RSTNet: Recurrent Spatial-Temporal Networks for Estimating Depth and Ego-Motion

Tuo Feng[1] and Dongbing Gu[1]

***Abstract*—Depth map and ego-motion estimations from monocular consecutive images are challenging to unsupervised learning Visual Odometry (VO) approaches. This paper proposes a novel VO architecture: Recurrent Spatial-Temporal Network (RSTNet), which can estimate the depth map and ego-motion from monocular consecutive images. The main contributions in this paper include a novel RST-encoder layer and its corresponding RST-decoder layer, which can preserve and recover spatial and temporal features from inputs. Our RSTNet extracts appearance features from input images, and extracts structure and temporal features from intermediate results for ego-motion estimation. Our RSTNet also includes a pre-trained network to detect dynamic objects from the difference between full and rigid optical flows. A novel auto-mask scheme is designed in the loss function to deal with some challenging scenes. Our evaluation results on the KITTI odometry benchmark show our RSTNet outperforms some of the existing unsupervised learning approaches.**

***Index Terms*—Localisation, Visual Odometry, SLAM, Depth and Ego-Motion Estimation, Deep Learning SLAM.**

I. Introduction

AUTONOMOUS navigation of mobile robots or self-driving cars heavily depends on vision perception to understand their operation environments. Visual Odometry (VO) emerges as a technique to provide the key information for autonomous navigation from images or videos. Accurate depth and ego-motion estimations become the primary target of research works in the research area.

In recent years, deep learning methods have been proven to be more effective in processing robust and efficient estimation tasks. Researchers strive to create an end-to-end deep learning VO system for depth and ego-motion estimation [1]–[3].Unsupervised deep learning methods for depth or ego motion estimation do not need to label the data sets to train the learning systems. They use geometric constraints and unlabeled data sets to train and become one of the popular research approaches [4]–[7]. Unsupervised deep learning VO methods are able to provide even better results than classical geometric algorithms in some challenging scenes [8]. Monocular VO methods also demonstrate promising results under the framework of unsupervised learning [9]–[13].

In this paper, we make a few innovations in estimating depth, and ego-motion from monocular images. Firstly we propose novel RST-encoder and RST-decoder layers to build an unsupervised end-to-end deep learning VO system to estimate the depth and ego-motion from monocular videos. This is inspired by the 3D packing block [11] but our RST layers include additional channels to capture and retain temporal features. The results demonstrate our depth estimation outperforms some of the state of the art systems. This is due to the reason that the RST layers are able to preserve and retain high resolution features. Secondly our proposed system takes original images, estimated depth maps, and estimated optical flow as input for ego-motion estimation. Most existing systems only use the combination of original images and estimated depth maps [1], [3], [12], [14] as input for ego-motion estimation. Some utilises the optical flow estimation as mask in the cost function [9], [15], [16]. Some only use their estimated depth image and estimated optical flow for ego-motion estimation [17]. Our proposed system is able to provide appearance features from images, structure features from depth map, and temporal features from optical flow for ego-motion estimation. Thirdly we design an auto-mask scheme which adds a mask component constructed from the depth error on the top of the minimum reprojection loss [10]. We found they can better remove the objects moving at similar speeds to the camera and the scenes where the camera is temporarily static in our evaluation. We also use a semantic segmentation network to detect dynamic objects. Our innovation is the use of the difference between full optical flow and rigid flow as inputs to the segmentation network.

Our novel recurrent spatial-temporal network (RSTNet) is shown in Fig. 1. The details of this architecture is elaborated in Section III. The outline of this paper is organized as follows: We introduce the related work in Section II. Section III provides a detailed illustration to the RST-encoder and RST-decoder layers. Section IV gives an overview of our proposed RSTNet architecture and its dynamic auto-masking loss function. Section V presents our experimental results on the KITTI odometry dataset with comparisons of some previous works. Finally, the conclusion and future work are drawn in Section VI.

II. Related Work

A. Unsupervised depth and ego-motion estimation networks

Unsupervised stereo depth estimation networks were proposed in [4], [5] where the left-right consistency was used as the supervision signal. Exploring temporal features between two consecutive images leads to the ego-motion estimation under unsupervised learning frameworks, as proposed in [1]. UnDeepVO [3] extended to learn depth and ego-motion estimations by using stereo image pairs with additional loss functions. SfMLearner++ [2] enforced a further epipolar constraint on the loss. Zhan et al. [18] explored the use of

Manuscript received 05/2022, revised 08/2022, accepted 12/2023

[1]Tuo Feng and Dongbing Gu are with the School of Computer Science and Electronic Engineering, University of Essex, Colchester, CO4 3SQ, UK. {tfeng, dgu}@essex.ac.uk

dense features as an additional supervision signal in monocular depth and ego-motion estimations. GANVO [14] formulated the monocular depth and ego-motion learning framework into a GAN learning mechanism. SGANVO [19] put forward a stacked generative adversarial network to capture high-level spatial and temporal features.

B. Ego-motion estimation with optical flow and geometrical methods

The rigid optical flow of a static scene caused by camera motion is able to provide temporal feature correspondences between two consecutive images. DF-Net [13] can jointly learn depth, ego-motion, and optical flow estimations. Ranjan et al. [20] proposed four networks to predict monocular depth, ego-motion, optical flow and dynamic object masks. UnOS [15] also used separate networks for depth, ego-motion, and optical flow estimations. In [21], the flow estimation was used to enforce robust and direct constraints for end-to-end learning of camera pose.

In some other works, the depth was estimated from network, but the ego-motion was estimated directly from geometrical methods. In [22], an encoder-decoder network for depth estimation and a differentiable Direct Visual Odometry (DDVO) module for ego-motion estimation were proposed. A 3D geometric constraint was used in [6]. In [7] the ego-motion was estimated by using a geometrical method but a network was used to predict the pose correction.

C. Dynamic object masks

The assumption of unsupervised learning frameworks is a moving camera and a static scene. It is necessary to mask out the occlusion and non-rigidity to improve the estimation performance. The early work in [1] proposed to use a network to predict the uncertain mask. SC-SfMlearner [23] proposed a geometry consistency loss and a self-discovered mask with the depth inconsistency map for handling dynamic objects and occlusions. Godard et al. [10] proposed an auto-masking loss function to only select the pixels that remain the same between adjacent frames. Jiang [12] extended the minimum reprojection loss by adding the statistical information of photometric errors. Optical flow was explicitly exploited to mask out dynamic objects and occlusion areas in [15]. Yin et al. [9] proposed GeoNet to train a residual optical flow network for non-rigid structure. In [16] occlusions were masked out by using optical flow and depth map. Zhao et al. [17] used an occlusion map computed from optical flow to remove the occlusions. Wang et al. [24] explicitly segmented occlusion areas and used the occlusion masks in unsupervised training.

Using semantic segmentation networks can provide more accurate object information as introduced in [25]. Casser et al. [26] used instance segmentation masks to estimates motion objects. Klingner et al. [27] estimated depth and ego-motion with a semantic segmentation network. Using predicted depth maps together with original images as input to estimate ego-motion was also observed in [28].

