IEEE Robotics and Automation Letters (RAL) paper presented at the
2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Don't Forget The Past: Recurrent Depth Estimation from Monocular Video

Vaishakh Patil[1], Wouter Van Gansbeke[2], Dengxin Dai[1] and Luc Van Gool[1,2]

***Abstract*— Autonomous cars need continuously updated depth information. Thus far, depth is mostly estimated independently for a single frame at a time, even if the method starts from video input. Our method produces a time series of depth maps, which makes it an ideal candidate for online learning approaches. In particular, we put three different types of depth estimation (supervised depth prediction, self-supervised depth prediction, and self-supervised depth completion) into a common framework. We integrate the corresponding networks with a ConvLSTM such that the spatiotemporal structures of depth across frames can be exploited to yield a more accurate depth estimation. Our method is flexible. It can be applied to monocular videos only or be combined with different types of sparse depth patterns. We carefully study the architecture of the recurrent network and its training strategy. We are first to successfully exploit recurrent networks for real-time self-supervised monocular depth estimation and completion. Extensive experiments show that our recurrent method outperforms its image-based counterpart consistently and significantly in both self-supervised scenarios. It also outperforms previous depth estimation methods of the three popular groups.Please refer to our webpage [1] for details.**

I. Introduction

High precision depth estimation is essential for a variety of applications such as augmented reality, autonomous vehicles, and mobile robots. The last years have witnessed tremendous progress in depth estimation, especially after the wide deployment of deep neural networks. On one hand, supervised learning algorithms are constantly improving for depth estimation from RGB images [1], [2], [3], [4]. On the other hand, self-supervised depth estimation from camera motion are also steadily improving [5], [6], [7], [8]. Recently, several studies on depth completion were made, aiming at completing the depth map obtained by a high-end LiDAR sensor, namely HDL-64E, by using a paired image for guidance [9], [10], [11], [12], [13], [14].

Whereas great progress is being made in all three cases, none of those methods would seem optimal for mobile robot applications though. As they move, mobile platforms - be it cars or assistive robots - perceive the world more as a video stream than as isolated images. While videos are used in the training stage of self-supervised depth prediction methods for computing the view-synthesis loss across neighboring frames [5], [6], [7], [15], they ignore the intrinsic temporal dependency across frames at testing. The perceptual inputs along a trajectory and the underlying scene geometries are highly correlated [16]. Given our context of robotic applications, we propose a depth recovery method that both trains and tests on time series of data. This way, the perception-geometry correlation can be leveraged the best.

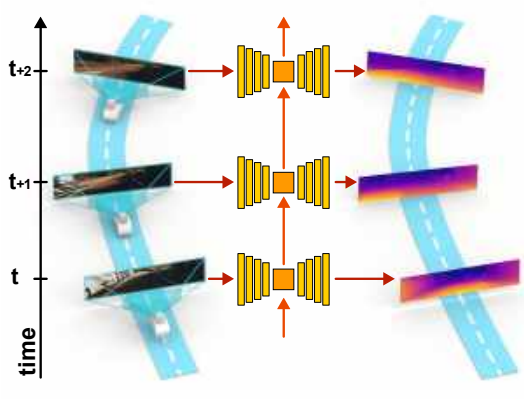

Fig. 1. Our method trains and tests on time series of data and produces accurate depth maps for robotic platforms which perceive the world more as a video stream than as isolated images.

Also, none of the three settings seem to be optimal by themselves. The setting of depth prediction from RGB images is cheap but requires large training datasets with accurate ground truth; the setting for self-supervised depth estimation from videos leverages ego-motion information (to some extent) but have yet to generate the best results; and the setting for depth completion with a LiDAR sensor and a camera yields good results but is quite costly.

Hence, in this work, we put the three different types of depth estimation, i.e. supervised depth prediction from RGB images [3], [4], self-supervised depth prediction with monocular videos [7], [15] and self-supervised depth completion [11] but with monocular videos, into a common framework, and then integrate their corresponding 'backbone' networks with a convolutional LSTM (ConvLSTM) such that spatiotemporal information across frames can be exploited for more accurate depth estimation.

ConvLSTMs have been designed to exploit temporal information but it is still unclear how they can be trained properly for self-supervised depth estimation from current literature. We claim to be the first to propose an novel and effective strategy to integrate ConvLSTM into the unified depth estimation pipeline. The training is challenging because 1) the size of the feature maps is large for dense prediction

This work is supported by Toyota Motor Europe.

[1]Vaishakh Patil, Dengxin Dai and Luc Van Gool are with the Toyota TRACE-Zurich at the Computer Vision Lab, ETH Zurich, 8092 Zurich, Switzerland {patil, dai, vangool}@vision.ee.ethz.ch

[2] Wouter Van Gansbeke and Luc Van Gool are with the Toyota TRACE-Leuven at the Dept. of Electrical Engineering ESAT, KU Leuven, 3001 Leuven, Belgium {wouter.vangansbeke, luc.vangool}@kuleuven.be

[1]https://www.trace.ethz.ch/publications/2020/rec_depth_estimation/

which limits the sequence length due to memory issues; and 2) under the standard training strategy, ConvLSTM based networks need a long sequence to learn the hidden state properly. Our training strategy addresses these issues.

In summary, this work makes three contributions: 1) a novel recurrent network to exploit spatiotemporal information for depth estimation, 2) an innovation to effectively train a ConvLSTM based network for dense prediction tasks with video inputs; and 3) extensive experiments and detailed ablation studies. Experiments show that our recurrent method outperforms its image-based counterpart and the current SOTA methods consistently and significantly in all the considered scenarios.

