2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Temporally Consistent Horizon Lines

Florian Kluger[1], Hanno Ackermann[1], Michael Ying Yang[2], and Bodo Rosenhahn[1]

[1]Institut für Informationsverarbeitung, Leibniz Universität Hannover
{kluger,ackermann,rosenhahn}@tnt.uni-hannover.de
[2]Scene Understanding Group, University of Twente
michael.yang@utwente.nl

***Abstract*— The horizon line is an important geometric feature for many image processing and scene understanding tasks in computer vision. For instance, in navigation of autonomous vehicles or driver assistance, it can be used to improve 3D reconstruction as well as for semantic interpretation of dynamic environments. While both algorithms and datasets exist for single images, the problem of horizon line estimation from video sequences has not gained attention. In this paper, we show how convolutional neural networks are able to utilise the temporal consistency imposed by video sequences in order to increase the accuracy and reduce the variance of horizon line estimates. A novel CNN architecture with an improved residual convolutional LSTM is presented for temporally consistent horizon line estimation. We propose an adaptive loss function that ensures stable training as well as accurate results. Furthermore, we introduce an extension of the KITTI dataset which contains precise horizon line labels for 43699 images across 72 video sequences. A comprehensive evaluation shows that the proposed approach consistently achieves superior performance compared with existing methods.**

I. Introduction

Horizon lines are important low-level geometric image features that provide essential information about the relation between a 3D scene and the camera observing it. They can be used for a variety of applications including camera pose estimation [1], [2], vanishing point estimation [3], image metrology [4], and perspective correction [5]. These tasks in turn enable higher-level applications, for example inference of semantic properties of dynamic environments [6], [7] in constrained settings, such as autonomous driving, small drones, wearables or handheld devices.

For many applications, utilising temporal consistency has been demonstrated to improve performance. Examples include depth estimation [8], motion segmentation [9], action recognition [10], [11], super resolution [12], people tracking [13], human motion estimation [14] and superpixel segmentation [15]. Single image approaches for horizon line estimation may do gross mistakes when the image provides few or misleading clues. As illustrated by Fig. 1, an approach based on multiple images is less susceptible to these problems if it is able to transfer information from previous images of a sequence.

A. Contributions

In this work, we present a novel approach for temporally consistent horizon line estimation based on a convolutional neural network combined with an improved convolutional long short-term memory (LSTM). A comprehensive evaluation demonstrates the ability of this approach to generate more accurate horizon line estimates with less variance. Since a naïve loss function does not track the geometric error of horizon lines very well, and a loss based on the geometric error exhibits singularities that may cause instability, we propose an adaptive loss function that combines both losses with a cosine annealing schedule. This loss function yields significantly more accurate horizon estimates, yet ensures that the neural network training remains stable. In an ablation study, we investigate the influence of several hyperparameters and architecture choices on the performance of the neural network models. Furthermore, the *KITTI Horizon* dataset is presented, an extension of the well established KITTI benchmark [16]. It contains accurate horizon line annotations for all video sequences of the KITTI dataset [17].

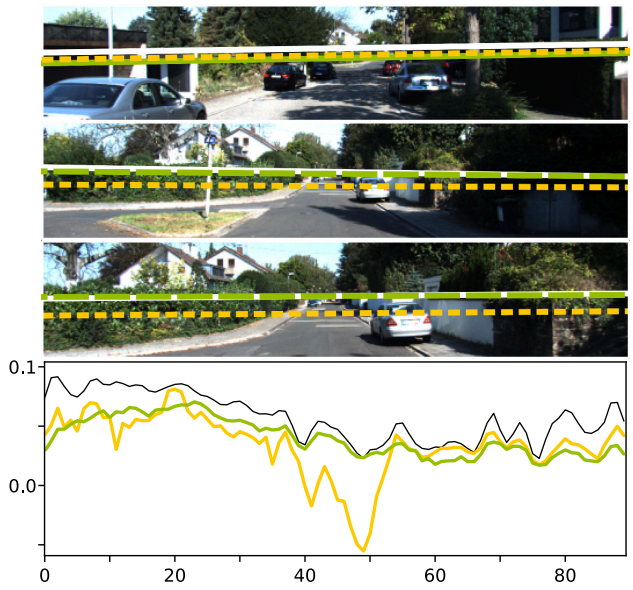


Fig. 1: Example sequence with our temporally consistent estimation in green (long dashes) and the best single frame algorithm in yellow (short dashes). Ground truth in white/black. Top rows: sample frames with horizons from the sequence. Bottom row: Horizon offset trajectory over time, best viewed in colour. The temporally consistent estimation is more accurate on average and contains fewer outliers.