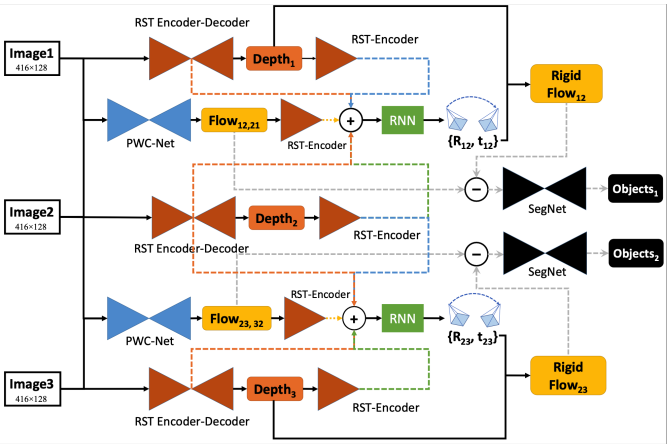


Fig. 1: Our proposed RSTNet architecture for self-supervised learning. The RST-encoder and RST-decoder components consist of multiple RST-encoder and RST-encoder layers. The depth is estimated from the network consisting of a RST-encoder component and a RST-decoder component. The ego-motion is estimated from the RNN network with inputs from appearance features in input images, structure features from depth maps, and dynamic features from optical flows. A pre-trained PWC-Net [33] is used to estimate the full optical flow. The dynamic objects are detected by a pre-trained SegNet [34] with input from the difference between full and rigid flows.

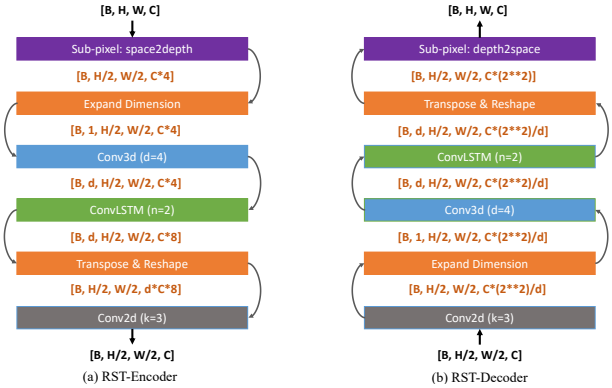


Fig. 2: Recurrent Spatial-Temporal layers: (a) a single RST-encoder layer and (b) a single RST-decoder layer.

D. High resolution features

It is well known that input images could lose high resolution features after pooling and striding operations in CNNs. Inspired by the sub-pixel network [29], Superdepth [30] and Zhou et al. [31] used a sub-pixel network to improve the performance for monocular depth estimation. Further digging into the network architecture, Packnet-SfM [11] proposed 3D packing and unpacking blocks to replace the max-pooling layer and bilinear upsample layer of traditional depth networks. Its detail-preserving property of the architecture could reconstruct the near-lossless features of images. In [32], a Recurrent Modulation Unit (RMU) was used for feature fusion in the depth network.

III. Recurrent Spatial-Temporal Layers

Most convolutional layers in CNNs pursue enhancing its receptive field size through aggressive striding and pooling operations. But these operations potentially could lead to the loss of detailed pixel representations and the resize-convolution operation in upsampling layers could not recover the details. We propose a novel recurrent spatial-temporal encoder layer (RST-encoder layer) which can preserve detailed spatial and temporal features from its inputs. We also proposes an RST-decoder layer, which is symmetrical to the RST-encoder layer. Both of them are the building blocks in our RSTNet.

A. RST-encoder layer

In Fig. 2a, the RST-encoder layer starts with a space2depth operation, which is taken from the sub-pixel network in [29]. This layer can fold the height (H) and width (W) dimensions of a feature map into extra feature channels ($C*4$). This operation is a reversible transformation without any loss, which is different from striding or pooling operations. This is followed by a dimension expanding operation which adds an extra dimension to the tensor for a 3D convolution layer (Conv3d) with $d = 4$. This extra dimension is used to retain the detailed spatial features. The 3D convolution layer is used to compress the concatenated features and outputs the same size tensor. Then a convolutional LSTM layer (ConvLSTM) consisting of two ConvLSTM cells ($n = 2$) is used to capture the temporal features. The ConvLSTM layer outputs the same size tensor with feature channels ($C*8$). After that, we transpose and reshape the extra dimension (d) of the tensor to the channel dimension ($d*C*8$), which includes the detailed spatial-temporal features. In the end, a 2D convolution layer (Conv2d) is used to produce the tensor with the desired number of feature channels. This structure of cascading multiple complex convolutional layers allows the RST-encoder layer to preserve the detailed spatial and temporal features extracted from input tensor.

B. RST-decoder layer

The RST-decoder layer is a symmetrical architecture with cascading multiple deconvolution layers as shown in Fig. 2b. The compressed concatenated features from the RST-encoder layer are the input tensor to the first layer (Conv2d). This layer is to adjust the number of channel dimensions ($C*4/d$) to adapt to the following 3D convolutional layer (Conv3d). Then this 3D convolution layer expands back the compressed spatial features to a *ConvLSTM* layer. The *ConvLSTM* layer also consists of two *ConvLSTM* cells to preserve the temporal features. The channel dimension in its output is $c*4/d$. Next, we transpose and reshape the tensor to $d*c*4/d = c*4$ channels. This layer ends with a depth2space operation that reduces the channel dimensions to the original C.

C. RST layers for Depth Estimation

Our RST-encoder and decoder layers can be used to estimate the depth from monocular consecutive images. To show the capability of our RST layers in recovering lossless spatial-temporal features, we use the RST encoder and decoder layers to replace the pooling, striding, and upsampling layers of the depth estimation network in [1] without changing any other data processing, loss function and network parameters. The input images are shown in the first column in Fig. 3. The estimated depth maps from the network in [1] replaced with our RST layers are shown in the second column (RST-SfMlearner). The estimated depth maps from [1] are shown in the third column (SfMlearner). These results show our RST layers can learn more dynamic structures and detailed spatial features than other networks for specific tasks. In Fig. 3, the depth estimation with our RST layers has more spatial details than the original network, such as road sign, vehicles, and irregular branches. In addition, the results also show our RST layers perform much better on dynamic objects, such as persons by motorcycle or by bike.

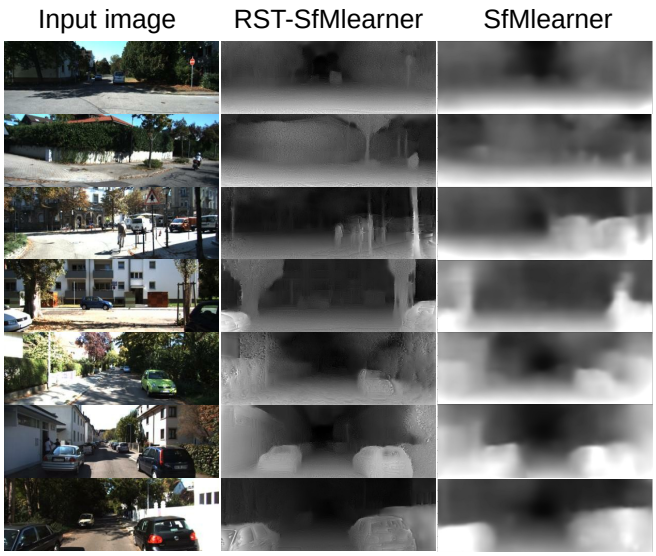


Fig. 3: Depth estimation from our RST layers compared with SfMlearner [1]. The images on the first row show a small and distant view scene. The images on the second, third, and fourth rows show dynamic objects and irregular objects. The images on the fifth, sixth, and seventh rows show high resolution scenes such as dense foliage and car parts.