II. Related Work

Supervised Depth Estimation from RGB Images. A large body of work focuses on depth estimation from images with varying settings: from using image pairs [17], [18], to using multiple overlapping images captured from different viewpoints [19] Here we summarize work related to the supervised learning of depth from a single RGB image. [20] is among the earliest work popularizing this idea. Local image statistics are used to infer 3D planes for local patches and the final results are optimized globally over a defined Markov Random Field. Later on, deep convolutional neural networks were used for this task [2], [21], [22], [3]. The research focus was mainly on improving the network architecture [2], [3], formulating multi-task learning [21], and combining CNNs and CRFs [22]. In order to alleviate the dependency on large-scale ground-truth depth images, methods that learn directly from stereo pairs were developed [18]. The core idea is to use left-right view similarity as the supervisory signal. This line of work has been further extended to a semi-supervised setting [23], where direct supervision from LiDAR sensors and indirect supervision from image warping are combined.

Self-supervised Depth Estimation from Videos. To lower the dependency on ground truth depth images, many recent works have shifted the focus to self-supervised depth estimation from monocular videos by using view-synthesis or its variants as the supervisory signal [5], [6], [7], [24], [8], [15], [25]. While promising results have been shown, the training of self-supervised methods requires careful hyper-parameter tuning and suffers from scale ambiguity, which needs to be addressed, e.g. by using stereo images [24] or by data normalization [6]. While consecutive video frames are used for the view-synthesis loss, the spatiotemporal information is not exploited, especially at testing time.

Depth Completion. While steady progress has been made for depth estimation from RGB images, the performance can be improved when assisted by other sources. One notable example is that of sparse depth inputs, either from cheap LiDAR sensors [26] or from SLAM or structure-from-motion systems [27], [28]. There has been a large body of work [9], [10], [11], [12], [13], [14] developed for the task of depth completion defined by the KITTI Depth Completion Benchmark [9]. Methods have also been developed for depth completion with sparse Radar points recently [29]. The main research focus of this strand is how to spatially propagate Automotive LiDAR depth data under image guidance. The established knowledge, such as the design of network architectures for spatial propagation, can be borrowed to design our recurrent method. The main focus of our work, however, lies in how to fuse or propagate depth information over frames.

Depth Estimation for Videos. Our method is designed for online depth estimation in videos. Similar idea of online estimation is proposed in recent works [30], [31], where they use ConvLSTM but in supervised framework. There are also earlier methods for offline depth estimation from videos [32], in which local motion cues and optical flow are used to produce temporally consistent depth maps.

III. Approach

As stated in Sec. I, depth estimation has been tackled under multiple settings, each has its own strengths and weaknesses. These systems, however, are mostly image-based and lack the capability of integrating information over video sequences obtained by moving robotic platforms. In this section, we first summarize three existing depth estimation methods in a unified formulation and then define our recurrent framework for learning time series of depth maps for all the three methods.

Before going into the details, we define certain notations used by the methods. Let us denote by $I(x, y)$ the RGB vector-valued image, $\bar{D}(x, y)$ the input sparse depth, and $D(x, y)$ the ground truth depth map; all three have the same dimensions $H \times W$. While $I(x, y)$ is dense, $D(x, y)$ has regions with missing values which are indicated by zero. $\bar{D}(x, y)$ is much sparser compared to $D(x, y)$; it is usually a sub-sampled depth map from $D(x, y)$ to simulate sparse input patterns that can be obtained via a low resolution LiDAR sensor. The detailed sampling procedures for varying sparse input patterns are given in Sec. III-C. The timestamp of time series data is denoted by t in subscript, where $t \in \{1, 2, ..., T\}$ with T the total number of frames of a data sequence. We assume that the image and the depth maps are synchronized throughout the sequence.

A. Supervised Depth Prediction with RGB Images

Supervised depth prediction with monocular RGB images has been a popular research topic. Tremendous progress has been made in the past years [4], [33], [21], [2], [34]. The task is to learn a function $f : I \rightarrow \hat{D}$, where $\hat{D}$ has the same resolution $H \times W$ and has predictions for all pixel coordinates (x, y) where $x \in \{1, .., H\}$ and $y \in \{1, ..., W\}$. We represent the depth estimation network as an encoder-decoder type of network architecture. Then, the learning takes the form:

$$X = f_{\text{encoder}}(I), \quad (1) \qquad \hat{D} = f_{\text{decoder}}(X), \quad (2)$$

where X is the summarized representation by the encoder, which is compact and will be used to pass information across video frames for depth estimation from a video sequence as presented in Sec. III-D.

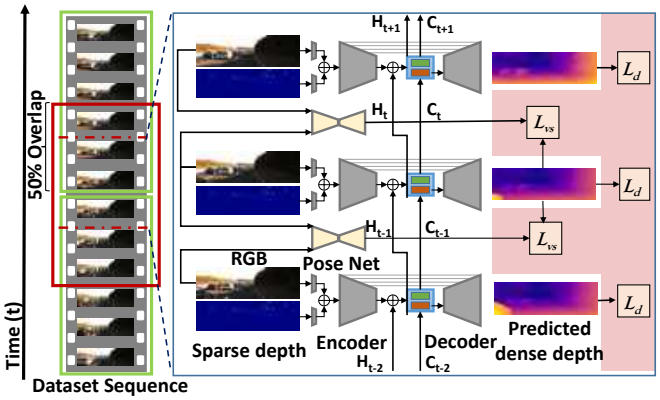

Fig. 2. The pipeline of our recurrent learning framework for depth recovery with monocular video and sparse depth sensing.

We follow [3] and use the berHu loss, which gives slightly better results than the L1 and L2 loss. Let us define a binary mask $M(x, y)$ of dimensions $H \times W$, where $M(x, y) = 1$ defines (x, y) locations of valid values for the ground truth depth map $D(x, y)$. The loss can then be formulated as

$$\mathcal{L}_{\text{berHu}} = \sum_{t} \|M_t \circ D_t - M_t \circ \hat{D}_t\|_\delta, \tag{3}$$

where '$\circ$' denotes the Hadamard product in order to ignore the invalid pixels of the ground truth depth picture D_t when computing the loss, and $\|.\|_\delta$ is set by following [3].

