	31
ICL/living1	0	13.038	21.071	21.964	22.774	**25.715**	3	2	38
ICL/living2	0.411	**26.560**	16.150	16.823	20.995	22.801	4	2	39
ICL/living3	0.401	-	10.087	11.506	**13.141**	12.751	1	1	20
TUM/fr1_xyz	7.76	-	20.918	17.957	26.615	**29.233**	2	2	28
TUM/fr2_xyz	1.71	-	15.07	15.070	15.07	**19.589**	2	0	0
TUM/fr1_desk	2.523	-	17.945	18.885	20.211	**21.537**	3	3	33
TUM/fr3_long_office	3.001	12.477	18.208	**20.771**	20.259	20.143	11	6	114
TUM/fr3_nstr_tex_near	1.878	24.077	25.796	27.793	29.014	**32.134**	7	2	27
TUM/fr3_str_tex_far	0.001	27.396	20.668	29.478	30.156	**35.771**	3	2	13
TUM/fr2_desk_person	0.087	-	10.373	8.348	13.324	**13.344**	6	5	80
TUM/fr3_sit_xyz	0	-	8.673	14.843	14.116	**14.117**	1	1	19
Average	1.121	-	18.299	19.407	21.588	**23.130**	-	-	-

time, we set the default batch size to 4 which achieves efficient online updating while offereing sufficient robustness to noisy training data.

B. Performance of Dense Mapping

We compare a complete version of our method with both stage-wise SGD based online adaptation and fusion (denoted as "Ours final") with three variants of our method: 1) without online adaptation and fusion (denoted as "Ours w/o online&fusion"), with normal SGD based online adaptation without fusion (denoted as "Ours w/ normal SGD") and with stage-wise SGD based online adaptation without fusion (denoted as Ours w/ stage-wise SGD). In addition, we compare our method with the state-of-the-art methods, including LSD-SLAM [3] and CNN-SLAM [13]. We evaluated all methods using the percentage of points whose predicted depths are within 10% difference from the corresponding ground truth depths, i.e., $\frac{|D-D_{gt}|}{D_{gt}} < 10\%$. Table II shows the comparison results, and Fig. 10 shows the visualization comparison.

Note that the CNN model of our method is progressively updated on-the-fly. Thus, each evaluated sequence is separated into several sub-sequences and the depth of each sub-sequence is predicted based on the corresponding adapted CNN model. Fig. 9 provides an illustrative example to explain the process of our online training and testing. In the example, the depths of the first 9 keyframes are estimated using the initial pre-trained model W_0. The system starts to collect training data at the 5^{th} keyframe and the model is updated from W_0 to W_1 at the 10^{th} keyframe. After that, depths of keyframes 10 to 16 are estimated using the adapted CNN model W_1. The percentage of correct depth prediction is evaluated for each keyframe and averaged over all keyframes for the evaluated sequence. The 8^{th} and 10^{th} columns show detailed online adaptation profiles including the total numbers of batches collected and used during each video sequence and the total number of iterations for CNN model tuning.

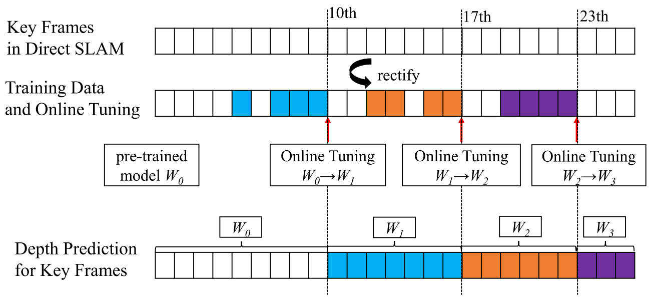

10th
17th
23th
Key Frames in Direct SLAM
rectify
Training Data and Online Tuning
pre-trained model W_0
Online Tuning $W_0 \rightarrow W_1$
Online Tuning $W_1 \rightarrow W_2$
Online Tuning $W_2 \rightarrow W_3$
W_0
W_1
W_2
W_3
Depth Prediction for Key Frames

Fig. 9. Illustration of online tuning and testing for different frames in an evaluated video sequence.