IV. Recurrent Spatial-Temporal Network: RSTNet

A. RSTNet Overview

Our proposed RSTNet targets four main estimation tasks: depth, ego-motion, optical flow, and dynamic objects. As shown in Fig. 1, three monocular consecutive images ($Image_{1,2,3}$) are used as its inputs. The RSTNet estimates three corresponding depth maps using three RST encoder and decoder networks. It estimates two full optical flow maps ($Flow_{12}$,$Flow_{23}$) using two pre-trained PWC-Nets [33]. It estimates two ego-motions ($R_{12}, t_{12}, R_{23}, t_{23}$) from two RNNs, each of which uses the features extracted from two monocular

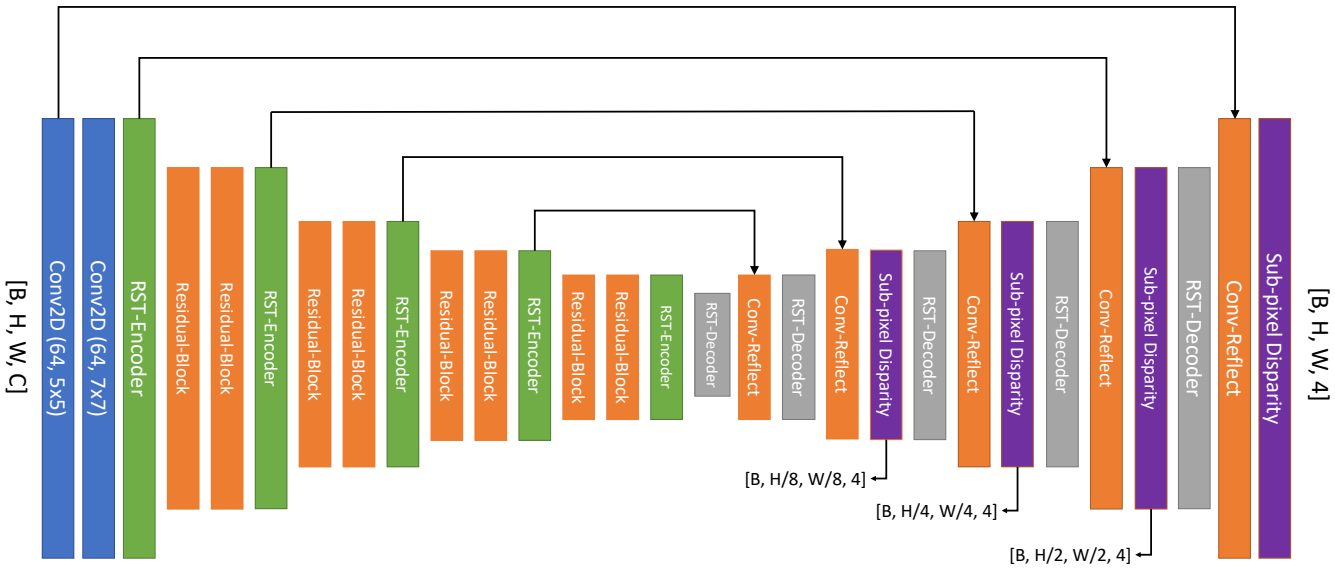


Fig. 4: The depth estimation network is constructed with an encoder-decoder architecture by using multiple RST-encoder and RST-decoder layers.

consecutive images, two estimated depth maps, and one estimated optical flow as the inputs. Two dynamic object maps $(Object_1, Object_2)$ are detected from two pre-trained networks (SegNet) [34], each of which uses the difference between a full flow from PWC-Net and a rigid flow computed from the depth and ego-motion estimations. The RST encoder and RST decoder layers are used as the building blocks in constructing our RSTNet.

B. Depth Estimation Network

Our depth network (Fig. 4) is constructed with the symmetrical encoder and decoder network structure, including multiple RST-encoder and RST-decoder layers. The monocular image input is a tensor with $[B,H,W,C]$ shape where B is the batch size dimension, H is the height dimension, W is the width dimension, and C is the channel dimension. The first two 2D convolution layers extract the features as 64 output channels. The first RST-encoder layer folds the feature tensor and expands the spatial-temporal features for the next level. Starting from the second level of encoding, each of the following encoding layers begins with two residual blocks and ends with a RST-encoder layer, shrinking with a scale of 2 times. The residual block is composed of two stacked 2D Convolution layers, Batch Normalization, and RELU Activation Function. After five RST encoder layers, the height and width of the input tensor are compressed by 16 times.

The decoder part begins with a RST-decoder layer to extend the size of its input tensor to 4 times. Afterwards, the Convolution Reflect block combines its output and the skip features to decode the tensor. Then each of the following four decoder layers consists of a RST-decoder layer, a Convolution Reflect block and a sub-pixel disparity layer. Each of them outputs a tensor 2 times larger in height and width of its input so that the decoder part can obtain the depth maps with four scales. The Convolution Reflect block is a 2D Convolution layer with the half size of filter pad. The sub-pixel disparity layer is the same architecture as the Convolution Reflect block with four output channels. The decoder part of this network can produce the disparity maps with 4 scales $[H/8,W/8]$, $[H/4,W/4]$, $[H/2,W/2]$, and $[H,W]$. Its output resolution on each scale is four times larger than most published depth estimation networks so it can estimate more details.

C. Ego-Motion Estimation Network

Depth map carries the structural information of scenes, optical flow represents the dynamic information, and original image provides a large number of appearance features in the view. To estimate the ego-motion with good accuracy, we use two adjacent images, their depth maps, and the corresponding optical flow as the source of feature extraction.

In the ego-motion network of RSTNet (see Fig. 1), we reuse the appearance features of two consecutive images extracted in the encoder part of the depth estimation network as inputs. We also use the structural features extracted from two consecutive depth maps as inputs. Furthermore, we use the dynamic features extracted from the estimated optical flow as inputs. The optical flow is estimated from the pre-trained PWC-Net. All these features are concatenated on the channel dimension of inputs and fed into a RNN network to estimate the ego-motion. The RNN network consists of two convolution LSTM cells with 256 output channels for decoding the rotation R and the translation t.

Fig. 5: The first row includes the input image (left) and our estimated depth (right). Our auto-mask (μ_d) (left) are compared with the self-discovered mask [23] (right) in the second row. The projected image errors from our RSTNet and [10] are shown in the third row. Our auto-mask $\mu_t + \mu_s$ is shown in the bottom two rows. The vehicle with a similar velocity as the camera is shown in the red circle (left) and eliminated as shown in the black (right) in the fourth row. The static distant sky (left) is shown in the fifth row and eliminated as shown in the black (right).

