2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

A Front-End for Dense Monocular SLAM using a Learned Outlier Mask Prior

Yihao Zhang and John J. Leonard

***Abstract*— Recent achievements in depth prediction from a single RGB image have powered the new research area of combining convolutional neural networks (CNNs) with classical simultaneous localization and mapping (SLAM) algorithms. The depth prediction from a CNN provides a reasonable initial point in the optimization process in the traditional SLAM algorithms, while the SLAM algorithms further improve the CNN prediction online. However, most of the current CNN-SLAM approaches have only taken advantage of the depth prediction but not yet other products from a CNN. In this work, we explore the use of the outlier mask, a by-product from unsupervised learning of depth from video, as a prior in a classical probability model for depth estimate fusion to step up the outlier-resistant tracking performance of a SLAM front-end. On the other hand, some of the previous CNN-SLAM work builds on feature-based sparse SLAM methods, wasting the per-pixel dense prediction from a CNN. In contrast to these sparse methods, we devise a dense CNN-assisted SLAM front-end that is implementable with TensorFlow and evaluate it on both indoor and outdoor datasets.**

I. Introduction

Simultaneous localization and mapping (SLAM) has been a long-standing problem [1]. In particular, monocular SLAM requires simultaneously solving for the depths of points in the images and the poses of the camera along its trajectory given only the sequence of images that the camera has captured as measurements. The problem is difficult because the data association between a pixel in a frame and the corresponding pixel in a second frame is also unknown along with the depths and the camera poses. Advances in computer vision and the recent success in the area of single-image depth prediction with CNNs [2]–[7] have brought us exciting new opportunities in monocular SLAM. The depth prediction from a CNN can provide a reasonable guess for one of the unknowns in the SLAM problem, the depth, helping to eliminate this unknown from the problem. Recent monocular SLAM methods that take advantage of the learned depth map often use it as an initial point in the depth and pose optimization scheme [8]–[14]. They have shown surpassing performance over the conventional SLAM methods in their accuracy and their robustness to ill conditions (e.g. pure rotations and textureless environments) [8]–[10].

A number of these CNN-SLAM approaches [8]–[11] build upon existing mature SLAM systems and integrate the CNN depth prediction into those systems. These methods show high reliability due to their foundation on tested SLAM technologies. In the meantime, they also inherit the underlying map representation, which is typically a sparse representation [9]–[11], from the SLAM systems that they build upon.

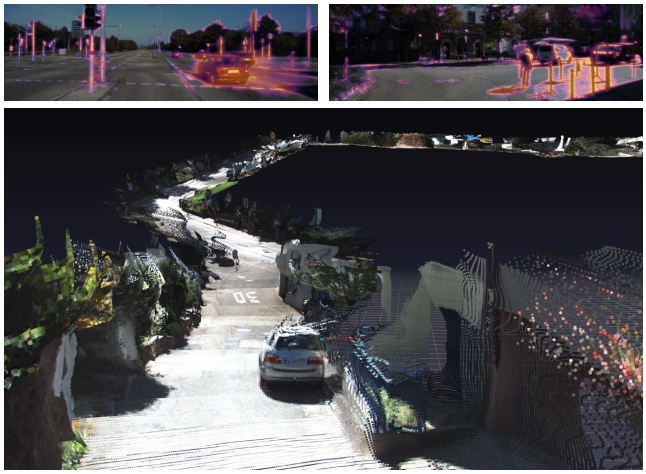

Fig. 1: Top row: The learned outlier mask. The brighter and yellower regions indicate the predicted outliers in the photometric consistency due to dynamic objects and static occlusions. Bottom row: A dense point cloud of KITTI [15] odometry sequence 10 generated by our SLAM front-end. (Far points are excluded for cleaner visualization.)

Some recent work has looked beyond merely utilizing CNNs to initialize the depth estimates. The line of work [12]–[14] following CodeSLAM [12] experimented reducing the depth map onto a lower dimensional manifold by using the learned CNN decoder for the purpose of efficient joint optimization on multiple frames. D3VO [11] also started to incorporate not only the learned depth but also the learned depth uncertainty into the SLAM pipeline.

We present a dense monocular CNN-assisted SLAM front-end that is fully implementable with TensorFlow. The focus is on the pose estimation performance without the back-end inference. We show that when the CNN training objective is aligned with the online pose estimation objective, the outlier mask, a by-product learned from the training process can be used together with a classical probability model to improve the pose estimation accuracy through better handling the outliers due to static occlusions and dynamic objects. As semantic reconstruction becomes more and more important for visual systems, our SLAM front-end also processes the semantic segmentation images using forward propagation to improve their quality. We evaluate our method on both indoor and outdoor datasets.