B. Self-supervised Depth Prediction with Monocular Video

Self-supervised depth estimation from monocular video has been quite successful in recent years [15], [17], [18], [5]. We will mostly follow the presentation of the state-of-the-art method Monodepth2 [15] in this section. We represent the depth estimation network with the same encoder-decoder network as defined in Eq. 1 and Eq. 2. Since there is no ground-truth depth map $D(.)$, the view-synthesis loss is used instead of the standard supervised loss functions.

In particular, if K denotes the camera intrinsic matrix, and $\Phi_{t\to t+\Delta t}$ the relative camera pose from view t to a neighboring view $t + \Delta t$, the warped image is:

$$I_{t+\Delta t\to t} = I_{t+\Delta t}\left\langle K\Phi_{t\to t+\Delta t}\hat{D}_t K^{-1}\right\rangle, \tag{4}$$

where $\langle . \rangle$ is a bilinear sampling function to sample the source image. The view-synthesis loss of our method is defined as

$$\begin{aligned}\mathcal{L}_{vs(t+\Delta t\to t)} = \frac{\alpha}{2}(1 - SSIM(I_t, I_{t+\Delta t\to t})) \\ +(1-\alpha)\|I_t - I_{t+\Delta t\to t}\|_1,\end{aligned} \tag{5}$$

where $\alpha = 0.85$.

In addition to estimating depth, the model also needs to estimate the camera pose $\Phi_{t\to t+\Delta t}$. This involves training a pose estimation network that predicts the corresponding camera transformations with the same sequence of frames as input. The $\mathcal{L}_{vs(t+\Delta t\to t)}$ is computed at multiple scales of the decoder, similar to [15].

The final view-synthesis loss is aggregated over all considered source (neighboring) frames. In this work, $\Delta t \in \{-1, 1\}$, i.e. for each frame t, the previous frame $t-1$ and the next frame $t+1$ are used to compute the loss. Following [15], at each pixel, we use the minimum photometric error over all source images. Thus, the final view-synthesis loss is

$$\mathcal{L}_{vs} = \min_{\Delta t\in\{-1,1\}} \mathcal{L}_{vs(t+\Delta t\to t)}. \tag{6}$$

Following [15], we also use the edge-aware smoothing loss:

$$\mathcal{L}_{smooth} = |\partial_x \hat{D}_t^*|e^{\|-\partial_x I_t\|} + |\partial_y \hat{D}_t^*|e^{\|-\partial_y I_t\|}, \tag{7}$$

where $\hat{D}_t^*$ is mean-normalized $\hat{D}_t$ to avoid shrinking the depth values. Our learning algorithm is trained with a combined loss:

$$\mathcal{L}_{\text{self\_pred}} = \mu\mathcal{L}_{vs} + \nu\mathcal{L}_{smooth}. \tag{8}$$

μ represents the pixel-wise masking of the view-synthesis loss to ignore certain objects. This addresses the problem that a car, travelling at the same speed as the camera, will be predicted at infinite depth [15].

C. Self-supervised Depth Completion with Monocular Video and Sparse Depth Maps

Supervised depth prediction methods (Sec. III-A) are able to achieve good performance but require large training sets with accurate ground truth depth and have difficulty to generalize to new scenarios. Self-supervised depth prediction methods (Sec.III-B) are easy to 'generalize', but have yet to yield the state-of-the-art results. In this section, we present another stream of method called self-supervised depth completion following [11].

The method requires a monocular video $(I_1, I_2, ..., I_T)$ and synchronized sparse depth maps $(\bar{D}_1, \bar{D}_2, ..., \bar{D}_T)$ as inputs. While it is hard to obtain dense depth map D_t, sparse depth map $\bar{D}_t$ are relatively cheap and easy to acquire, e.g. via 2D LiDAR sensors. Compared to self-supervised depth prediction, this vein of research also focuses on developing a suitable network architecture to better fuse the information from these two modalities. Typical examples include a simple concatenation of the two inputs as done in [27] or adding a distance transformation map to indicate the location of the sparse values [28].

We process both inputs individually with few convolutions before fusing them together. We find that this method works better than the ones used in [27] and [28]. As to the convolutional network, we again use an encoder-decoder type of network architecture. The encoder takes the form:

$$X = f_{\text{encoder}}(I, \bar{D}), \tag{9}$$

and the decoder is the same as defined in Eq. 2. As to the loss functions, in addition to the view-synthesis loss (Eq. 6) and the edge-aware smoothing loss (Eq. 7) used for self-supervised depth prediction task, the berHu loss is also used but applied to the input sparse depth map $\bar{D}_t$ and its binary mask $\bar{M}(x, y)$.

The total loss for self-supervised depth completion is

$$\mathcal{L}_{\text{self\_comp}} = \lambda\mathcal{L}_{berHu}^{\text{sparse}} + \mu\mathcal{L}_{vs} + \nu\mathcal{L}_{smooth}. \tag{10}$$

Sparsity loss can act as supplemental loss to View-synthesis loss. The self-supervision from sparsity loss can handle scale ambiguities and textureless regions. It also stabilizes the

training process and helps to converge faster. On the other hand, the view-synthesis loss is computed densely and can alleviate the effect of noise in the sparse depth maps.

We evaluate our method with three types of sparse patterns. Following the literature [27], our first pattern denoted by $\bar{D}^{\text{rand}}$ is created by randomly sampling pixels from ground truth depth image. The second pattern denoted by $\bar{D}^{\text{line}}$ is obtained by sampling the scanning lines ground truth depth images at a constant stride. The third pattern is the dense depth map D itself, which is still sparse compared to images.