Five observations could be made from the results. First, CNN-based methods, i.e., CNN-SLAM [13] and variants of our methods, could achieve much better reconstruction performance than the semi-dense mapping method LSD-SLAM [3]. Second, by comparing the results of Ours w/o online&fusion and Ours w/ normal SGD online adaptation we observe that online adapting the CNN model could enhance the mapping accuracy for various scenes and thus our method w/ normal SGD achieves 6.1% performance improvement compared with Ours w/o online&fusion. Third, the proposed stage-wise SGD could further improve the performance of normal SGD based online adaptation by 11.23%, i.e., the average completeness of "Ours w/ stage-wise SGD" is 11.23% higher than that of "Ours w/ normal SGD". Fourth, fusing CNN with direct monocular SLAM with absolute depth estimation could further improve the overall depth completeness by 7.2% than Ours w/ stage-wise SGD. Finally, our method with both stage-wise SGD based online adaptation and fusion outperforms CNN-SLAM for most video sequences, demonstrating the superiority of our integrated

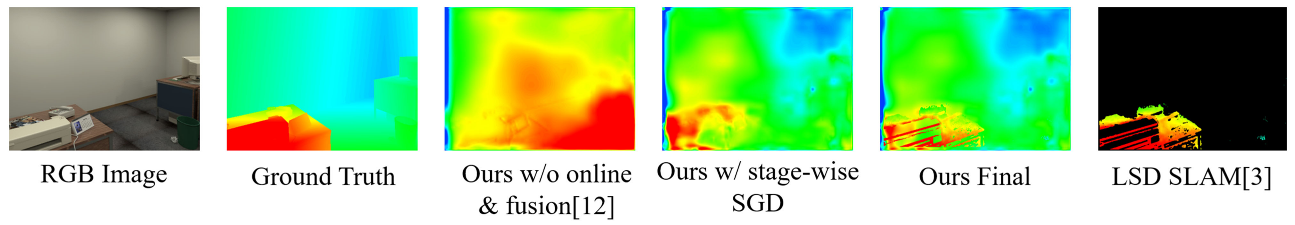


Fig. 10. Comparison in terms of depth map accuracy and density among (from the left) the ground-truth, our method w/o online&fusion [12], our method with only online adaptation and without fusion, our method with both online adaptation and fusion, and LSD-SLAM [3], on an exemplar keyframe from the *ICL-NUIM* dataset [37].

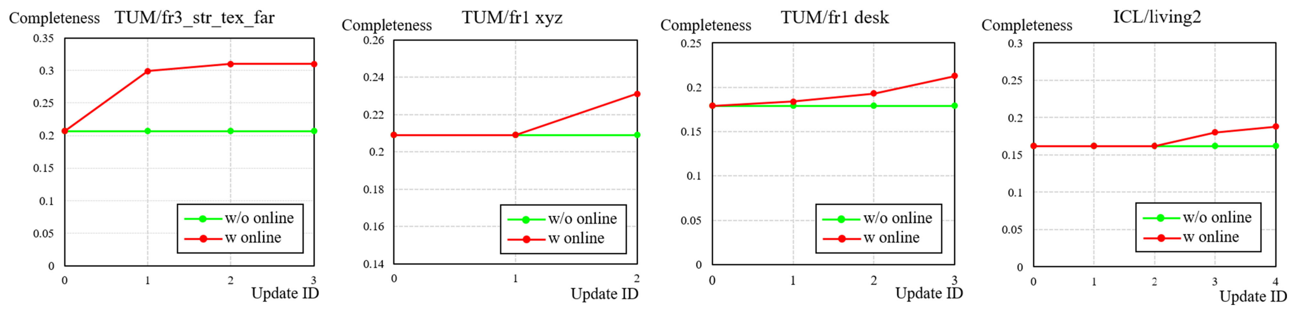


Fig. 11. Completeness with respect to the update ID for four exemplar video sequences.

approaches to those proposed in [13]. The last three columns show detailed online adaptation profiles including the total numbers of batches collected and used during each video sequence and the total number of iterations for CNN model tuning.