Yihao Zhang and John Leonard are with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 USA {yihaozh, jleonard}@mit.edu

This work was supported by ONR MURI grant N00014-19-1-2571, ONR grant N00014-18-1-2832, and ARPA-E award DE-AR0001218 under the DIFFERENTIATE Program.

II. Related Work

A. Classical Monocular SLAM Methods

There exist two branches of monocular SLAM methods, feature-based methods and direct methods. Represented by ORB-SLAM [16], the pose estimation in feature-based methods is done by first finding the corresponding points between two frames and then computing their relative pose given the correspondences. These methods create a sparse map since only a few hundred key points in an image are involved in the computation, while all the other pixels are discarded. Direct methods [17]–[20] rely on direct image alignment, which is based on photometric consistency, for pose estimation. These methods can be used to construct a sparse, semi-dense, or dense map depending on the choice of points of interest. The tracking module in a typical direct visual odometry pipeline [19] tracks the current frame against a previously established keyframe which is associated with a depth map initialized by given statistics (e.g. average scene depth). The tracking step commonly includes performing direct image alignment for pose estimation between the current frame and the keyframe and updating the keyframe depth map given the estimated pose. In this work, we leverage several techniques from the previous work in direct methods with modifications to build our dense SLAM front-end pipeline.

B. SLAM using Learned Depth

The first work of combining SLAM with CNN depth prediction is CNN-SLAM [8], where the predicted depth images from a CNN initialize the keyframe depth estimates in LSD-SLAM (large-scale direct monocular SLAM) [19], replacing the random initialization in the original LSD-SLAM. Their results showed that monocular SLAM armed with the CNN depth prediction could better recover the absolute scale of a scene, densify the depth estimates in texture-less regions, and deal with degenerative camera motions. CNN-SVO [9] shared the same idea with CNN-SLAM but instead of LSD-SLAM it was based on SVO (semi-direct monocular visual odometry) [21].

Later CNN-SLAM approaches become more sophisticated in the use of CNNs. DVSO [10], which is built upon monocular DSO (direct sparse odometry) [20], forms a virtual stereo loss term in the sparse bundle adjustment using the predicted left and right stereo disparity images so that the CNN depth prediction does not only provide an initial point at the start of the optimization but also strengthens the quality of the optimized depth map through the stereo loss term. CodeSLAM [12] uses a variational auto-encoder (VAE) [22] to learn a lower-dimensional latent representation of depth images. It updates the lower-dimensional latent representation instead of the depth image itself to achieve efficient optimization of the dense depth image. SceneCode [13] and DeepFactors [14] extends CodeSLAM by incorporating an additional semantic consistency loss and a feature-based re-projection loss to further improve the system performance.

DVSO [10] and CodeSLAM [12] explored the utility of CNNs in different ways, not only using them for depth estimate initialization. We present a third way that is fusing a learned outlier mask as a prior into a classical probability model to handle outliers in the photometric consistency loss caused by dynamic objects and static occlusions in the environment. Our method can be an add-on to their approaches.

C. Depth Prediction from a Single Image

CNNs can be trained in either a supervised manner [2] or an unsupervised manner [3]–[7] to enable depth prediction. We are interested in unsupervised training with monocular video [4]–[7]. We found that the outlier mask, a by-product from this training process, could be engineered into a SLAM system to improve its tracking performance.

III. Preliminaries

A. Photometric Consistency Loss

At the core of direct methods (e.g. [20]) and unsupervised learning of depth approaches (e.g. [4]) is the photometric consistency which states that if a pixel in one frame and a pixel in another frame correspond to the same 3D point, they should have the same intensity in the images. Therefore, the photometric consistency loss is defined in our work as:

$$L_{pho}(I', I) = \frac{1}{|V|} \sum_{p \in \Omega(I)} W(p) \| I'(\Pi(p)) - I(p) \|_m \quad (1)$$

$$\Pi(p) = \Pi(p, D(p), R, t) = \begin{bmatrix} u' \\ v' \end{bmatrix} \quad (2)$$

$$d' \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = K \begin{bmatrix} R & t \end{bmatrix} homo\left(K^{-1} D(p) \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \right) \quad (3)$$