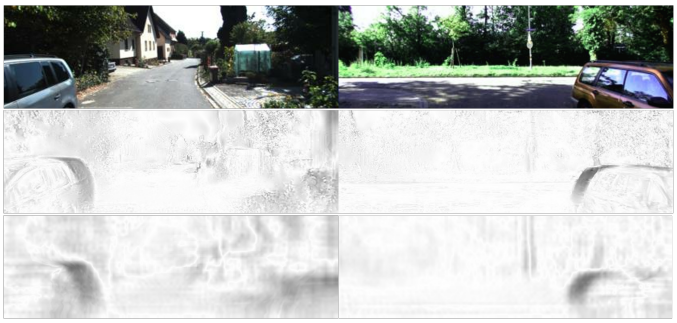


Fig. 6: The first row includes two input images. Our auto-masks (μ_d) are shown in the second row and the self-discovered mask [23] are shown in the third row.

D. Loss Functions and Auto-Masking Scheme

The synthetic target view $I^i_{s\to t}$ is generated from the ith source view I^i_s:

$$I^i_{s\to t}(x,y) = K\hat{T}_{s\to t}d_t^{-1}(x,y)K^{-1}I^i_s(x,y) \tag{1}$$

where K is the camera intrinsic matrix, $d_t(x,y)$ is the estimated disparity, $\hat{T}_{s\to t}$ is the camera coordinate transformation matrix from the ith source frame to the target frame and estimated by the ego-motion network. Our photometric reprojection loss function L_p is defined as below:

$$L_p = \sum_{i=1}^{N} \min_t pe\left(I_t, I^i_{s\to t}\right) + \sum_{i=1}^{N} \min_s pe\left(I^i_s, I^i_{t\to s}\right) \tag{2}$$

where N is the number of source views and $N = 2$ for three consecutive input frames. The pe is a photometric reconstruction error computed from $SSIM$ [35] and L_1 [26] [16]:

$$pe(I_a, I_b) = \frac{\alpha \times (1 - SSIM(I_a, I_b))}{2} + (1-\alpha) \times \|I_a - I_b\|_1 \tag{3}$$

We also use a depth edge-aware smooth loss function to regularize the depth in texture-less and low-image gradient regions.

$$L_s = |\delta_x d^*_t| e^{-|\delta_x I_t|} + |\delta_y d^*_t| e^{-|\delta_y I_t|} \tag{4}$$

where $d^*_t = d_t/\bar{d}_t$ is the mean-normalized disparity, δ_x and δ_x are the gradient.

Finally, our total loss function includes the photometric reprojection loss with a dynamic auto-masking weight μ and the edge-aware depth smooth loss:

$$L_{total} = \mu L_p + \lambda L_s \tag{5}$$

This loss function performs very well for the training under the assumption of static scenes with a moving camera. However, its performance could deteriorate rapidly when the camera is stationary or there are dynamic objects in the scene [10]. The auto-masking loss function in [10] alleviates some of these problematic scenes. It can mask the target image to reduce the influence of objects when moving at the same velocity as the camera or ignore the whole image when the camera stops moving. But it does not exploit dynamic information provided by moving objects.

We propose a novel dynamic auto-masking scheme. The full mask includes three parts $\mu = \mu_t + \mu_s + \mu_d$. The first part μ_t is the same as the one purposed in [10].

$$\mu_t = \left[\sum_{i=1}^{N} (\min_t pe\left(I_t, I^i_{s\to t}\right) < \min_t pe\left(I_t, I^i_s\right))\right] \tag{6}$$

This mask can remove the pixels representing other objects moving at a similar velocity as the camera or all pixels when the camera is static.

The second part μ_s plays a similar role as μ_t but with reverse projection $I^i_{t\to s}$:

$$\mu_s = \left[\sum_{i=1}^{N} (\min_s pe\left(I^i_s, I^i_{t\to s}\right) < \min_s pe\left(I^i_s, I_t\right))\right] \tag{7}$$

The third part μ_d is defined as a depth error between the estimated depth map d_t and its warped map $d^i_{s\to t}$ from the ith source depth based on optical flow:

$$\mu_d = \sum_{i=1}^{N} \left(1 - \frac{\|d_t - d^i_{s\to t}\|}{d_t + d^i_{s\to t}}\right) \tag{8}$$

Different from the auto-masks [10] and the self-discovered masks [23], we use a full optical flow to warp a depth map to generate the full dynamic auto-masks. A full optical flow could record all the dynamic information within a scene. In Fig. 5, there is a clear dynamic object (motorcycling) in the input image (left on the first row). Our depth estimation clearly shows the scene (right on the first row). Our auto-mask (μ_d) and the self-discovered mask [23] are shown on the left and right of the second row, respectively. It is clear that our mask can provide a much more distinct figure and also a thinner

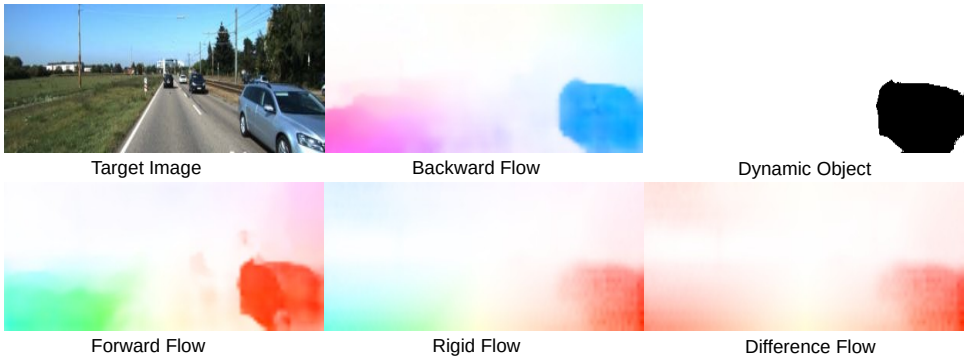


Fig. 7: Dynamic object detection in our RSTNet. The forward flow $Flow_{12}$ and backward flow $Flow_{21}$ are predicted by the PWC-Net [33]. The rigid flow is computed through the estimated depth and ego-motion. The flow difference is the difference between full and rigid flows. The pre-trained SegNet uses the flow difference to estimate the dynamic objects.

outline of the trees than the self-discovered mask. Given a more precise object detection, the mask used in the loss function could avoid the information loss. The projected image errors from our RSTNet and [10] are shown in the left and right of the third row, respectively. It is clearly demonstrated that our mask detects dynamic objects much better. In the last two rows, the performance of our auto-mask ($\mu_t + \mu_s$) is shown. The vehicle with a similar velocity as the camera is shown in the red circle (left) and eliminated as shown in the black (right) in the fourth row. The static distant sky (left) is shown in the fifth row and eliminated as shown in the black (right).

Two additional images for our auto-mask (μ_d) and the self-discovered mask [23] are shown in Fig. 6. The moving parts between two consecutive images are detected in mask images. We can find that much clearer outlines of the cars can be observed in our masks (the second row) than self-discovered mask [23](the third row).

E. Dynamic Object Estimation

We use a pre-trained network (SegNet) to detect the dynamic objects in our RSTNet. The SegNet is an encoder-decoder architecture and used here to classify each pixel as two classes: static and dynamic. The input is the difference between the full flow estimated from the PWC-Net and the rigid flow reconstructed by using the estimated depth and ego-motion (see Fig. 1). This is a different use against the work in [9] [36] where they estimate the residual flow using a network, then reconstruct the full flow with a rigid flow.