We also illustrate the performance improvement for each update for four exemplar video sequences. Specifically, we evaluated the completeness of each online updated CNN model(i.e., percentage of depth predictions with $\frac{|D-D_{gt}|}{D_{gt}} < 10\%$) throughout the entire testing keyframes of the ICL-NUIM [37] and TUM [36] datasets. It is worth mentioning that the evaluation strategy is slightly different from that reported in Table II. In Table II, for a fair comparison of our method with LSD-SLAM [3] and CNN-SLAM [13], each online updated model is only evaluated using keyframes during the period when the model is available before being replaced by the next updated model, as illustrated in Fig. 9. An online updated model ready at time t would not be used for depth estimation before time t, neither after being replaced with next update. In this experiment, we aim at comparing the performance of each update, thus it is important to use identical testing images for models under comparison. Therefore, we used the entire keyframes of the two datasets [36], [37] for the evaluation. In practice, the performance improvement for each update would be case dependent, i.e., the completeness varies for different video sequences and for different update. Fig. 11 shows the completeness with respect to update ID for four exemplar video sequences. For TUM/fr3_str_tex_far [36] the performance improves significantly for the first online update and the improvement for the 2^{nd} and 3^{rd} updates are relatively small. In comparison, for the other three exemplar sequences, the improvements are negligible for the 1^{st} online update while become more significant for the subsequent updates.

C. Performance of Tracking Accuracy

We evaluate the tracking accuracy based on a widely-used metric *Absolute Trajectory Error* (ATE), which is computed as the root mean square error between the estimated translation and the ground truth for each evaluated video. In this experiment, we first examined the impact of online adaptation and fusion on ATE by comparing a complete version of our method with two variants of our method on the ICL-NUIM [37] and TUM datasets [36] : 1) our method excluding online adaptation (denoted as "Ours w/o online") and 2) our method excluding fusion (denoted as "Ours w/o fusion"). For both variants of our method, the scale regression method is applied to the initialization step of SLAM which largely completes before the online adaptation step. Experimental results demonstrate that online adaptation and fusion do not affect the final tracking accuracy too much. Because the tracking accuracy results of "Ours w/o online" and "Ours w/o fusion" are identical as those of Ours, we do not list them separately in Table III. This result implies that fusion and online adaptation mainly improve depth estimation accuracy of textureless regions and have little impact on the high gradient pixels. This result is also quite understandable because direct tracking mainly relies on depth estimation of high gradient pixels and, therefore, excluding online adaptation and fusion would not degrade the tracking accuracy. Furthermore, absolute scale regression is highly important for the tracking accuracy and the proposed scale regression could significantly improve the tracking results. Table III compares our method with scale regression (denoted as "Ours") with LSD-SLAM [3], the results show that due to the aforementioned absolute scale ambiguity problem, our method always achieves a much smaller tracking error than LSD-SLAM [3], i.e., the average ATE of our method is 0.207 m vs 0.510 m for LSD-SLAM [3].