where I and I' are two nearby frames, $p = \begin{bmatrix} u & v \end{bmatrix}^T \in \Omega(I)$ is a pixel in the image space $\Omega(I)$, $W(p)$ is the weight applied to the loss term at pixel p, V is the set of valid pixels projected within the image boundaries, $\| \cdot \|_m$ is the m-norm, and $\Pi(p)$ is the perspective projection function, which is expressed in detail in (3) where $D(p)$ is the depth at pixel p, R and t are the rotation matrix and the translation vector from I to I', K is the 3-by-3 camera instrinsic matrix, $homo$ denotes transforming to the homogeneous coordinates, and d' is the projected depth in frame I'. The 1-norm is typically used in CNN training for robustness to outliers, while the 2-norm is standard in direct visual odometry methods. $\Pi(p)$ yields non-integer values, which are inappropriate to index pixels, and therefore we use bilinear interpolation to interpolate between pixels. The bilinear interpolation also retains the differentiability of the photometric loss for use in gradient-based optimization methods. In addition, the projected pixel $\Pi(p)$ can fall outside the image boundaries. We exclude these invalid pixels and normalize the photometric loss by the number of valid pixels $|V|$, as done in [6]. The photometric loss in (1) can be easily extended to RGB-channels often used in CNN training.

B. Learning Depth and Outlier Mask from Video

Prior work [4]–[6] has discussed in detail how to learn depth and ego-motion from monocular video. We will focus on discussing the outlier mask. The closest definition of the outlier mask in our work is the explainability mask in [4]. For each pixel in the image, an outlier mask value within $[0, 1]$ is predicted. This value serves as the weight term $W(p)$ in (1) during training. Intuitively, if a pixel corresponds to a 3D point on a dynamic object or on an occlusion surface, being an outlier, the photometric consistency will be violated at that pixel, so its photometric consistency loss should not be counted and $W(p)$ will tend to zero. In order to avoid the trivial case of the learned outlier mask being assigned to zero for every pixel during training, a cross-entropy loss with an all-ones mask is employed as regularization [4]. Unlike [4] where the outlier mask is predicted from the ego-motion network given the two frames for pose prediction, our model ties the mask prediction to the depth network. In this way, the mask can be predicted from a single image. The detailed training procedures that we followed are in [23].

IV. Method

The main system pipeline is described in this section. An overview of the pipeline is given in Algorithm 1. The system is a SLAM front-end. It follows the paradigm of alternating between mapping and tracking [24]. The final outputs are an odometry chain, the depth maps and semantically segmented images for the keyframes. The detailed computing procedures are illustrated in the following subsections.

Algorithm 1 System Overview

Input: image sequence $\{I_i\}$, $i = 1, ..., n$
Output: camera poses $\{T_i\}$ for all frames, depth maps $\{D_j\}_{kf}$ and semantically segmentated images $\{S_j\}_{kf}$ for keyframes, $\{j\} \subseteq \{1, ..., n\}$

function INSERT A KEYFRAME (I)
  Predict the depth map, outlier mask, and semantic segmentation for I.
  Initialize the probability model parameters α, β, μ, and σ for each pixel in I using the predicted depth map and outlier mask.
end function

INSERT A KEYFRAME (I_1)
for i = 2 to n **do**
  Estimate T_i for I_i against the keyframe.
  Update the depth map of the keyframe given T_i.
  Update α, β, μ, and σ for each pixel in the keyframe.
  if the keyframe criteria are satisfied **then**
    INSERT A KEYFRAME (I_i)
    Fuse the semantic class probabilities between the last keyframe and this keyframe.
  end if
end for

A. Keyframe Insertion

A frame is inserted as a keyframe when the system initializes on the first frame of the sequence or when the frame has passed the keyframe criteria (IV-E). A keyframe is associated with multiple attributes, a pose, a color image, a gray-scale image, a semantic segmentation image, a depth map, an outlier mask, and a map of the parameters in the probability model that we will use. Upon the keyframe insertion, the color image is fed into our CNN models to predict the semantic segmentation image, the depth map, and the outlier mask. The outlier mask and the depth map are used to compute the prior distributions for the probability model illustrated next.