978-1-7281-7395-5/20/$31.00 ©2020 IEEE

Fig. 2: Cropped images from a KITTI sequence with annotated horizon lines (left), and a sketch of the trajectory of the car with gravity vector $\mathbf{g}$ and plane normal $\mathbf{n}$ (right).

In summary, our **main contributions** are:

1) We present a novel CNN architecture for temporally consistent horizon line estimation based on an improved residual convolutional LSTM[1].
2) We propose an adaptive loss function that yields accurate horizon line estimates and ensures stable training.
3) A large-scale video dataset for temporally consistent horizon line estimation, the *KITTI Horizon* dataset[2]. To the best of our knowledge, this is the first video dataset with accurate horizon line ground truth.

B. Types of Horizon Lines

It is possible to distinguish three types of horizon lines: the *visible horizon*, the *true horizon* and the *geometrical horizon*. The visible horizon is the apparent line which separates earth and sky. Its appearance is often shaped by the surroundings of an observer in the presence of entities like mountains, buildings or trees. If the view of an observer is unobstructed – at sea, for example – the visible horizon becomes identical to the true horizon. Assuming a spherical earth surface, the true horizon is the projection of a circle containing all points on the earth which are tangent to light rays passing through the point of view of an observer.

The geometrical horizon $\mathbf{h}$ is defined as the vanishing line, i.e. the projection of the line at infinity, for any plane orthogonal to the local gravity vector $\mathbf{g}$:

$$\mathbf{h} \propto \mathbf{K}^{-\mathsf{T}}\mathbf{R}\mathbf{g} \quad , \tag{1}$$

with $\mathbf{R}$ being the orientation and $\mathbf{K}$ being the intrinsic calibration of the camera. Without loss of generality, we assume that $\mathbf{g} \propto \begin{bmatrix}0, & 1, & 0\end{bmatrix}^{\mathsf{T}}$ is parallel to the zenith direction. As illustrated by Fig. 2, the geometrical horizon is generally not identical to the vanishing line of the plane an observer is standing on, as its normal vector may not be parallel to $\mathbf{g}$, e.g. when located on an incline. Being a theoretical construction, the geometrical horizon is imperceptible to an observer. However, given the intrinsic calibration $\mathbf{K}$, knowledge of the geometrical horizon is sufficient to estimate camera tilt and roll w.r.t. a global coordinate system. Fig. 3 illustrates the conceptual differences between the three horizons. Since the remainder of this paper considers the geometrical horizon, it will be simply referred to as the horizon from hereon.

C. Related Work

In the past, numerous approaches for horizon line estimation have been proposed, and they can be differentiated into a number of categories. Most methods rely on vanishing points (VPs) [18], [19], [20], [21], [22], [23], [24], [25], [26] which they detect by grouping oriented elements like line segments into clusters which have the same orientation in 3D space. If at least two vanishing points are known, the horizon line can be derived. Some of these methods [20], [27], [23], [25] rely on the Manhattan-world assumption [28], i.e. they restrict their solution space to three VPs of orthogonal directions. Several of the aforementioned methods consider two benchmark datasets in their evaluation: the York Urban Dataset [29] (YUD) and the Eurasian Cities Dataset [18]. Both are relatively small and of limited diversity w.r.t. the types of scenes they depict. The Horizon Lines in the Wild (HLW) [30] dataset contains horizon line ground truth for 100553 images taken at various locations. Availability of such a large-scale dataset has lead to an emergence of deep-learning based algorithms [31], [30], [3] more recently. Workman et al. [30] present a convolutional neural network (CNN) which directly estimates the horizon line from a single image, formulated as either a regression or a classification task. Lee et al. [31] randomly sample lines within the image borders and feed them, along with the image, into a CNN which incorporates their proposed *line pooling layer*. This CNN then provides a classification whether the sampled line is the horizon of the image, and computes refined line coordinates. The method of Zhai et al. [3] is a hybrid approach. It uses a CNN, similar to [30], to predict a horizon line, but then jointly optimises its location together with VPs which are estimated based on line segments that have been detected in a preprocessing step. All these works have in common that they target the problem of *single image* horizon line estimation. To the best of our knowledge, general datasets and algorithms targeted specifically at horizon line estimation from *video sequences* do not exist.