D. Learning Time Series of Depth Maps

This section presents a framework to extend the three groups of methods such that they both train and test on time series of data. We formulate the depth recovery problem as a translation problem from a spatiotemporal sequence of multimodal data (i.e. images and sparse depth maps) to a spatiotemporal sequence of data (i.e. dense depth maps). In order to model the spatiotemporal dependencies, we add the ConvLSTM module to the backbone network presented in the depth prediction section. The ConvLSTM determines the future state of a certain cell in the grid from the inputs and past states of its local neighbors. As argued in [35], if the hidden state is considered as the hidden representations of visual structures (objects), then ConvLSTM is able to capture motions of those visual components via its transitional kernels. Similarly, for the task of learning depth maps from monocular videos, we try to capitalize on temporal information to boost performance. The correlation of the geometry of the scene and the perceived visual stimuli along motion trajectories should be captured and exploited. Another well established approach to exploit temporal information is by concatenating multiple frames at the input. However, these approaches don't scale to longer sequences and require expensive 3D convolutions. We argue that long sequences are potentially beneficial for depth estimation from video combined with online learning methods.

Our encoder-decoder network, defined in Eq. 1 and Eq. 9, generates feature representations at varying levels. The output by the encoder X is chosen as the input to our ConvLSTM. The choice is made due to the compactness of X and its high information density, which leads to a more efficient optimization for the ConvLSTM. More specifically, the learning process at frame t starts with spatial convolutions with the encoder to get X_t, which is followed by temporal convolutions with the ConvLSTM

$$H_t, C_t = f_{\text{ConvLSTM}}((X_t \oplus H_{t-1}), C_{t-1}), \qquad (11)$$

and then followed by spatial convolutions with the decoder, such that $\hat{D} = f_{\text{decoder}}(H_t)$. The whole network is trained in an end-to-end manner. Its pipeline is sketched in Fig. 2 for the most complicated task self-supervised depth completion. The pipelines for supervised depth prediction and self-supervised depth prediction can be inferred according to their inputs and loss functions.

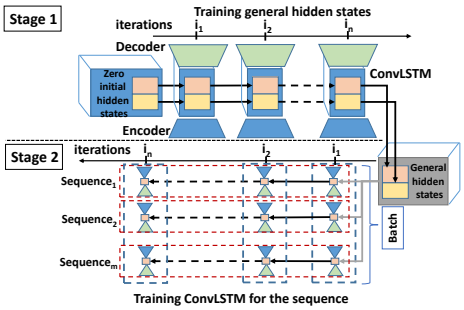

Fig. 3. Training procedure for hidden state of ConvLSTM.

IV. Training Framework

A. Network Architecture

The network architecture consists of a depth prediction network and a pose network. The encoder branch of both networks contains separate ResNet-18 modules [36]. The decoder unit of the depth network contains four upconvolutional blocks inspired from DispNet [37]. The output of the encoder is connected with the ConvLSTM [35] module. The ConvLSTM module receives hidden state H_{t-1} and cell state C_{t-1} of ConvLSTM from the previous frame $t-1$ (details in Sec. III-D). The output of this network is disparity extracted at different spatial resolution from each unit of decoder. The pose decoder consist of stack of Conv(1) and Conv(3) blocks and produces a 6 element vector representing axis angle and translation. For simplicity, we combine the hidden representation H and cell state C in the ConvLSTM and refer to it further as the *hidden state*. The *initial hidden state* refers to the initialization of the *hidden state*.

B. ConvLSTM Training Strategy

In vanilla LSTM based network training, the default strategy is to initialize the hidden state to zero. This is a well established practise in sequence to sequence learning models. Here, the impact of the initial hidden state is either trivial or the length of the sequence is relatively long compared to the size of the hidden state. In case of training a ConvLSTM based network with monocular video, the size of the feature maps in the bottleneck increases drastically. This demands increased capacity of the hidden state due to the concatenation (Eq. 11). With more learnable parameters and shorter sequence lengths during training, the effect of the initial hidden state becomes dominant. In our training procedure, we try to mitigate the effects of a zero initial hidden state. The training is divided into two stages as depicted in Fig. 3. In the first stage, the initial hidden state is considered as a learnable parameter. The training begins with the hidden state initialized with zeros. Further on, for every iteration we first load the initial hidden state and backpropagate through the hidden state to update the initial hidden state of the ConvLSTM. The training is performed on fully randomized batches. This procedure allows us to learn a general initial hidden state. In the second stage, the trained initial hidden state is used at the start of every new sequence. Hence, the training is performed on the video