Some results from this network are shown in Fig. 7. We use an image with a size of 384×184. The forward flow ($Flow_{12}$) and backward flow ($Flow_{21}$) are predicted by the PWC-Net. The rigid flow is computed by using the estimated depth and ego-motion. The flow difference is the full flow minus the rigid flow. The dynamic object is separated from the static background by the SegNet. The result shows the profile of a moving car is clearly detected. There are three cars far away from the camera, which are not detected in Fig. 7. The main reason is due to the fact that their displacements projected in 2D image from 3D motion are too small for detection.

V. Experiments

We implemented the proposed RSTNet by using the Tensorflow framework and trained it with one NVIDIA GTX 1080TI GPU. The Adam optimizer was employed to speed up the network convergence for up to 30 epochs with parameter $\beta_1 = 0.9$. The learning rate started from 0.0001 and decreased to 0.00001 after $3/4$ of total iterations. Because of our GPU's memory size limitation, the size of input images was 416×128 under the consideration of comparison with other networks. For data prepossessing, we used different kinds of data augmentation methods introduced in the previous work [1] [10] that can enhance the performance and mitigate possible over-fitting.

A. Depth Estimation Evaluation

For depth estimation, we chose the KITTI dataset [37] with a benchmark split from [38] as the benchmark. The camera parameters are required for constructing the loss function, and reconstructing 3D scenes. As shown in Fig. 8, we compared our method with the state-of-the-art unsupervised depth estimation methods, such as Monodepth2 [10], SC-SfMLearner [23], and Packnet-SfM [11]. The dynamic challenging scenes including pedestrians are shown in the first and second rows. Our RSTNet performs best compared with the other three methods such as the shape details for pedestrians. This can also be seen in the third to the sixth rows where the motorcycle and vehicles are dynamic objects. In addition, the RSTNet depth estimation of static objects, such as the trees, street lights, and signposts, has a higher resolution and a clearer view of structures. These results are from a smaller size image (416×128) than others (Packnet-SfM (640×192), SC-SfMLearner (832×256), and Monodepth2 (640×192)).

We list the quantitative depth estimation results in Table I, where M stands for using monocular image sequence for training and S stands for using segmentation inputs for training. Compared with the existing state-of-the-art unsupervised learning-based methods, our RSTNet has a better performance in terms of the metrics used (see the smallest values in Abs Rel, Sq Rel, RMSE, RMSE log columns and the highest values δ of the last three columns in the table). Our RSTNet uses the smallest image size but achieves better depth estimation results without additional inputs like work in [26] and [16].

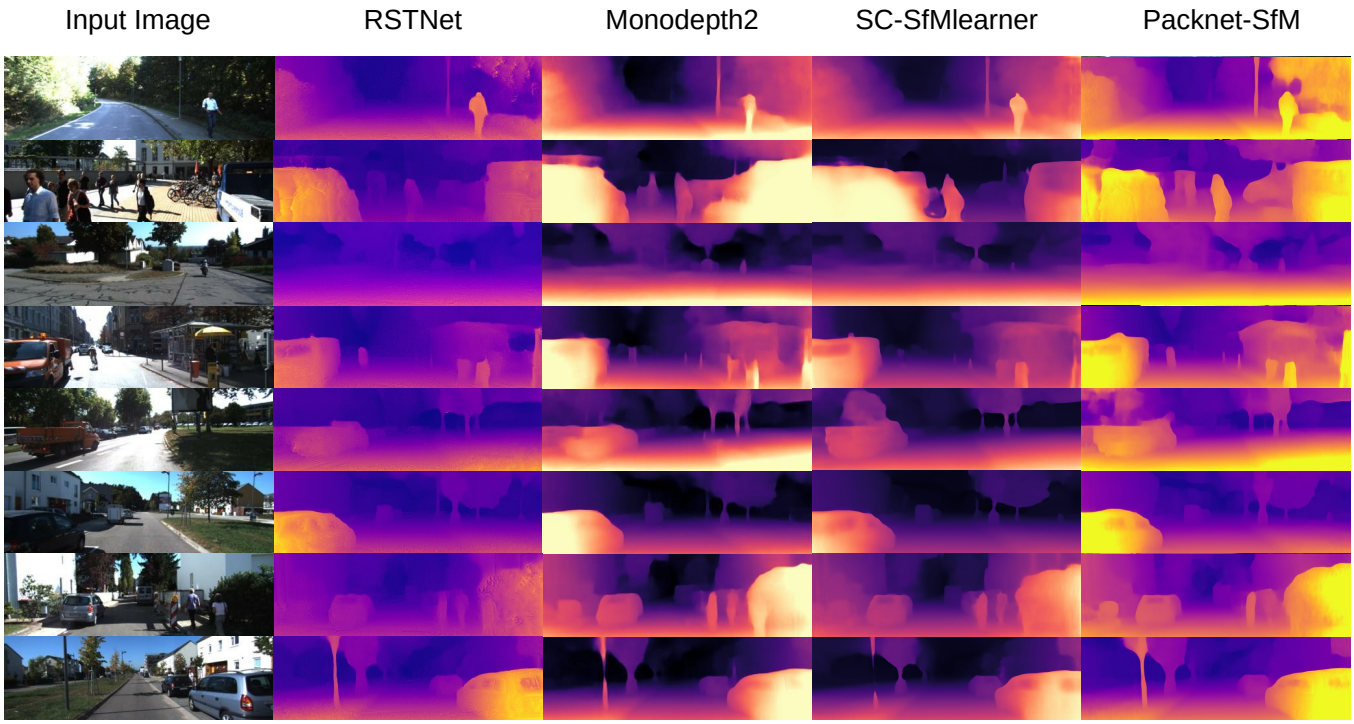


Fig. 8: Illustrated above are qualitative comparisons of our RSTNet depth estimation (image size: 416×128) with Monodepth2 [10] (image size: 640×192), SC-SfMLearner [23] (image size: 832×256), and Packnet-SfM [11] (image size: 640×192). The depth map comparisons show that our approach produces qualitatively better estimations with crisper boundaries and finer details.

The above results show own proposed RST-encoder and RST-decoder layers play an important role in the depth estimation. These layers replaced the normal pooling, striding, and upsampling layers of encoder-decoder architectures. They are able to preserve the detailed features in spatial and temporal domains from input image sequence. Consequently, our depth network can estimate more accurate results than some other methods which are based on normal pooling, striding, and upsampling layers.

B. Ego-Motion Estimation Evaluation

We used the KITTI dataset [37] to test the performance for ego-motion estimation. The KITTI dataset only provides the ground truth of 6-DoF poses for Sequence 00-10. We used Sequence 00-08 for training and Sequence 09-10 for testing. The metrics are the average translational root-mean-square error (RMSE) drift and average rotational RMSE drift $(^\circ/100m)$ on length of $100m - 800m$. The results are shown in Table II. We compared the results with SGANVO [19], Zhao et al. [17], ORB-SLAM2 (without loop closure) [40], UndeepVO [3], and UnOS [15]. SGANVO is based on monocular image training. UndeepVO and Undepthflow are based on stereo image training. ORB-SLAM2 is a geometry method. Zhao et al. [17] used the geometry-based methods and deep learning to estimate the ego-motion. It can be seen that our RSTNet shows a better performance in the testing sequences (09, 10) with these state-of-the-art methods in terms of the translational and rotational RMSE drift metrics. It outperforms other end-to-end learning-based methods (SGANVO, UndeepVO, and Undepthflow). But the average rotational drifts of learning based methods (RSTNet, SGANVO, UndeepVO, and Undepthflow) are rifely larger than the geometric methods such as ORB-SLAM2 and Zhao et al. [17]. This shows that the deep learning method has a limited learning capability on rotation estimation compared with the geometry methods.