TABLE III
COMPARISON OF ABSOLUTE TRAJECTORY ERROR (M) ON ICL-NUIM AND TUM DATASETS

	Abs. Tracking Error[m]				Regressed Scale		
	LSD-SLAM [3]	CNN-SLAM [13]	CNN initialization	Ours	Regressed based on Ground truth	Regressed based on CNN	$\frac{Scale_{gt}}{Scale_{cnn}}$
ICL/office0	0.514	0.266	0.403	**0.243**	3.42	3.03	1.12
ICL/office1	0.622	0.157	**0.101**	0.248	2.64	3.26	0.81
ICL/office2	0.792	0.213	**0.204**	0.206	3.40	3.39	1.00
ICL/office3	0.541	-	0.275	**0.085**	2.65	2.75	0.96
ICL/living0	0.414	**0.196**	0.642	0.302	2.82	3.19	0.88
ICL/living1	0.130	**0.059**	0.170	0.169	1.75	2.71	0.65
ICL/living2	0.665	0.323	0.148	**0.064**	3.04	2.92	1.04
ICL/living3	0.664	-	0.204	**0.138**	2.44	2.70	0.90
TUM/fr1_xyz	0.028	-	**0.026**	0.027	1.18	1.22	0.97
TUM/fr2_xyz	0.077	-	0.226	**0.026**	1.27	1.24	1.02
TUM/fr1_desk	**0.062**	-	0.798	0.192	1.23	0.96	1.28
TUM/fr3_long_office	1.207	**0.542**	0.664	0.635	2.45	1.70	1.44
TUM/fr3_nstr_tex_near	0.382	0.243	0.508	**0.111**	1.27	1.23	1.03
TUM/fr3_str_tex_far	0.87	0.214	0.133	**0.105**	1.86	1.81	1.03
TUM/fr2_desk_person	1.021	-	0.695	**0.628**	1.93	1.23	1.58
TUM/fr3_sit_xyz	0.170	-	0.171	**0.139**	2.51	1.80	1.39
Average	0.510	-	0.335	**0.207**	-	-	-

TABLE IV
RUNTIME OF EACH MODULE IN OUR SYSTEM

Processing Unit	Module	Runtime
CPU	Direct SLAM	36ms/frame
GPU1	Depth Prediction	25ms/frame
GPU2	Online Tuning	86ms/iteration
CPU+GPU1+GPU2	Entire System	36ms/frame

We also provide a direct comparison of our scale regression method with the CNN-based initialization method (denoted as "CNN initialization") which is used in CNN-SLAM [13] for addressing the scale ambiguity problem. Specifically, in CNN-SLAM the authors addressed the scale ambiguity problem by initializing depth estimation for pixels with unknown depth in each keyframe using the CNN predicted results and then refine the predictions in the following subsequent pixel matching and triangulation step. The results in Table III show that for the majority of dataset sequences the tracking error of CNN-SLAM [13] is greater than our method, confirming the effectiveness of the proposed absolute scaling factor regression method. In addition, Table III also shows the regressed scale for each test video and compares it with the corresponding ground truth scale. Results show that the majority of the regressed scale values are very close to the ground truth, i.e., $\frac{Scale_{gt}}{Scale_{cnn}}$ is mostly within $0.8 \sim 1.2$.

D. Runtime Analysis

Table IV shows the average runtime of each module of our system based on all evaluated video sequences. The direct SLAM module (including the pose tracking and semi-dense mapping threads) runs at an average of 28 frames per second (fps) on a CPU core (i.e., 36 ms per frame). The depth prediction module takes an average of 25 ms to process a keyframe on a GPU core and the runtime for fusing the results from CNN prediction and direct SLAM is trivial. The process of online CNN update runs on another GPU core and takes about 86 ms for an update iteration. Recall that the diversity constraint for selecting online training samples defined in *Sec. III-C* requires training samples being selected from frames associated with different keyframes. Thus, a complete round of model adaptation takes a duration consisting of at least four keyframes. That is an online adapted model can be obtained at the earliest 4 keyframes after the online adaptation module is invoked. As mentioned in *Sec. III*, before the convergence of the tuning process, the depth of keyframes will still be predicted using the previous CNN model to ensure uninterrupted dense map prediction. In addition, both depth prediction and fusion are sufficiently fast, thus the speed of the entire system mainly depends on the efficiency of direct SLAM, which is 28 fps.

E. Application to Obstacle Avoidance

To demonstrate the effectiveness of our method for practical applications, we integrated our system into a drone for obstacle avoidance in real circumstances. Specifically, our system takes images from a monocular camera and produce its depth prediction in real time. Then we transform the depth map into the ego dynamic space as the method [39], quantize the transformed depth map using the mean shift clustering algorithm [40] and choose the way points based on the method proposed in [41]. Once waypoints are obtained, we input their 3D coordinates in the world frame to a trajectory generator based on the method proposed in [42]. A demo video of automated obstacle avoidance of an AR Parrot drone based on the proposed system is provided in the supplementary material.