Suppose we are at time step k and have collected a depth measurement d_k for a pixel. We can model the distribution on this depth measurement as the following [25], [26]:

$$p(d_k|\hat{d}, \rho) = \rho\mathcal{N}(d_k|\hat{d}, \tau_k^2) + (1-\rho)\mathcal{U}(d_k|d_{min}, d_{max}) \quad (4)$$

where $\rho \in [0, 1]$ is the inlier probability of that measurement, $\hat{d}$ is the true depth, τ_k^2 is the variance for an inlier depth measurement in a Gaussian probability model, $[d_{min}, d_{max}]$ is the interval where an outlier depth measurement is uniformly distributed. Essentially, we assume if the depth measurement is an inlier, it follows a Gaussian distribution, and otherwise, it should follow a uniform distribution. The inlier ratio ρ is modeled to follow a Beta distribution parameterized by α and β. This model (4) is used for depth estimate fusion in [26] assuming camera poses are given a priori. Evidence of why this model is sound can be found in [26].

To perform the posterior update using this probability model (i.e. to compute $p(\hat{d}, \rho|d_1, ..., d_k)$), we need a prior Gaussian distribution on $\hat{d}$ and a prior Beta distribution on ρ. For the Beta prior parameterized by α_0 and β_0, we set the ratio of α_0 to $\alpha_0 + \beta_0$, which is the Beta mean $\mathbb{E}[\rho]$, to be the predicted outlier mask value. We further set $\alpha_0 + \beta_0$ as a fixed tuning parameter in order to compute the values of α_0 and β_0. This tuning parameter essentially controls the variance of the Beta distribution.

The prior Gaussian distribution has its mean μ_0 set equal to the depth prediction from the CNN and its standard deviation σ_0 set equal to a percentage of the predicted depth. This percentage is set as another tuning parameter. Alternatively, the standard deviation can be predicted by a CNN using the concept of heteroscedastic aleatoric uncertainty [11], [27]. However, this would introduce extra complexity in the coupling between the SLAM system and the CNN, making the network training process difficult. We thus opted not to use a predicted standard deviation.

The prior parameters, α_0, β_0, μ_0, and σ_0, will be updated in the posterior computation (IV-D). We denote the updated parameters at the current time step as α_k, β_k, μ_k, and σ_k.

B. Pose Estimation

After the keyframe has been established, subsequent frames are tracked against the keyframe. The pose of a frame with respect to the keyframe is estimated with direct image

alignment [19], [20]:

$$\min_{T,a,b} \left\{ L_{pho}(I_f, aI_{kf} + b) + w[(a-1)^2 + b^2] \right\} \quad (5)$$

where $T = [R, t]$ is the pose matrix, a and b are the affine lighting transformation coefficients [20], and I_f and I_{kf} are the gray-scale images of the current frame and the keyframe. The second term in (5) is the regularization on the affine lighting coefficients [11]. This regularization loss is weighted by w. The depth map involved in the computation of L_{pho}, as seen in (2), consists of the Gaussian mean μ_{k-1} for every pixel. The weight $W(p)$ used in the computation, as seen in (1), is the mean inlier probability $\mathbb{E}[\rho]$ which is the ratio of α_{k-1} to $\alpha_{k-1} + \beta_{k-1}$ for pixel p. In other words, we down-weight the photometric loss at a pixel if the pixel is likely to be an outlier according to our probability model.

The optimization in (5) is solved by Newton's method to take advantage of the built-in Hessian matrix computation in TensorFlow. To further reduce the outlier effects, we use the Huber loss function, implemented as iteratively re-weighted least squares, for the photometric loss L_{pho}. During the iterative optimization, the increments are accumulated on the $SE(3)$ pose matrix [20].

The pose estimation is iteratively performed on an image pyramid which accommodates the resolution of the image and the camera motion aggressiveness. In addition, the initial pose estimate at the beginning of the iteration is computed by a constant motion model given the past pose estimates [20].

C. Discrete Depth Search

After the pose is estimated, we re-estimate the depth map D_{kf} associated with the keyframe given the estimated pose and the estimated affine lighting transformation coefficients a and b. For each pixel in I_{kf}, we maximize the normalized cross-correlation (NCC) between I_{kf} and I_f over a 3×3 patch (ω) around that pixel:

$$\max_{D_{kf}(\omega)} \frac{\sum_{p \in \omega} I'_{kf}(p) I_f(\Pi(p))}{\sqrt{\sum_{p \in \omega} I'_{kf}(p)^2} \sqrt{\sum_{p \in \omega} I_f(\Pi(p))^2}} \quad (6)$$

where $I'_{kf} = aI_{kf} + b$. This optimization is solved by the simple gridding method. For each pixel in ω, the depth range of two standard deviations ($2\sigma_{k-1}$) above and below the Gaussian mean μ_{k-1} is discretized into a few points. We discretize the entire patch ($D_{kf}(\omega)$) together to be compatible with the batch processing in TensorFlow. The center pixel is assigned with the discrete depth point that maximizes the NCC score (6). This re-estimated depth is treated as a depth measurement (d_k) in the posterior update.