Another metric used is the absolute trajectory error (ATE) averaged over all overlapping 5-frame snippets. We concatenated all the estimations together for the entire sequences without any post-processing. The estimated trajectories of sequences 09 and 10 from our RSTNet and its ground truth are shown in Fig. 9. Although the estimated results include some drifts, our RSTNet can estimate all the features of the trajectory and performs well when no loop closure detection was used. The quantitative results are shown in Table III where we compare our results with ORB-SLAM2(Full) [40], SfM-Learner [1], GeoNet [9], and Monodepth2 [10]. In comparison, our RSTNet produced a good result. This is mainly due to the use of better depth estimation result which is important in computing the cost function in Eq. (1), the use of optical flow estimation which is directly related to the pose estimation, and the use of mask estimation which is able to provide more dynamic information in the scenes.

C. Dynamic Object Estimation Evaluation

The pre-trained SegNet was used for detecting dynamic objects. The performance of using this network for dynamic

TABLE I: Depth estimation results on the KITTI dataset using the split of Eigen et al. [38]. For training data, $K = KITTI$, $CS = Cityscapes$ [39], $M = Monocular$ and $S = Segmentation\ Inputs$. For testing data, our RSTNet uses monocular images.

Method	Data size	Dataset	Train	Abs Rel	Sq Rel	RMSE	RMSE log	δ <1.25	δ <1.25^2	δ <1.25^3
Garg et al. [4]	620 × 188	K	M	0.169	1.080	5.104	0.273	0.740	0.904	0.962
SfMLearner [1]	416 × 128	K	M	0.208	1.768	6.856	0.283	0.678	0.885	0.957
SfMLearner [1]	416 × 128	CS+K	M	0.198	1.836	6.565	0.275	0.718	0.901	0.960
GeoNet [9]	416 × 128	K	M	0.155	1.296	5.857	0.233	0.793	0.931	0.973
GeoNet [9]	416 × 128	CS+K	M	0.153	1.328	5.737	0.232	0.802	0.934	0.972
Vid2Depth [25]	416 × 128	K	M	0.163	1.240	6.220	0.250	0.762	0.916	0.968
Vid2Depth [25]	416 × 128	CS+K	M	0.159	1.231	5.912	0.243	0.784	0.923	0.970
GANVO [14]	416 × 128	K	M	0.150	1.414	5.448	0.216	0.808	0.939	0.975
SC-SfMLearner [23]	416 × 128	K	M	0.137	1.089	5.439	0.217	0.830	0.942	0.975
Monodepth2 [10]	640 × 192	K	M	0.115	0.882	4.701	0.190	0.879	0.961	0.982
Packnet-SfM [11]	640 × 192	K	M	0.111	0.785	4.601	0.189	0.878	0.960	0.982
Struct2Depth [26]	416 × 128	K	S + M	0.183	1.730	6.570	0.268	-	-	-
Gordon et al. [16]	640 × 192	K	S + M	0.129	1.112	5.180	0.205	0.851	0.952	0.978
RSTNet	416 × 128	K	M	**0.108**	**0.767**	**4.524**	**0.188**	**0.891**	**0.974**	**0.991**

TABLE II: Ego-motion estimation results on the KITTI dataset with our proposed RSTNet. We also compare with four learning methods (Monocular: SGANVO [19], Zhao et al. [17]; Stereo: UndeepVO [3], Undepthflow [15]), and one geometric based methods (ORB-SLAM2 [40]). RSTNet, SGANVO, and UndeepVO use images with 416×128. Zhao et al. and Undepthflow use 832×256 images. ORB-SLAM2 uses 1242×376 images. The best results among the learning methods are made in bold.

Seq.	Monocular								Stereo			
	RSTNet (416×128)		SGANVO [19] (416×128)		Zhao et al. [17] (832×256)		ORB-SLAM2 [40] (1242×376)		UndeepVO [3] (416×128)		Undepthflow [15] (832×256)	
	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$
03	**4.67**	**3.08**	10.56	6.30	-	-	0.67	0.18	5.00	6.17	-	-
04	**2.28**	**0.69**	2.40	0.77	-	-	0.65	0.18	4.49	2.13	-	-
05	**2.97**	**1.04**	3.25	1.31	-	-	3.28	0.46	3.40	1.50	-	-
06	**3.81**	**1.13**	3.99	1.46	-	-	6.14	0.17	6.20	1.98	-	-
07	**2.83**	**1.77**	4.67	1.83	-	-	1.23	0.22	3.15	2.48	-	-
09*	**4.75**	**2.22**	4.95	2.37	6.93	**0.44**	15.30	0.26	7.01	3.61	13.98	5.36
10*	**5.54**	**2.72**	5.89	3.56	**4.66**	**0.62**	3.68	0.48	10.63	4.65	19.67	9.13

- t_{rel}: average translational RMSE drift (%) on length of 100m-800m.
- r_{rel}: average rotational RMSE drift ($^\circ/100m$) on length of 100m-800m.
- train sequence: 03,04,05,06,07; test sequence: 09*, 10*

TABLE III: Absolute Trajectory Error (ATE) on the KITTI odometry dataset. The results of other baselines are taken from [9].

Method	frames	Sequence 09	Sequence 10
ORB-SLAM2(Full) [40]	All	0.014 ± 0.008	0.012 ± 0.011
SfMLearner [1]	5	0.016 ± 0.009	0.013 ± 0.009
GeoNet [9]	5	0.012 ± 0.007	0.012 ± 0.009
Monodepth2 [10]	2	0.017 ± 0.008	0.015 ± 0.010
RSTNet	3	0.011 ± 0.006	0.010 ± 0.005

object detection is shown in Fig. 10. There are dynamic objects such as vehicles and pedestrians in each input image. Visually, our estimation has good accuracy and the outline and shape of dynamic objects (car in the first to third rows and pedestrians in the fourth row) are clear. It can be seen that the shadow of a moving car in the second, third, and fourth rows is also detected as the part of dynamic objects, which is a reasonable result given the shadow was also moving with the car. However, the detection network is still far from perfection. Some static patches are misidentified as dynamic objects in the second and third rows. One of the reasons for this could be we just used the pre-trained SegNet without further training on the dataset.