V. CONCLUSION AND FUTURE WORK

This paper presents a monocular SLAM system which effectively combines an online-adapted CNN with a direct semi-dense SLAM to achieve real-time, accurate and dense 3D reconstruction. Several novel methods are introduced to solve two long-standing limitations of traditional monocular SLAM, i.e., low map completeness and scale ambiguity, and problems of a pre-trained depth prediction network in arbitrary application scenarios, i.e., significant performance degradation for environments of different types from the training data. Specifically, we propose a stage-wise SGD with a robust online training data collection mechanism and a selective update strategy to facilitate efficient online CNN model adaptation and meanwhile ensure progressively improved CNN model with adequate generalization ability. Depth predictions from adapted CNN are employed to regress an absolute scale for each keyframe based on the RANSAC algorithm. The depth predictions result from both adapted CNN and the refined semi-dense map of the monocular SLAM with an absolute scale are fused via NCC-based voting and inter-keyframe Gaussian fusion for further improving the accuracy of the final reconstruction. Extensive experiments demonstrate the superior performance of the proposed system to the state-of-the-art approaches.

Due to the CNN depth prediction network, our system can continuously predict depth even for camera motion with pure rotational motion and/or fast camera motion. But for cases with pure rotational motion and fast camera motion, the online updating module of our system is unable to collect effective training data, and hence the performance of the depth prediction network cannot be improved for these scenarios. So far the online training can only work for horizontal motion and the baseline within a batch needs to be fixed, which are the two limitations of our current system and will be addressed in our future work. In addition to the demonstrated application for obstacle avoidance of a drone, our method should also be beneficial to several other monocular video based applications, such as augmented reality [9], 3D scene understanding [13] and event summarization from multi-view videos [43]–[45]. We will explore these applications and assess the performance of our method in our future work. In addition, we also plan to investigate binarization of the CNN for further acceleration of depth prediction and online adaptation, more advanced trajectory planning approaches and deployment of the entire system to drones for autonomous navigation.

References

[1] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Proc. 6th IEEE ACM Int. Symp. Mixed Augmented Reality*, 2007, pp. 225–234.

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

[3] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular slam," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 834–849.

[4] A. Dame, V. A. Prisacariu, C. Y. Ren, and I. Reid, "Dense reconstruction using 3D object shape priors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 1288–1295.

[5] S. Sengupta and P. Sturgess, "Semantic octree: Unifying recognition, reconstruction and representation via an octree constrained higher order MRF," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 1874–1879.

[6] F. Chhaya *et al.*, "Monocular reconstruction of vehicles: Combining slam with shape priors," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 5758–5765.

[7] W. N. Greene, K. Ok, P. Lommel, and N. Roy, "Multi-level mapping: Real-time dense monocular slam," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 833–840.

[8] A. Concha and J. Civera, "DPPTAM: Dense piecewise planar tracking and mapping from a monocular sequence," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 5686–5693.

[9] T. Xue, H. Luo, D. Cheng, Z. Yuan, and X. Yang, "Real-time monocular dense mapping for augmented reality," in *Proc. ACM Multimedia Conf., Mountain View*, CA, USA, Oct. 23–27, 2017, pp. 510–518.

[10] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas, "Probabilistic data association for semantic slam," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 1722–1729.

[11] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6602–6611.

[12] R. Garg, G. Carneiro, and I. Reid, "Unsupervised CNN for single view depth estimation: Geometry to the rescue," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 740–756.

[13] K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: Real-time dense monocular slam with learned depth prediction," *in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6565–6574.

[14] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 2320–2327.

[15] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, Mar. 2018.

[16] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct EKF-based approach," in *Proc. IEEE/RSJ Int. Conf. Intell Robots Syst.*, 2015, pp. 298–304.

[17] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *Int. J. Robot. Res.*, vol. 34, no. 3, pp. 314–334, 2015.

[18] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2366–2374.