II. KITTI Horizon Dataset

We introduce the *KITTI Horizon Dataset*, a new addition to the KITTI raw dataset [17] with accurate horizon line annotations for all video sequences.

1) Limitations of Existing Datasets: Three datasets have been commonly used for horizon line estimation in recent years: the York Urban Dataset [29] (YUD), the Eurasian Cities Dataset [18] (ECD) and Horizon Lines in the Wild [30] (HLW). YUD is a relatively small dataset of 102 images depicting in- and outdoor scenes within a confined area, taken with the same camera under similar conditions. While ECD is somewhat more diverse than YUD, it is still very small with just 103 images. HLW, on the other hand, is significantly larger and contains 100553 images, making it much better suited for data-intensive deep learning approaches. Beyond

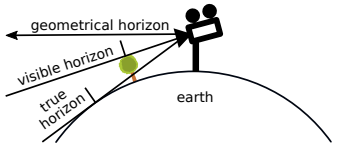


Fig. 3: Illustration of horizon line types (Sec. I-B).

[1]Source code: https://github.com/fkluger/tchl.
[2]Available at https://github.com/fkluger/kitti_horizon.

that, all three datasets have in common that they do not contain video sequences, which means that they can only be used for *single image* horizon line estimation and are ill-suited for *temporally consistent* horizon line estimation. To our knowledge, the Singapore Maritime Dataset [32] (SMD) is the only video dataset with annotated horizon lines. However, the horizon labels in SMD describe the *true horizon* as opposed to the *geometrical horizon*. Consequently, a new dataset is needed for temporally consistent geometrical horizon line estimation.

2) KITTI Horizon: KITTI [17] is a computer vision dataset which was captured using a sensor array mounted on top of a vehicle. Sensors used for the recordings include four front-facing video cameras and a high accuracy inertial measurement unit (IMU), among others. Several benchmarks for various applications, such as object detection, depth estimation or semantic segmentation, have been published [16]. For horizon line estimation such a benchmark does not exist. We can, however, compute accurate horizon line ground truth using the IMU data provided by KITTI, at no additional cost.

KITTI provides an accurate absolute pose $\mathbf{R}_{\text{IMU}}$ of the IMU in 3D space for every image. Together with the relative pose between IMU and camera $N \in \{1, 2, 3, 4\}$, $\mathbf{R}_{\text{IMU}\to N}$, we can compute the normalised gravity vector $\mathbf{g}_N \propto \mathbf{R}_{\text{IMU}\to N}\mathbf{R}_{\text{IMU}} \begin{bmatrix}0, & 1, & 0\end{bmatrix}^\mathsf{T}$ in the coordinate system of the camera. As explained in Sec. I-B, the projection of a gravity vector $\mathbf{g}$ into the camera using Eq. 1 yields the horizon line in homogeneous coordinates:

$$\mathbf{h}_N \propto \mathbf{K}_N^{-\mathsf{T}}\mathbf{R}_{\text{IMU}\to N}\mathbf{R}_{\text{IMU}} \begin{bmatrix}0, & 1, & 0\end{bmatrix}^\mathsf{T} . \tag{2}$$

As this process requires no manual labelling or other human intervention, we can compute the ground truth horizon for all images fully automatically. Fig. 1 shows the horizon offset trajectory of a KITTI sequence and three example images.

3) Train, validation and test split: The complete published KITTI dataset consists of 47962 frames across 157 sequences. Several sequences show the same stationary scene, and only differ w.r.t. the people walking across the image. As these are of negligible value for our task, we discarded all but one, so that 72 sequences with 43699 frames remain. As no official split exists for the raw dataset, we divided the video sequences into roughly 70% training, and 15% validation and test data each. Care was taken to ensure that sequences showing very similar scenes, e.g. the same intersection, do not end up in different parts of the split. As there is a strong imbalance in sequence length, we divided one of the longer videos equally and put it into the test and validation sets.