D. Posterior Update

The posterior given our measurement probability model in (4) is non-trivial, but the authors of [26] showed that the posterior can be approximated as the product of a Beta distribution and a Gaussian distribution:

$$q(\hat{d}, \rho) = Beta(\rho | \alpha_k, \beta_k) \mathcal{N}(\hat{d} | \mu_k, \sigma_k^2) \quad (7)$$

Therefore, computing the posterior only involves updating the four parameters, α_k, β_k, μ_k, and σ_k from their values at time step $k-1$. The detailed update method can be found in the supplementary material of [26]. During the update, the depth measurement standard deviation τ_k in (4) is needed. In [25], this measurement uncertainty is obtained by assuming a standard deviation of one pixel in the image during the depth search step and back-projecting this one pixel uncertainty to depth uncertainty using geometry. We instead use the simple approximation found in [18] to back-propagate the one pixel uncertainty:

$$\tau_k^2 = (\frac{\delta_d}{\delta_\lambda})^2 \tau_\lambda^2 \quad (8)$$

where τ_λ is the assumed one-pixel standard deviation, δ_d is the depth search range in IV-C, and δ_λ is the corresponding pixel search range along the epipolar line in image I_f. In this way, we back-propagate the uncertainty using a numerically approximated Jacobian of the projection function.

E. Keyframe Criteria and Keyframe Propagation

The steps described in IV-B – IV-D are repeated for each incoming frame until the current frame is too far from the keyframe. We use two criteria to decide whether to insert the current frame as a new keyframe. First, we set a threshold for the maximum number of frames allowed to pass after a keyframe. Second, we compute the number of valid pixels projected within the image boundaries, which is $|V|$ in (1), in the last iteration of the pose estimation step (IV-B). If the percentage of valid pixels is below a threshold, we insert a new keyframe. More sophisticated criteria can be found in [20]. However, the two criteria mentioned here are enough for us to evaluate our front-end.

After a frame is chosen to be a keyframe, we proceed back from the keyframe insertion step (IV-A). The estimated pose of the frame is stored as the keyframe pose. The semantic labels in the form of class probabilities for pixels that have high inlier probabilities in the last keyframe are propagated onto the new keyframe using the perspective projection function (2) with the latest depth map estimate and the relative pose estimate. The propagated class probabilities are fused with the newly predicted softmax probabilities [28]:

$$P(c | I_{0,\dots,j}) = \frac{1}{Z} P(c | I_{0,\dots,j-1}) P(O = c | I_j) \quad (9)$$

where j indexes keyframes, $P(O = c | I_j)$ is the predicted softmax probability at a pixel for class c, $P(c | I_{0,\dots,j-1})$ is the class probability propagated from the last keyframe, and $P(c | I_{0,\dots,j})$ is the fused class probability.

V. Results

We evaluate our SLAM front-end on the KITTI dataset [15] and the ScanNet dataset [29]. KITTI is an outdoor driving dataset commonly used for evaluating odometry systems, whereas the indoor dataset, ScanNet, provides various ground truth information that is useful for evaluating intermediate system results.

Fig. 2: Visualization of the predicted outlier mask. The darker regions indicate inlier pixels, whereas the brighter and yellower regions represent the predicted outlier pixels. Dynamic objects such as cars and people, and static occlusions such as poles and object boundaries are clearly identified.

A. Training on KITTI

For the KITTI dataset, the prior work on unsupervised learning of depth provides high quality trained weights. As our network architecture is compatible with the one in [5], we used their publicly released weights to initialize the weights in our depth network. The semantic segmentation network was trained on the Cityscapes dataset [30]. We followed [23] to further train the depth network on Eigen et al.'s training split [31] (a subset of KITTI raw data) to enable the outlier mask prediction. Our depth network and semantic segmentation network both have the DispNet architecture [32] for simplicity and run on 128×416 images.