D. Network Complexity

Our RSTNet demonstrated a significant performance in terms of various estimation tasks, such as depth, ego-motion

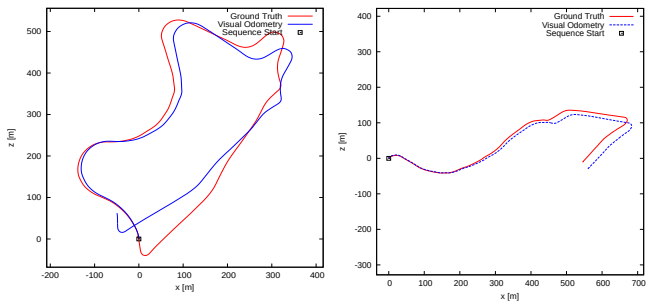


Fig. 9: Our proposed RSTNet estimates the trajectory on the KITTI odometry benchmark, Sequence 02 (upper left), Sequence 08 (upper right), Sequence 09 (lower left) and Sequence 10 (lower right). Our results are coloured in blue while the ground truth is coloured in red.

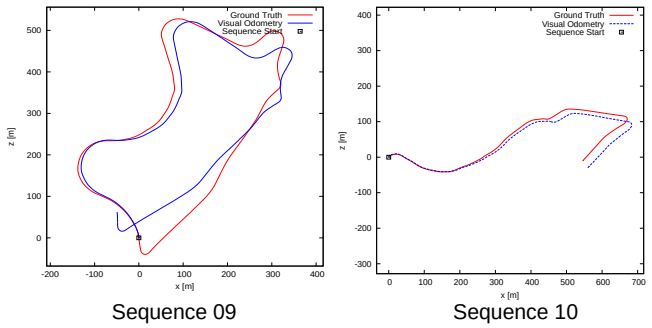


Fig. 10: Our proposed RSTNet to detect dynamic objects on the KITTI odometry benchmark.

and dynamic objects. But it comes at a price of network computational complexity. For a fair comparison, all of our experiments were conducted under Tensorflow 1.14 version with the same size (416×128) for input image. We implemented our RST-encoder and RST-decoder layers based on the RES18 network. We replaced the striding, pooling and upsampling operations in each layer of the RES18 network with our RST layers. To demosntrate the network complexity, we constructed five versions of networks (RST-V1 to RST-V5) by replacing each of five filters in the RES18 network with RST layers, and a full version (RST-FULL) by replacing all five filters together. We implemented the 3D-Packenet [11] in the same Tensorflow environment. Table IV presents the network parameters and inference time on the NVIDIA GTX 1080TI GPU. The inference time is the average over the last 10 iterations in training epoch 2.

Table IV reveals the number of parameters for version V2 to V4 is larger than the RES18 network, and accordingly longer inference time is required. The full version in the last row shows both the number of parameters and inference time became very large but produced good estimation results as demonstrated in Fig. 3.

E. Ablation Analysis

We conducted three experiments for an ablation analysis with three network versions (RST-V1, RST-V12 to RST-V123). The version RST-V1 is a result of replacing the first filter in the RES18 network with an RST layer. The version RST-V12 is a result of replacing the first two filters in the RES18 network with two RST layers. The version RST-V123 is a result of replacing the first three filters in the RES18 network with three RST layers. In Table V, we set the same network and training parameters to explore the influence of increasing the number of RST layers on the estimation performance. The main evaluation is focused on the ego-motion and depth estimation for sequence 09 and sequence 10.

TABLE IV: The number of parameters and inference time for different versions of networks.

Network	Number of parameters	Time(s/it)
3D-Packnet [11]	23170736	1.24357
RST-V1	23125232	1.23886
RST-V2	37062512	1.42846
RST-V3	26444144	1.33599
RST-V4	23789168	1.30512
RST-V5	23125232	1.23902
RST-FULL	40863344	1.89274

It can be seen from Table V that both ego-motion and depth estimations perform better and better when the number of RST layers increases from 1 to 3. More RST layers could lead to more accurate results, but with a longer computational time as indicated before.

We also conducted seven experiments for an ablation analysis on the features used for ego-motion estimation in our RSTNet. In Fig.1, the RNN ego-motion network uses three kinds of input features extracted from two consecutive input images, two estimated depth maps, and one optical flow map. The input image features (F_i) provide the appearance information and they are the most effective features for ego-motion estimation as demonstrated in most research works. The depth features (F_d) provide the structural information of scenes. The flow features (F_f) provide the dynamic information of objects.

The average translational RMSE drift t_{rel} and rotational RMSE drift r_{rel} for Sequence 09 are used for evaluation in Table VI. It can be seen the results are not improved when the depth features (F_d) or flow features (F_f) are used individually when comparing with image features (F_i). But the results can be improved when the depth features or flow features are used together with image features (see F_{i+d} and F_{i+f} columns).

When the combination of depth features and flow features are used (F_{d+f} column), the results get worse, which implies that the image features are the most effective factor. When all three kinds of features are used (F_{i+d+f} column), the results are the best.

VI. Conclusion

In this paper, we propose a novel monocular recurrent spatial-temporal network (RSTNet) for estimating depth, and ego-motion from videos. Since our proposed RST layers can

TABLE V: Performance comparison for different versions of RSTNet. Sq.09 and Sq.10 (ATE) are the ego-motion estimation results. From Abs Rel to $\delta<1.25^3$ are the depth estimation results.

RST Versions	Sq.09 (ATE)	Sq.10 (ATE)	Abs Rel	Sq Rel	RMSE	RMSE log	δ <1.25	δ <1.25^2	δ <1.25^3
RST-V1	0.0123 ± 0.0078	0.0121 ± 0.0095	0.1162	1.083	5.114	0.197	**0.860**	**0.957**	**0.980**
RST-V12	0.0118 ± 0.0070	0.0109 ± 0.0061	0.1120	1.067	5.024	0.195	**0.863**	**0.961**	**0.981**
RST-V123	0.0116 ± 0.0066	0.0106 ± 0.0056	0.1103	0.794	4.623	0.191	**0.880**	**0.963**	**0.984**
RST-FULL	0.0112 ± 0.0064	0.0104 ± 0.0053	0.1084	0.767	4.524	0.188	**0.891**	**0.974**	**0.991**

TABLE VI: Impact of image features (F_i), depth features (F_d), and flow features (F_f) on the ego-motion estimation of Sequence 09.

Features	F_i	F_d	F_f	F_{i+d}	F_{i+f}	F_{d+f}	F_{i+d+f}
$t_{\text{rel}}(\%)$	5.67	8.21	9.12	4.64	4.75	9.06	4.35
$r_{\text{rel}}(^\circ/100m)$	2.94	7.82	8.34	3.25	3.30	7.69	2.22

- t_{rel}: average translational RMSE drift (%) on length of 100m-800m.
- r_{rel}: average rotational RMSE drift ($^\circ/100m$) on length of 100m-800m.

capture detailed spatial and temporal features from consecutive input frames, the RSTNet can generate more accurate depth and ego-motion estimation results compared with other existing unsupervised learning networks. In addition, our proposed network takes a holistic approach to estimating key visual clues (dense depth, ego-motion, optical flow, and dynamic object) with the combination of multiple features (appearance, structure, and temporal features).

As demonstrated in the network complexity, our network full version requires significant amount of time for computation. This is far from real time requirement. This is the main limitation of our approach where real time inference is infeasible in current computing platforms. In general, most deep learning based VO approaches are short in term of real time performance due to the use of deep neural networks. But we believe the depth and ego-motion estimation with deep learning still has the potential in long time perspective. In the future, we would like to explore the opportunity to improve the time efficiency. This could be explored with new light weighted network architectures. Also we would like to combine geometric methods with deep learning networks together to not only improve the accuracy but also reduce computational time.