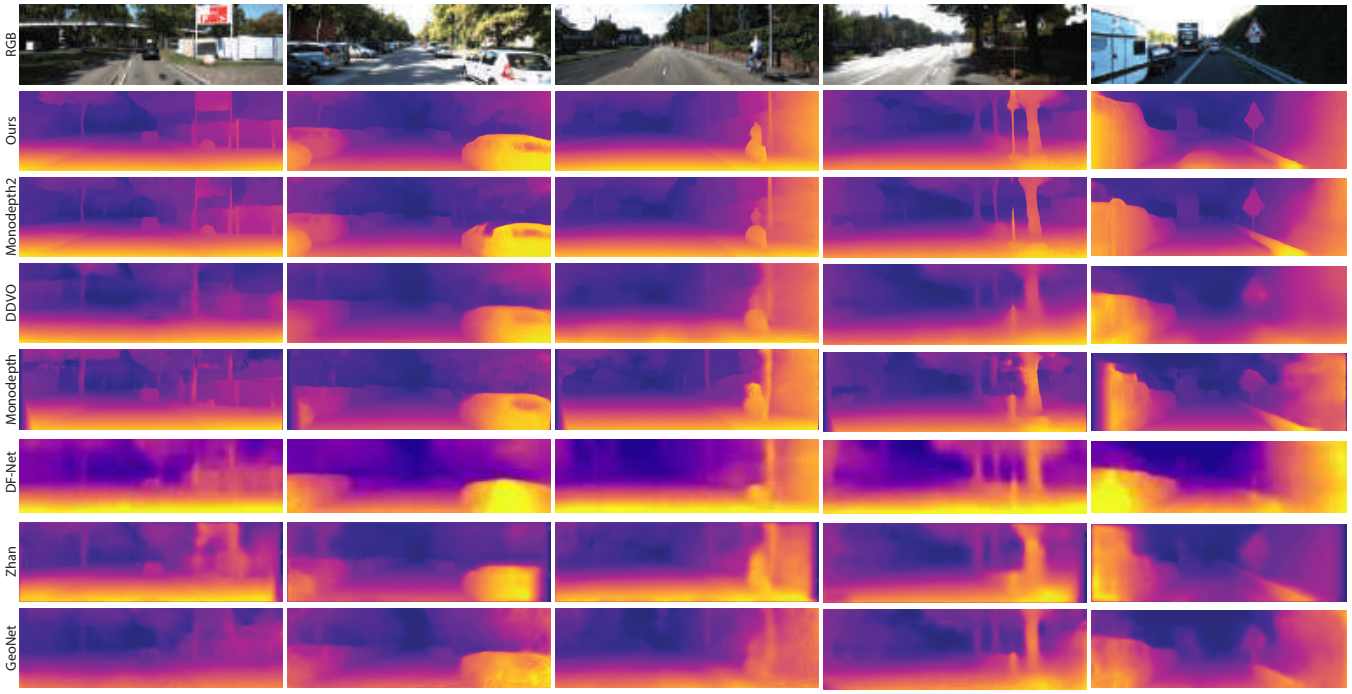

Fig. 4. **Qualitative results on the KITTI Eigen split.** Our self-supervised recurrent method generates more accurate depth maps compared to competing methods [15], [6], [18], [8], [24], [7], especially for small objects like poles, tree trunks and traffic signs.

sequences as opposed to the previous stage. This means that the weights of ConvLSTM module are adjusted based on the pretrained hidden state from the first stage. The training weights from the second stage along with learned initial hidden state from the first stage are used during testing. It enables the ConvLSTM to adapt to the sequence faster, resulting in superior performance as shown in Sec. V-B. The weights are updated for every frame to optimize for training speed and memory footprint (i.e. truncated BPTT [38] with window size of 1). Although this technique is effective for a large dataset with long sequences, it can result in overfitting when updating the weights per single image. To alleviate this issue, we also train in batches during the second stage. This allows us to update the batch statistics and average the gradients. This greatly benefits the generalization. Not all sequences have the same length, therefore we first divide the original video sequences into smaller "sub-sequences" for training, and load multiple random "sub-sequences" in parallel afterwards. This whole procedure is depicted in Fig. 3. The batch is shown by the dotted blue rectangle. The influence of the sequence length is evaluated in Sec. V-B.

C. Implementation Details

Training Details. The weights ν and λ are respectively used for the smoothing and sparsity loss in Eq. 10. The former is set to 0.001 while the latter is iteration dependent. In fact, $\lambda(i)$ changes during training to prevent overfitting on a low number of LiDAR points. We start with the view-synthesis loss and smoothing loss first and gradually increase the influence of the sparsity loss afterwards. This happens linearly with the number of iterations i, such that $\lambda(i) = 10^{-2} \cdot \min(1, 10^{-3} \cdot i)$. After all, the network should be prevented from learning the identity function in order to discover semantics and depth. For all experiments, we use a batch size of 12, with the Adam optimizer and a learning rate of 10^{-4}. The images are resized to a resolution of 192x640 as in Monodepth2 [15] baseline. Training takes 10 epochs for the first stage, while we finetune on sequences during the second stage for 20 epochs. For the encoder, we used pretrained ImageNet [43] weights. This is important to achieve competitive results as in [40], [33], [15], [39]. All the other weights are initialized with He initialization [44], except for the biases of the convolution layer at the forget gate. To make the ConvLSTM focus on the hidden state at the start of training, we set the biases to 1 as in [45]. We replaced the Tanh activations in the ConvLSTM with ELU [46] in order to match the scale with the output of encoder (Eq. 11).

V. Experiments

To show the effectiveness of our approach, we address the previously defined conditions in Sec. III: 1) the supervised depth prediction setup with raw LiDAR ground truth, 2) the self-supervised depth prediction setup and 3) the self-supervised depth completion setup. For each case, numbers are reported on the KITTI dataset in order to evaluate with other monocular depth estimation methods. We consider Monodepth2 [15] as our baseline in the following self-supervised depth estimation experiments.

The supervised depth prediction and self-supervised depth completion experiments are evaluated on the Eigen split defined by Eigen *et al.* [21]. This split contains 28 raw KITTI sequences for training, 5 sequences for validation and 28 sequences for testing, all with variable length. Our approach is

TABLE I

RESULTS FOR SELF-SUPERVISED DEPTH PREDICTION.

Method	↓ RMSE	↓ RMSE(log)	↓ Abs Rel Diff	↓ Sq Rel Diff	↑ $\delta < 1.25$	↑ $\delta < 1.25^2$	↑ $\delta < 1.25^3$
SFMLearner [5]	6.709	0.270	0.183	1.595	0.734	0.902	0.959
DDVO [6]	5.583	0.228	0.151	1.257	0.810	0.936	0.974
GeoNet [7]	5.567	0.226	0.149	1.060	0.796	0.935	0.975
CC [39]	5.464	0.226	0.148	1.149	0.815	0.935	0.973
EPC++ [40]	5.350	0.216	0.141	1.029	0.816	0.941	0.976
Struct2depth (w/o ref.) [41]	5.291	0.215	0.141	1.026	0.816	0.945	0.979
GL-Net (w/o ref.) [42]	5.230	0.210	0.135	1.070	0.841	0.948	0.980
Monodepth2 [15]	4.863	0.193	0.115	0.903	0.877	0.959	0.981
Ours (Average over 5 runs)	**4.730**	**0.188**	**0.112**	**0.863**	**0.879**	**0.960**	**0.981**
Ours (Best)	**4.650**	**0.187**	**0.111**	**0.821**	**0.883**	**0.961**	**0.982**

TABLE II

RESULTS OF SELF-SUPERVISED DEPTH COMPLETION.