III. Single Image Estimation

We obtained the source code of recent single image algorithms [19], [21], [22], [30], [3]. In addition, we compare our own single image algorithm along with these methods. Our single image algorithm is based on a CNN, similar to the regression approach presented in [30]. We parametrise the horizon line $\mathbf{h}$ by offset ω and slope θ. With image width W, it is defined in homogeneous coordinates as:

$$\mathbf{h}(\omega, \theta) = \begin{bmatrix}\sin\theta, & \cos\theta, & -\frac{W}{2}\sin\theta - \omega\cos\theta\end{bmatrix}^\mathsf{T} . \tag{3}$$

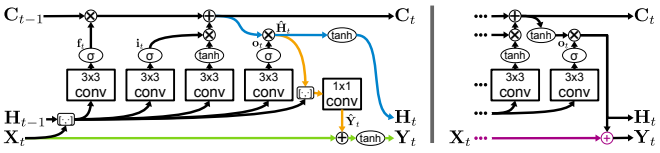


Fig. 4: ConvLSTM with residual paths as described in Sec. IV-A.1. The $[\cdot,\cdot]$-operator denotes concatenation along the channel axis. *Left*: Our proposed ConvLSTM with residual paths and dense connections. Changes w.r.t. a standard ConvLSTM: residual connection from $\mathbf{X}_t$ to $\mathbf{Y}_t$ in green; dense connection from $\mathbf{X}_t$, $\hat{\mathbf{H}}_t$ and $\mathbf{H}_{t-1}$ to $\mathbf{Y}_t$ in orange; reversal of operation order in blue. *Right*: Part of a standard ConvLSTM, with a naïve residual connection in purple.

We replace the GoogleNet [33] of [30] with the most shallow 18-layer variant of the more recent and efficient ResNet [34] (ResNet18). Its classification layer is replaced by fully connected layers with single real valued outputs for ω and θ.

IV. Temporally Consistent Estimation

Possibly the simplest way to utilise the temporal consistency of video sequences is applying a single-frame algorithm first, and then averaging the results. For online applications, a reasonable choice of filter would be the exponential moving average, or exponential smoothing filter [35]. Given a sequence x_t, the output of the filter is defined as:

$$s_t = \alpha x_t + (1 - \alpha)s_{t-1} \tag{4}$$

While easy to implement, it always just achieves a compromise between suppressing noise and outliers, and preserving actual trajectory changes. Bai et al. [36] propose temporal convolutional networks (TCN), an extension of regular CNNs by causal convolutions [37] along an additional temporal dimension. Across time, the TCN has a fixed field of view which limits the sequence length along which it is able to infer correlations. We therefore chose to investigate an approach based on long-short term memory (LSTM) [38]. We devised a novel approach combining the ResNet [34] architecture with an improved convolutional LSTM layer.

A. Convolutional LSTM

LSTM cells are a particular type of recurrent neural network (RNN) that have been proven effective in modelling both long- and short-term dependencies of sequential data [39], [40], [41]. The convolutional LSTM (ConvLSTM) [8], [42] is a variant that operates on 3D tensors instead of vectors and replaces all matrix multiplications with kernel convolutions. Given a sequence of inputs $\mathbf{X}_1, \ldots, \mathbf{X}_t$, the cell state $\mathbf{C}_t$ and hidden state $\mathbf{H}_t$ of a ConvLSTM can be computed as follows, where '$*$' is the convolution operator

and '$\odot$' denotes the Hadamard product:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{xi} * \mathbf{X}_t + \mathbf{W}_{hi} * \mathbf{H}_{t-1} + \mathbf{b}_i) \quad (5)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{xf} * \mathbf{X}_t + \mathbf{W}_{hf} * \mathbf{H}_{t-1} + \mathbf{b}_f) \quad (6)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{xo} * \mathbf{X}_t + \mathbf{W}_{ho} * \mathbf{H}_{t-1} + \mathbf{b}_o) \quad (7)$$

$$\mathbf{C}_t = \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_{xc} * \mathbf{X}_t + \mathbf{W}_{hc} * \mathbf{H}_{t-1} + \mathbf{b}_c) \quad (8)$$

$$\mathbf{H}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t) \quad (9)$$

The hidden state is usually treated as the output of the cell, i.e. $\mathbf{Y}_t = \mathbf{H}_t$. Variants with additional connections [43], or other activation functions [8] exist as well.