We provide some visualization of the outlier mask predicted from a single image in Fig. 1 and Fig. 2. As seen in the visualization, the mask is concentrated on dynamic objects, such as people and cars, and object boundaries, which are usually occlusions since the scene behind these boundaries is typically revealed in the next frame. There are also cases where the mask is on thin surface textures (e.g. the white lane markers on the road). These regions tend to have high photometric consistency error because it is difficult to project the pixels exactly to match these thin textures due to the errors in the predicted pose and depth. The network has learned to down-weight the photometric consistency loss at these difficult image regions using the mask. Overall, the outlier mask predicted from a single image looks similar to the explainability mask predicted from two nearby frames in [4]. It is reasonable to set the expected inlier probability $\mathbb{E}[\rho]$ in the prior to be the outlier mask value (IV-A).

B. Trajectory Evaluation on KITTI

We evaluated the performance of our method on the KITTI odometry sequences 09 and 10 which were excluded in the network training set (for both our training and the training of the public checkpoint [5]). Our pipeline is only a front-end without any back-end smoothing, so drift is accumulated more quickly in our system than in a full SLAM system. To fairly evaluate our system, we chose the evaluation method in [4], where the odometry system is run on 5-frame snippets, and the mean absolute trajectory error (ATE) is computed over all snippets. For each snippet, the first frame was inserted as the reference keyframe and the following four frames were tracked against that keyframe. We compare five different settings: (1) the full system, (2) the system without the posterior update, (3) with the posterior update but using an all-ones mask instead of the predicted outlier mask as the prior, (4) with the posterior update and the outlier mask prior but without down-weighting during the pose estimation, and (5) without the posterior update and down-weighting.

TABLE I: Absolute Trajectory Error (ATE RMSE) on the KITTI odometry sequences 09 and 10 computed on 5-frame snippets with the evaluation method in [4] (lower is better). O.M., D.W., and P.U. stand for outlier mask, down-weighting in the pose estimation, and posterior update respectively.

Method	Seq. 09	Seq. 10
ORB-SLAM (full)	0.014 ± 0.008	0.012 ± 0.011
ORB-SLAM (short)	0.064 ± 0.141	0.064 ± 0.130
O.M.+P.U.+D.W.	$\mathbf{0.060 \pm 0.140}$	$\mathbf{0.035 \pm 0.064}$
O.M.+D.W.	0.063 ± 0.152	0.036 ± 0.075
P.U.+D.W.	0.077 ± 0.175	0.045 ± 0.090
O.M.+P.U.	0.075 ± 0.167	0.045 ± 0.090
None	0.075 ± 0.168	0.044 ± 0.090

TABLE II: Absolute Trajectory Error (ATE RMSE) on four KITTI raw sequences (2011_09_26) for the full system and the system without the posterior update and down-weighting.

Method	0009	0046	0059	0084
Full	**4.09**	**0.59**	**2.78**	**3.35**
None	4.88	0.64	2.98	3.90

The public evaluation code from [4] was used to obtain our results in Table I. The ORB-SLAM results (from [4]) were obtained by running ORB-SLAM [16] only on the 5-frame snippets (ORB-SLAM short), and by running ORB-SLAM on the full sequence and chopping the full estimated trajectory into 5-frame snippets (ORB-SLAM full).

As in Table I, the most significant improvement is achieved by using the outlier mask to down-weight the photometric consistency loss (None vs. O.M.+D.W.). This is not surprising since the outlier mask was specifically learned to down-weight pixels to improve pose estimation. The posterior update further brings the error down (O.M.+D.W. vs. O.M.+P.U.+D.W.) through fusing the depth estimates probabilistically and updating the inlier probability for down-weighting. Without the outlier mask prior, the probability model alone (P.U.+D.W.) has limited performance, which demonstrates the importance of imposing such a prior.

Our front-end surpassed the performance of ORB-SLAM (short) in pose estimation. On sequence 09, the three components (O.M., P.U., and D.W.) successfully reduced the error to a level below the ORB-SLAM error. On sequence 10, our front-end is approaching the performance of a full SLAM system which performs joint optimization (bundle adjustment) on multiple keyframes.

We also evaluated the full system compared to the system without the three components on sequences from KITTI raw in Eigen et al.'s test split (excluded during training). evo [33] was used to perform $sim(3)$ alignment with the ground truth trajectory and compute the estimated trajectory RMSE. In Table II, the full system outperforms the baseline system, in agreement with the KITTI odometry results (Table I).