Input	Method	↓ RMSE	↑ $\delta < 1.25$
$\bar{D}^{\text{rand}}_{500}$	Image-based	2.885	0.970
	Recurrent	**2.738**	**0.973**
$\bar{D}^{\text{line}}_{8}$	Image-based	2.750	0.962
	Recurrent	**2.586**	**0.968**
$\bar{D}^{\text{line}}_{64}$	Ma *et al.* [11]	1.922	0.985
	Image-based	1.653	0.988
	Recurrent	**1.592**	**0.990**

not limited to the fixed sequence length adopted during training. To show this generalization towards longer sequences, we always evaluate on the complete video sequences during testing. Only for the self-supervised depth prediction task, we use a filtered version in order to leave out static frames, as defined by Zhou *et al.* [5]. Furthermore, to achieve absolute depth, our predictions are rescaled with the median ground truth depth per frame, as done in previous works [15], [41], [5], [39], [40], [6]. It is worth noting that the predictions of the self-supervised depth completion setup do not require rescaling since the sparsity loss (Eq. 10) enables the predictions to be scale-aware. The quantitative results are reported on the selected 697 frames from the 28 test sequences unless mentioned otherwise. In all our experiments, we cap the maximum predictions of all networks to 80m.

A. Analysis

In this section, we discuss the qualitative and quantitative results. We achieve a new state-of-the art for both self-supervised setups, proving its effectiveness.

Self-Supervised Depth Prediction The results in Table I shows that our method outperforms recent state-of-the-art methods. We improve over our baseline by a relatively large margin (-0.133m RMSE). The qualitative results are depicted in Fig. 4. The baseline method only use short-range video information when computing the view-synthesis loss, while our method leverages longer-range temporal information. Highly reflective surfaces (e.g. mirrors) or dynamic objects can still cause problems due to limitations of the self-supervision loss. We do not report results obtained by refining the model during test time using test images as in [41] [42].

Self-Supervised Depth Completion. We perform experiments with three sparse patterns as defined in Sec. III-C. For all patterns, we compare our recurrent method to its image-based counterpart. Since our method also uses the input patterns of 64 LiDAR lines by Velodyne HDL-64E, we report the results on the common 652 images of Eigen set [21] and the KITTI depth benchmark dataset for which the corrected ground truth are available. The results are reported in Table II. The table shows that the proposed recurrent framework outperforms its image-based counterpart for all three different sparse patterns. Our method also outperforms the state-of-the-art self-supervised depth completion method [11]. In addition to the use of longer-range temporal information, the better performance can also be attributed to the good features of our baseline Monodepth2: pixel-wise masking of the loss and better strategy to handle occlusions.

Supervised Depth Prediction. The quantitative results are shown in the Table III. Performing regression towards the re-projected LiDAR points is not ideal, due to the noisy LiDAR data [10]. We hypothesize that our recurrent approach can add consistency and produce more accurate predictions (-0.172m RMSE). This can be supported by 1) only marginal improvement is observed, when training on the corrected KITTI ground truth (dense), 2) δ_1 is considerably higher (+1%) than in our baseline supporting our claim. In this setup we are still outperformed by [4]. However, they use a complex network architecture (ResNet-101) with fully connected layers with inference time of 500 ms. Thi is not applicable to real-time depth prediction tasks, as in autonomous driving compared to our inference time of 10 ms. Note that, here we re-evaluate [4] with our setting. Also, we report corrected result of [47] in supervised setting based

TABLE III

RESULTS OF SUPERVISED DEPTH PREDICTION.

Method	↓ RMSE	↓ Abs Rel Diff	↑ $\delta < 1.25$
Eigen *et al.* [2] fine	7.156	0.215	0.692
Liu *et al.* [22]	6.986	0.217	0.647
Kumar *et al.* [31]	5.187	0.137	0.809
Wang *et al.* [47]*	5.106	0.128	0.836
Kuznietsov *et al.* [23]	4.621	0.113	0.862
Yang *et al.* [33]	4.442	0.097	0.888
Guo *et al.* [34]	4.422	0.105	0.874
Zhang *et al.* [30]	4.139	0.104	0.883
Fu *et al.* [4]*	**3.714**	**0.099**	**0.897**
Ours (w/o recurrent)	4.320	0.113	0.874
Ours	**4.148**	**0.102**	**0.884**

TABLE IV

RESULTS ON EIGEN SPLIT WITH LIDAR SUPERVISION EVALUATED FOR DIFFERENT ACTIVATION LAYERS AND SEQUENCE LENGTHS.

Activat.	Frames	RMSE ↓	Abs Rel ↑	$\delta < 1.25$ ↑	$\delta < 1.25^2$
None	30	4.210	0.108	0.881	0.965
Tanh	30	4.370	0.115	0.874	0.964
ReLU	30	4.173	0.104	**0.884**	0.965
ELU	30	**4.148**	**0.102**	**0.884**	**0.966**
ELU	15	4.234	0.107	0.882	0.964
ELU	50	4.155	0.103	**0.884**	**0.966**
ELU	100	4.170	0.103	0.883	0.965

on predictions provided by the authors and omit erroneous results for the unsupervised case in [47].