[19] F. Liu, C. Shen, G. Lin, and I. Reid, "Learning depth from single monocular images using deep convolutional neural fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 2024–2039, Oct. 2016.

[20] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *Proc. IEEE 4th Int. Conf. 3D Vis.*, 2016, pp. 239–248.

[21] R. Phan and D. Androutsos, "Robust semi-automatic depth map generation in unconstrained images and video sequences for 2D to stereoscopic 3D conversion," *IEEE Trans. Multimedia*, vol. 16, no. 1, pp. 122–136, Jan. 2014.

[22] P. Wu, Y. Liu, M. Ye, J. Li, and S. Du, "Fast and adaptive 3D reconstruction with extensively high completeness," *IEEE Trans. Multimedia*, vol. 19, no. 2, pp. 266–278, Feb. 2017.

[23] Y. Guo, F. Sohel, M. Bennamoun, J. Wan, and M. Lu, "An accurate and robust range image registration algorithm for 3D object modeling," *IEEE Trans. Multimedia*, vol. 16, no. 5, pp. 1377–1390, Aug. 2014.

[24] B. Bodinm *et al.*, "Integrating algorithmic parameters into benchmarking and design space exploration in 3D scene understanding," in *Proc. Int. Conf. Parallel Architectures Compilation*, 2016, pp. 57–69.

[25] D. Hoiem, A. A. Efros, and M. Hebert, "Geometric context from a single image," in *Proc. 10th IEEE Int. Conf. Comput. Vis.*, 2005, vol. 1, pp. 654–661.

[26] B. Liu, S. Gould, and D. Koller, "Single image depth estimation from predicted semantic labels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 1253–1260.

[27] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison, "Codeslam-learning a compact, optimisable representation for dense visual slam," in *proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018.

[28] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 1449–1456.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[30] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3431–3440.

[31] M. Pollefeys, R. Koch, and L. Van Gool, "A simple and efficient rectification method for general motion," in *Proc. 7th IEEE Int. Conf. Comput. Vis.,* 1999, vol. 1, pp. 496–501.

[32] J.-Y. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel Corporation*, vol. 5, no. 1–10, p. 4, 2001.

[33] J.-L. Blanco-Claraco, F.-Á. Moreno-Dueñas, and J. González-Jiménez, "The málaga urban dataset: High-rate stereo and lidar in a realistic urban scenario," *Int. J. Robot. Res.*, vol. 33, no. 2, pp. 207–214, 2014.

[34] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[35] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[36] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D slam systems," in P*roc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 573–580.

[37] A. Handa, T. Whelan, J. McDonald, and A. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *Proc. IEEE Intl. Conf. Robot. Automat.*, Hong Kong, China, May 2014, pp. 1524–1531.

[38] H. Alismail, B. Browning, and M. B. Dias, "Evaluating pose estimation methods for stereo visual odometry on robots," in *Proc.11th Int. Conf. Intell. Auton. Syst.*, vol. 3, p. 2, 2010.

[39] J. Minguez, L. Montano, and O. Khatib, "Reactive collision avoidance for navigation with dynamic constraints," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2002, pp. 588–594.

[40] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 8, pp. 790–799, Aug. 1995.

[41] K. Bipin, V. Duggal, and K. M. Krishna, "Autonomous navigation of generic monocular quadcopter in natural environment," in *Proc. IEEE Int. Conf. Robot. Automat.,* 2015, pp. 1063–1070.

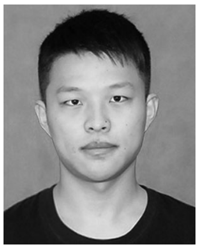
[42] J. Chen, T. Liu, and S. Shen, "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1476–1483.

[43] K. Kumar and D. D. Shrimankar, "F-DES: Fast and deep event summarization," *IEEE Trans. Multimedia*, vol. 20, no. 2, pp. 323–334, Feb. 2018.