We provide visualization of the dense point cloud gener-

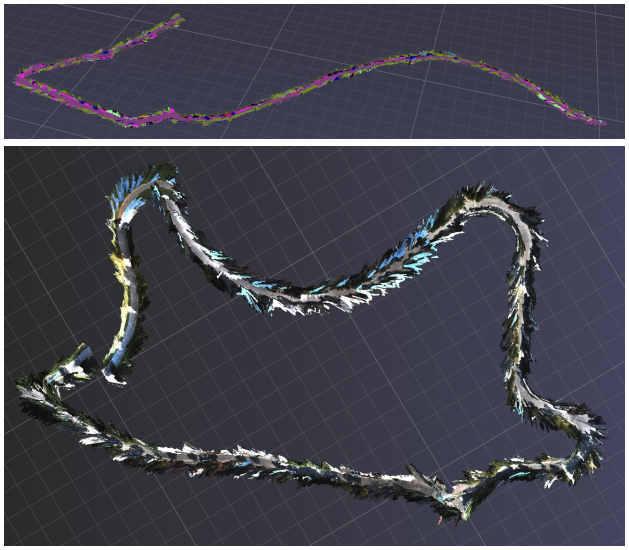

Fig. 3: Top row: A semantically labeled dense point cloud of KITTI odometry sequence 10 generated by our front-end. Bottom row: A dense point cloud of sequence 09. (Far points are excluded for cleaner visualization.)

TABLE III: Mean Absolute Trajectory Error (ATE RMSE) on 30-frame snippets made from ScanNet [29] sequences. O.M., D.W., and P.U. stand for outlier mask, down-weighting in the pose estimation, and posterior update respectively.

Method	0144_00	0559_01	0565_00	0606_02
Full	**0.017**	**0.021**	**0.018**	**0.012**
O.M.+D.W.	0.025	0.042	0.035	0.018
P.U.+D.W.	0.075	0.054	0.020	0.020
O.M.+P.U.	0.081	0.024	0.019	0.015
None	0.018	0.027	0.021	0.026

ated by our front-end run on the full sequences 09 and 10 in Fig. 1 and Fig. 3. In Fig. 3, there exists a gap, which should be closed, between the beginning and the end of the trajectory due to the drift accumulated along the trajectory. However, overall the point clouds are visually correct.

C. Evaluation on ScanNet

As in the KITTI case, we followed [23] to train our depth, ourlier mask, and sementic segmentation predictions on the ScanNet dataset [29]. Four sequences in the validation set (not used during training) were selected for evaluation. We ran the system on 30-frame snippets and computed the mean ATE RMSE for all the estimated poses. Since the snippet is longer, the trajectory evaluation code from [4] was modified to perform trajectory alignment, using the same definition of ATE RMSE as in the TUM evaluation scripts [34], to obtain the results in Table III.

In Table III, we observe that the full system consistently performs the best whereas the other down-graded variations have mixed performance. The combination of O.M.+D.W. is less effective than how it was in the KITTI evaluation, possibly because there are no dynamic objects in ScanNet, rendering the outlier mask less useful. We further look at the fused and raw semantic segmentation images given by the full system in Table IV. As shown in the table, our simple keyframe propagation step (IV-E) can bring a light

TABLE IV: mIoU of the raw CNN semantic segmentation prediction and the fused semantic labeling through keyframe propagation (IV-E).

	0144_00	0559_01	0565_00	0606_02
Raw (%)	16.77	11.80	15.64	16.74
Fused (%)	**16.82**	**11.93**	**15.69**	**16.80**

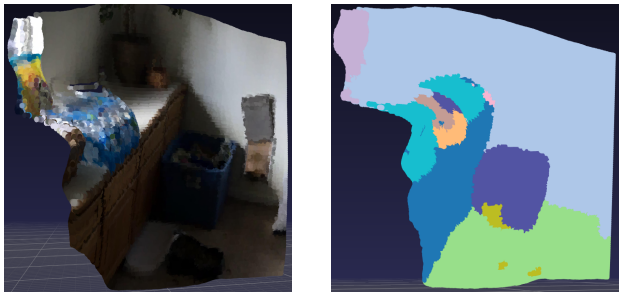

Fig. 4: A dense point cloud and its semantically labeled counterpart of a keyframe generated from the ScanNet [29] data by our front-end.

improvement to the semantic segmentation. A point cloud generated from a processed keyframe is shown in Fig. 4.

VI. Implementation And Performance

The system parameters were tuned separately for the indoor and outdoor datasets. We found it critical to disable the constant speed motion model (IV-B) on ScanNet because it could exaggerate the random and jagged motion of the hand-held camera, occasionally causing the initial point in the pose estimation to leave the converging basin. Other parameters are less critical but adapting them can improve the system efficiency. For example, we can use a shallower image pyramid on ScanNet since the camera motion is small and heavy down-sizing to smooth out the optimization landscape is not needed [18].