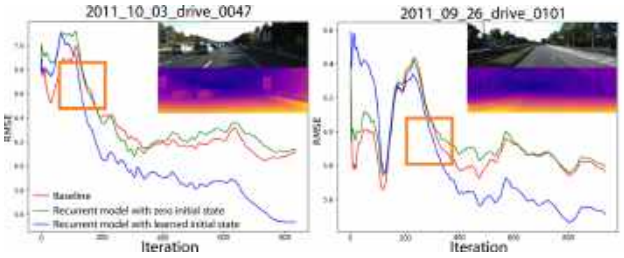

Fig. 5. Accumulated average RMSE (RMSE averaged over all previous frames) for three KITTI video sequences.

In Fig. 6, we evaluate the performance of our method as a function of the number of sparse points in $\bar{D}^{\text{rand}}$ and the number of scanning lines in $\bar{D}^{\text{line}}$. As expected, our method benefits from having denser depth samples as the inputs for both evaluated scenarios. Our recurrent framework is able to exploit the spatial and temporal structures of the scenes in the case of sparse points and scanning lines as well and consistently outperforms its frame-based counterpart. The improvement in RMSE score drops as number of input points increases. When supervision from LiDAR gets stronger, the reliance on other sources decreases.

B. Ablation Study

Pretrained Initial Hidden State. We compare zero-initialized training strategy with ours and report the results over KITTI sequences in Fig. 5. The figure shows that training the initial hidden state as a variable is more effective

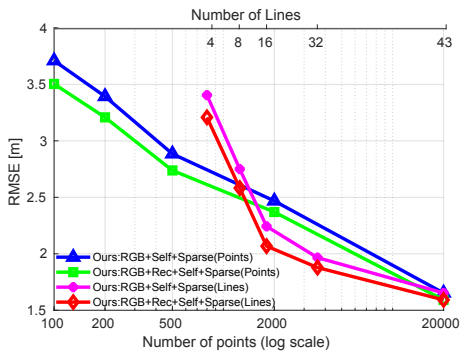

Fig. 6. Performance of our method as a function of the number of sparse points and the number of scanning lines.

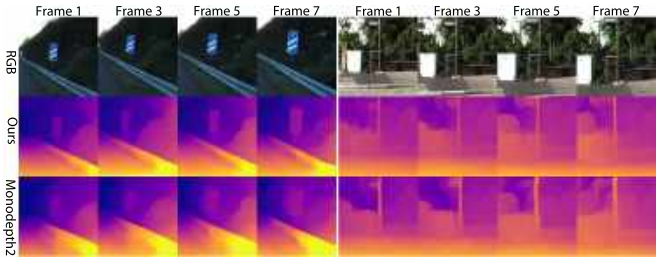

Fig. 7. Recurrent method demonstrates better temporal Consistency on KITTI video sequences over image based method.

than using zeros as initial states. The pretrained initial state helps to speed up adaptation and improves generalization at the beginning of a sequence. For example, we observe better initial predictions for the recurrent model with a learnable initial hidden state in (eg. sequence 47) Fig. 5 than with the zero initialization. However, in sequence 101 we show a counter example. Interestingly, the network is still able to outperform the baseline over time in this sequence. As one can see from the figure, our training strategy achieves considerable and consistent improvement over the zero-initialized training after certain number of frames. The improvements are noticeable when the car is stationary or in constant motion. The re-scaling factor in the self-supervised setup varies less in those regions.

TABLE V

REL. IMPROVEMENT OF RECURRENT METHOD COMPARED TO BASELINE

Method	Δ RMSE	Δ $\delta < 1.25$	Δ ARTE
Self sup. depth prediction	-0.133	0.002	-0.010
Self sup. depth completion	-0.061	0.002	-0.005

Temporal consistency. We also evaluate our method for temporal consistency Fig. 7. The quantitative metrices defined by [30] are not suitable for Datasets with sparse ground truth. We define our evaluation metric, Absolute Relative Temporal Error(ARTE), as follows: $\frac{1}{T}\sum_{i\in T}\frac{|(|\hat{D}_i-\hat{D}_{i-1}|-|D_i-D_{i-1}|)|}{|D_i-D_{i-1}|+\epsilon}$. We set ϵ to 0.001 and evaluate our predictions for self-supervised depth estimation first. Compared to the image-based baseline, our frame-recurrent method reduces the ARTE from 0.1401 to 0.1297. This improvement is also reflected in the RMSE (Table V). Additionally, we perform the same experiment for depth completion. The improvement over the image-based baseline is lower, i.e. 0.005. This is in line with the numbers in Table II, indicating that temporal consistency is more beneficial when fewer LiDAR points are available as input.

Activation Functions. The Tanh activation was introduced in LSTMs to deal with vanishing gradient problem in very long sequences. Intuitively, we expect to see better results with activations which preserve the input range. This is not the case for Tanh. The scale of Tanh does not match the scale of the input, a necessity for concatenation of the input with the hidden state. We evaluate the effect of different activation functions operating on the states inside the ConvLSTM. The results are shown in the Table IV. In our case, Tanh is inferior

to ReLU, ELU [46] and even to no activation. The lowest RMSE score is achieved with ELU.

Training Sequence Length. The effect the training sequence length is shown in Table IV. Experiments show that a sequence length of 30 strikes a good balance between performance and cost. Using short sequences leads to worse results; using longer ones does not boost the performance further. Training on very long sequences, 100 or higher, achieves similar results. This means that the ConvLSTM is able to capture temporal information by propagating the hidden state through the whole sequence.

VI. Conclusion

This work has introduced a novel method for estimating time series of depth maps with monocular video and optionally sparse depth. Our method exploits the spatio-temporal structures over data frames both at train and test time for accurate depth maps. Specifically, a recurrent framework has been developed and evaluated for three different tasks: supervised depth prediction, self-supervised (SS) depth prediction and self-supervised depth completion. For both SS scenarios, we outperform current SOTA methods significantly.