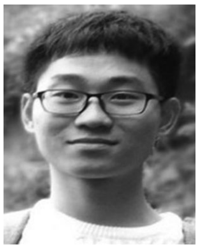
[44] K. Kumar and D. D. Shrimankar, "Deep event learning boost-up approach: Delta," *Multimedia Tools Appl.*, pp. 1–21, Mar. 2018.

[45] K. Kumar, D. D. Shrimankar, and N. Singh, "Event bagging: A novel event summarization approach in multiview surveillance videos," in *Proc. IEEE Int. Conf. Innovations Electron., Signal Process. Commun.*, 2017, pp. 106–111.

Hongcheng Luo received the B.E. degree from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2016. He is currently working toward the Graduate degree with School of Electronic Information and Communications, HUST. He has published two papers on ACM MM 2017. His research interests include monocular simultaneous localization and mapping and augmented reality.



Yang Gao received the B.E. degree from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2017. He is currently working toward the Graduate degree with School of Electronic Information and Communications, HUST. His research interests include monocular simultaneous localization and mapping and augmented reality.



Yuhao Wu received the B.E. degree from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2016. He is currently working toward Graduate degree with the School of Electronic Information and Communications, HUST. His research interests include monocular simultaneous localization and mapping and robotics.

Chunyuan Liao received the Ph.D. degree from the University of Maryland, College Park, MD, USA, in 2008. He was a Research Scientist with Fuji Xerox Palo Alto Laboratory, Silicon Valley, where he was granted significant achievement Awards three times and ACM conference Awards twice due to his outstanding research in Augmented Reality. In the year of 2012, He founded HiScene, a leading Chinese Augmented Reality technology provider. He is one of the national distinguished experts in Chinese government's "The Thousand Talents Plan." He has published more than 60 technical papers, including ACM Transactions on CHI, UIST, ICCV, ACM MM, CHI, etc., and holds 10+ U.S. Patents.

Xin Yang (M'14) received the Ph.D. degree from the University of California, Santa Barbara, (UCSB), Santa Barbara, CA, USA, in 2013. She worked as a Postdoctoral in Learning-based Multimedia Lab with UCSB (2013–2014). She is currently an Associate Professor with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China. She has published more than 30 technical papers, including TPAMI, TVCG, MIA, ACM MM, MICCAI, ISMAR, etc., co-authored two books and holds 2 U.S. Patents. Her research interests include monocular simultaneous localization and mapping, augmented reality, and medical image analysis. She is a member of ACM.

Kwang-Ting (Tim) Cheng (F'00) received the Graduate degree from the University of California, Berkeley, Berkeley, CA, USA, in 1988, and the Ph.D. degree in electrical engineering and computer sciences. Before joining Hong Kong University of Science and Technology (HKUST), he was a Professor of Electrical and Computer Engineering with the University of California, Santa Barbara (UCSB), where he served since 1993. Prior to teaching at UCSB, he spent five years with AT&T Bell Laboratories. From 1992 to 2002, he was a Founding Director of the Computer Engineering Program with UCSB. From 2005 to 2008, he was a Chair of the Department of ECE, an Acting Associate Vice-Chancellor for Research in 2013. From 2014 to 2016, he was an Associate Vice-Chancellor for Research, where he helped oversee the research development, infrastructure, and compliance of UCSBs research enterprise with over US$200 million extramural research funding. He was the Director of the US Department of Defense Multidisciplinary University Research Initiative Center for 3D Hybrid Circuits, which integrated CMOS and nano-memristors for future computing systems. He has published more than 400 technical papers, received 11 best paper awards, co-authored 5 books, held 12 US patents, and transferred several of his inventions into successful commercial products. He was the Editor-in-Chief of IEEE Design and Test of Computers, is on the boards of IEEE Council on Electronic Design Automations Board of Governors and IEEE Computer Societies Publications Board, and on various technology advisory or working groups including the International Technology Roadmap for Semiconductors. He joined HKUST in May 2016 and is currently the Dean of Engineering in concurrence with his appointment as a Chair Professor jointly in the Department of Electronic and Computer Engineering and in the Department of Computer Science and Engineering.