All the major computations were implemented with TensorFlow. Inverse depth instead of depth was used in the computations [35]. In our experiments, the system was run on an NVIDIA GTX-1070 GPU at 3 – 5 Hz depending on the image pyramid and how often the keyframes were inserted. We attribute most of the computation effort to the pose estimation step since the iterative computation of the Hessian in Newton's method is expensive. If we can leverage some better implementation of the direct image alignment method that uses only the Jacobian (e.g. as in Gauss-Newton method), the real-time performance can be greatly improved.

VII. Conclusions

We have presented a dense CNN-assisted SLAM front-end that can alleviate the outlier effects due to dynamic objects and static occlusions to achieve better pose estimation accuracy. Unlike the traditional approach of using CNN-based object recognition and segmentation to remove dynamic objects and occlusions (e.g. [36]), our method leverages the outlier mask, a by-product from unsupervised learning of depth, which can be more convenient to deploy in a novel environment where only unlabeled data are available. The utility of other outlier-handling techniques used in unsupervised learning of depth (e.g. [7]) can be further explored in the context of SLAM in the future.

REFERENCES

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[2] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *2016 Fourth international conference on 3D vision (3DV)*. IEEE, 2016, pp. 239–248.

[3] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 270–279.

[4] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1851–1858.

[5] R. Mahjourian, M. Wicke, and A. Angelova, "Unsupervised learning of depth and ego-motion from monocular video using 3D geometric constraints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5667–5675.

[6] J.-W. Bian, Z. Li, N. Wang, H. Zhan, C. Shen, M.-M. Cheng, and I. Reid, "Unsupervised scale-consistent depth and ego-motion learning from monocular video," *arXiv preprint arXiv:1908.10553*, 2019.

[7] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *Proceedings of the IEEE international conference on computer vision*, 2019, pp. 3828–3838.

[8] K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6243–6252.

[9] S. Y. Loo, A. J. Amiri, S. Mashohor, S. H. Tang, and H. Zhang, "CNN-SVO: Improving the mapping in semi-direct visual odometry using single-image depth prediction," *arXiv preprint arXiv:1810.01011*, 2018.

[10] N. Yang, R. Wang, J. Stuckler, and D. Cremers, "Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 817–833.

[11] N. Yang, L. v. Stumberg, R. Wang, and D. Cremers, "D3VO: Deep depth, deep pose and deep uncertainty for monocular visual odometry," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1281–1292.

[12] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison, "CodeSLAM – learning a compact, optimisable representation for dense visual SLAM," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2560–2568.

[13] S. Zhi, M. Bloesch, S. Leutenegger, and A. J. Davison, "SceneCode: Monocular dense semantic reconstruction using learned encoded scene representations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 776–11 785.

[14] J. Czarnowski, T. Laidlow, R. Clark, and A. J. Davison, "DeepFactors: Real-time probabilistic dense monocular SLAM," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 721–728, 2020.

[15] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *International Journal of Robotics Research (IJRR)*, 2013.

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

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

[18] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 1449–1456.

[19] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European conference on computer vision*. Springer, 2014, pp. 834–849.

[20] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

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

[22] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[23] Y. Zhang and J. J. Leonard, "Bootstrapped self-supervised training with monocular video for semantic segmentation and depth estimation," *arXiv preprint arXiv:2103.11031*, 2021.

[24] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. IEEE Computer Society, 2007, pp. 1–10.

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

[26] G. Vogiatzis and C. Hernández, "Video-based, real-time multi-view stereo," *Image and Vision Computing*, vol. 29, no. 7, pp. 434–441, 2011.

[27] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *Advances in neural information processing systems*, 2017, pp. 5574–5584.

[28] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "SemanticFusion: Dense 3D semantic mapping with convolutional neural networks," in *2017 IEEE International Conference on Robotics and automation (ICRA)*. IEEE, 2017, pp. 4628–4635.

[29] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[30] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[31] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Advances in neural information processing systems*, 2014, pp. 2366–2374.

[32] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4040–4048.

[33] M. Grupp, "evo: Python package for the evaluation of odometry and slam." https://github.com/MichaelGrupp/evo, 2017.

[34] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 573–580.

[35] J. Civera, A. J. Davison, and J. M. Montiel, "Inverse depth parametrization for monocular slam," *IEEE transactions on robotics*, vol. 24, no. 5, pp. 932–945, 2008.

[36] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, "DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4076–4083, 2018.