References

[1] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *CVPR*, 2012.

[2] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *NIPS*, 2014.

[3] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *3DV*, 2016.

[4] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep Ordinal Regression Network for Monocular Depth Estimation," in *CVPR*, 2018.

[5] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *CVPR*, 2017.

[6] C. Wang, J. Miguel Buenaposada, R. Zhu, and S. Lucey, "Learning depth from monocular videos using direct methods," in *CVPR*, 2018.

[7] Z. Yin and J. Shi, "Geonet: Unsupervised learning of dense depth, optical flow and camera pose," in *CVPR*, 2018.

[8] Y. Zou, Z. Luo, and J.-B. Huang, "Df-net: Unsupervised joint learning of depth and flow using cross-task consistency," in *ECCV*, 2018.

[9] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity invariant cnns," in *3DV*, 2017.

[10] W. Van Gansbeke, D. Neven, B. De Brabandere, and L. Van Gool, "Sparse and noisy lidar completion with rgb guidance and uncertainty," *MVA*, 2019.

[11] F. Ma, G. V. Cavalheiro, and S. Karaman, "Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera," *ICRA*, 2019.

[12] M. Jaritz, R. De Charette, E. Wirbel, X. Perrotton, and F. Nashashibi, "Sparse and dense data with cnns: Depth completion and semantic segmentation," in *3DV*, 2018.

[13] X. Cheng, P. Wang, and R. Yang, "Depth estimation via affinity learned with convolutional spatial propagation network," in *ECCV*, 2018.

[14] A. Eldesokey, M. Felsberg, and F. S. Khan, "Confidence propagation through cnns for guided sparse depth regression," *arXiv preprint*, 2018.

[15] C. Godard, O. M. Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *ICCV*, 2019.

[16] P. Agrawal, J. Carreira, and J. Malik, "Learning to see by moving," in *ICCV*, 2015.

[17] R. Garg, B. V. Kumar, G. Carneiro, and I. Reid, "Unsupervised cnn for single view depth estimation: Geometry to the rescue," in *ECCV*, 2016.

[18] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *CVPR*, 2017.

[19] Y. Furukawa and C. Hernández, "Multi-view stereo: A tutorial," *Found. and Trends in Comp. Graphics and Vision*, 2015.

[20] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: Learning 3d scene structure from a single still image," *PAMI*, pp. 824–840, 2009.

[21] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *ICCV*, 2015.

[22] F. Liu, C. Shen, G. Lin, and I. Reid, "Learning depth from single monocular images using deep convolutional neural fields," *PAMI*, 2016.

[23] Y. Kuznietsov, J. Stuckler, and B. Leibe, "Semi-supervised deep learning for monocular depth map prediction," in *CVPR*, 2017.

[24] H. Zhan, R. Garg, C. Saroj Weerasekera, K. Li, H. Agarwal, and I. Reid, "Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction," in *CVPR*, 2018.

[25] A. Gordon, H. Li, R. Jonschkowski, and A. Angelova, "Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras," in *ICCV*, 2019.

[26] Y. Liao, L. Huang, Y. Wang, S. Kodagoda, Y. Yu, and Y. Liu, "Parse geometry from a line: Monocular depth estimation with partial laser observation," in *ICRA*, 2017.

[27] F. Ma and S. Karaman, "Sparse-to-dense: Depth prediction from sparse depth samples and a single image," in *ICRA*, 2018.

[28] Z. Chen, V. Badrinarayanan, G. Drozdov, and A. Rabinovich, "Estimating depth from rgb and sparse sensing," in *ECCV*, 2018.

[29] J.-T. Lin, D. Dai, and L. Van Gool, "Depth estimation from monocular images and sparse radar data," in *IROS*, 2020.

[30] H. Zhang, C. Shen, Y. Li, Y. Cao, Y. Liu, and Y. Yan, "Exploiting temporal consistency for real-time video depth estimation," in *ICCV*, 2019.

[31] A. C. S. Kumar, S. M. Bhandarkar, and M. Prasad, "Depthnet: A recurrent neural network architecture for monocular depth prediction," in *CVPRW*, 2018.

[32] K. Karsch, C. Liu, and S. B. Kang, "Depth transfer: Depth extraction from video using non-parametric sampling," *PAMI*, 2014.

[33] N. Yang, R. Wang, J. Stuckler, and D. Cremers, "Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry," in *ECCV*, 2018.

[34] X. Guo, H. Li, S. Yi, J. Ren, and X. Wang, "Learning monocular depth by distilling cross-domain stereo networks," in *ECCV*, 2018.

[35] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *NIPS*, 2015.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[37] N. Mayer, E. Ilg, P. Häusser, 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 *CVPR*, 2016.

[38] R. J. Williams and J. Peng, "An efficient gradient-based algorithm for on-line training of recurrent network trajectories," *Neural Computation*, pp. 490–501, 1990.

[39] 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 *CVPR*, 2019.

[40] C. Luo, Z. Yang, P. Wang, Y. Wang, W. Xu, R. Nevatia, and A. Yuille, "Every pixel counts++: Joint learning of geometry and motion with 3d holistic understanding," *arXiv preprint*, 2018.

[41] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova, "Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos," in *AAAI*, 2019.

[42] Y. Chen, C. Schmid, and C. Sminchisescu, "Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera," in *ICCV*, 2019.

[43] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *ICCV*, 2015.

[45] R. Józefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *ICML*, 2015.

[46] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint*, 2015.

[47] R. Wang, S. M. Pizer, and J.-M. Frahm, "Recurrent neural network for (un-) supervised learning of monocular video visual odometry and depth," in *CVPR*, 2019.