REFERENCES

[1] 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.

[2] V. Prasad and B. Bhowmick, "SfMLearner++: Learning Monocular Depth & Ego-Motion using Meaningful Geometric Constraints," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 2087–2096.

[3] R. Li, S. Wang, Z. Long, and D. Gu, "UnDeepVO: Monocular Visual Odometry through Unsupervised Deep Learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7286–7291.

[4] R. Garg, V. K. BG, G. Carneiro, and I. Reid, "Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue," in *European Conference on Computer Vision*. Springer, 2016, pp. 740–756.

[5] C. Godard, O. M. 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.

[6] 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.

[7] B. Wagstaff, V. Peretroukhin, and J. Kelly, "Self-Supervised Deep Pose Corrections for Robust Visual Odometry," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2331–2337.

[8] R. Li, D. Gu, and S. Wang, "DeepSLAM: A Robust Monocular SLAM System With Unsupervised Deep Learning," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 4, pp. 3577–3587, 2021.

[9] Z. Yin and J. Shi, "GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1983–1992.

[10] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging Into Self-Supervised Monocular Depth Estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3828–3838.

[11] V. Guizilini, R. Ambrus, S. Pillai, A. Raventos, and A. Gaidon, "3D Packing for Self-Supervised Monocular Depth Estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2485–2494.

[12] H. Jiang, L. Ding, Z. Sun, and R. Huang, "DiPE: Deeper into Photometric Errors for Unsupervised Learning of Depth and Ego-motion from Monocular Videos," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 061–10 067.

[13] Y. Zou, Z. Luo, and J.-B. Huang, "DF-Net: Unsupervised Joint Learning of Depth and Flow using Cross-Task Consistency," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 36–53.

[14] Y. Almalioglu, M. R. U. Saputra, P. P. de Gusmao, A. Markham, and N. Trigoni, "GANVO: Unsupervised Deep Monocular Visual Odometry and Depth Estimation with Generative Adversarial Networks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5474–5480.

[15] Y. Wang, P. Wang, Z. Yang, C. Luo, Y. Yang, and W. Xu, "UnOS: Unified Unsupervised Optical-Flow and Stereo-Depth Estimation by Watching Videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8071–8081.

[16] A. Gordon, H. Li, R. Jonschkowski, and A. Angelova, "Depth from Videos in the Wild: Unsupervised Monocular Depth Learning from Unknown Cameras," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8977–8986.

[17] W. Zhao, S. Liu, Y. Shu, and Y.-J. Liu, "Towards Better Generalization: Joint Depth-Pose Learning without PoseNet," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9151–9161.

[18] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. Reid, "Unsupervised Learning of Monocular Depth Estimation and Visual Odometry with Deep Feature Reconstruction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 340–349.

[19] T. Feng and D. Gu, "SGANVO: Unsupervised Deep Visual Odometry and Depth Estimation with Stacked Generative Adversarial Networks," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4431–4437, 2019.

[20] A. Ranjan, V. Jampani, L. Balles, K. Kim, D. Sun, J. Wulff, and M. J. Black, "Competitive Collaboration: Joint Unsupervised Learning of Depth, Camera Motion, Optical Flow and Motion Segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 240–12 249.

[21] C. M. Parameshwara, G. Hari, C. Fermüller, N. J. Sanket, and Y. Aloimonos, "Diffposenet: Direct differentiable camera pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 6845–6854.

[22] C. Wang, J. M. Buenaposada, R. Zhu, and S. Lucey, "Learning Depth from Monocular Videos using Direct Methods," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2022–2030.

[23] J. 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," *Advances in Neural Information Processing Systems*, vol. 32, pp. 35–45, 2019.

[24] Y. Wang, X. Ma, J. Wang, S. Hou, J. Dai, D. Gu, and H. Wang, "Robust AUV visual loop closure detection based on variational auto-encoder network," *IEEE Transactions on Industrial Informatics*, 2022.

[25] P. Z. Ramirez, M. Poggi, F. Tosi, S. Mattoccia, and L. Di Stefano, "Geometry Meets Semantics for Semi-supervised Monocular Depth Estimation," in *Asian Conference on Computer Vision*. Springer, 2018, pp. 298–313.

[26] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova, "Unsupervised Monocular Depth and Ego-motion Learning with Structure and Semantics," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[27] M. Klingner, J.-A. Termöhlen, J. Mikolajczyk, and T. Fingscheidt, "Self-Supervised Monocular Depth Estimation: Solving the Dynamic Object Problem by Semantic Guidance," in *European Conference on Computer Vision*. Springer, 2020, pp. 582–600.

[28] R. Ambrus, V. Guizilini, J. Li, and S. P. A. Gaidon, "Two Stream Networks for Self-Supervised Ego-Motion Estimation," in *Conference on Robot Learning*. PMLR, 2020, pp. 1052–1061.

[29] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1874–1883.

[30] S. Pillai, R. Ambruş, and A. Gaidon, "SuperDepth: Self-Supervised, Super-Resolved Monocular Depth Estimation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9250–9256.

[31] L. Zhou, J. Ye, M. Abello, S. Wang, and M. Kaess, "Unsupervised Learning of Monocular Depth Estimation with Bundle Adjustment, Super-Resolution and Clip Loss," *arXiv preprint arXiv:1812.03368*, 2018.

[32] T.-W. Hui, "Rm-depth: Unsupervised learning of recurrent monocular depth in dynamic scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 1675–1684.

[33] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8934–8943.

[34] V. Badrinarayanan, A. Handa, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling," *arXiv preprint arXiv:1505.07293*, 2015.

[35] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

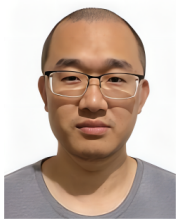
[36] S. Lee, S. Im, S. Lin, and I. S. Kweon, "Learning Residual Flow as Dynamic Motion from Stereo Videos," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1180–1186.

[37] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[38] D. Eigen, C. Puhrsch, and R. Fergus, "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network," *arXiv preprint arXiv:1406.2283*, 2014.

[39] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3213–3223.

[40] 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.



Feng Tuo received the BSc. degree in communication and information system from University of Electronic Science and Technology of China, Chengdu, China, in 2013, the MSc. Degree in electronic engineering, and the PhD. degree in computer science from University of Essex, Colchester, UK. in 2018 and 2021, respectively. His research interests include deep learning, computer vision, simultaneous localization and mapping (SLAM).

Dongbing Gu (SM'07) received the BSc. and MSc. degrees in control engineering from the Beijing Institute of Technology, Beijing, China, in 1985 and 1988, respectively, and the PhD. degree in robotics from the University of Essex, Colchester, U.K., in 2004.

From October 1996 to October 1997, he was an Academic Visiting Scholar with the Department of Engineering Science, University of Oxford, Oxford, UK. In 2000, he joined the University of Essex as a Lecturer. He is currently a Professor of Robotics with the School of Computer Science and Electronic Engineering, University of Essex. His research interests include robotics, multiagent systems, cooperative control, model predictive control, visual simultaneous localization and mapping, wireless sensor networks, and machine learning.