1) Residual Convolutional LSTM: We propose an improved convolutional LSTM structure that incorporates both residual and dense connections. As previous works [34], [41] have shown, residual connections improve gradient flow in deep neural networks, which makes them easier and faster to train. He et al. [34] integrated residual connections into a CNN. If we consider a shallow stack l of convolutional layers performing an operation $F_l(\mathbf{x})$ on an input $\mathbf{x}_{l-1}$, the output $\mathbf{x}_l$ of such a stack is: $\mathbf{x}_l = g(F_l(\mathbf{x}_{l-1}) + \mathbf{x}_{l-1})$, with $g(\cdot)$ being a nonlinear activation function, e.g. ReLU. In [41], this idea was applied to a network of stacked LSTM cells. Each LSTM cell computes a hidden state $\mathbf{h}_t$ and a cell state $\mathbf{c}_t$ based on an input $\mathbf{x}_t$ and the states at the previous time step: $\mathbf{h}_t, \mathbf{c}_t = \mathrm{LSTM}(\mathbf{h}_{t-1}, \mathbf{c}_{t-1}, \mathbf{x}_t)$. A residual connection is then applied to generate the final output of the layer:

$$\mathbf{y}_t = \mathbf{h}_t + \mathbf{x}_t \,. \quad (10)$$

In this case, the non-linearity $g(\cdot)$ is part of the LSTM, i.e. it is applied before the residual connection. The notion of improving information flow through a neural network via connections that skip a number of layers was implemented in yet another manner by Huang et al. [44]. In their DenseNet CNN architecture, feature-maps of M preceding layers $\mathbf{x}_{l-M}, \ldots \mathbf{x}_{l-1}$ are concatenated channel-wise and fed into the current layer $F_l(\mathbf{x})$: $\mathbf{x}_l = g(F_l([\mathbf{x}_{l-M}, \ldots, \mathbf{x}_{l-1}]))$. In order to arrive at our improved ConvLSTM, we combine the aforementioned principles and incorporate them as follows. Fig. 4 illustrates our proposed structure on the left side, while the right side shows the standard ConvLSTM with a naïve residual connection as per Eq. 10 for comparison. In keeping with the original ResNet definition, we define a residual connection between input and output:

$$\mathbf{Y}_t = \tanh(\hat{\mathbf{Y}}_t + \mathbf{X}_t) \,. \quad (11)$$

As Eq. 9 shows, the hidden state $\mathbf{H}_t$ amounts to a masked cell state $\mathbf{C}_t$. We argue that this inhibits the flow of information from both $\mathbf{X}_t$ and $\mathbf{H}_{t-1}$ to the output $\mathbf{Y}_t$. Normally, information must pass through Eqs. 5-9 and thus through $\mathbf{C}_t$ before it eventually reaches $\mathbf{Y}_t$. We therefore introduce an additional convolutional layer into the ConvLSTM, which directly takes the concatenation of $\mathbf{X}_t$, $\mathbf{H}_{t-1}$ and an intermediate hidden state $\hat{\mathbf{H}}_t$ as an input, similar to the way convolution layers in DenseNet operate, in order to produce

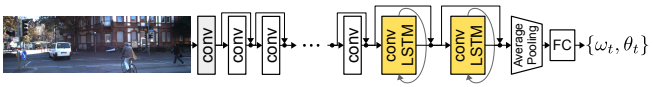


Fig. 5: Proposed neural network structure employing ConvLSTM layers as described in Sec. IV-B. Two ConvLSTM layers are inserted between the last convolutional layer and the global average pooling layer of our ResNet18-based CNN. The outputs ω and θ are the offset and slope, respectively.

an intermediate output $\hat{\mathbf{Y}}_t$:

$$\hat{\mathbf{Y}}_t = \mathbf{W}_{xy} * \mathbf{X}_t + \mathbf{W}_{hy} * \mathbf{H}_{t-1} + \mathbf{W}_{\hat{h}y} * \hat{\mathbf{H}}_t \,. \quad (12)$$

Finally, in order to avoid application of the tanh activation twice onto the information from $\mathbf{C}_t$, we switch the order of operation in Eq. 9, i.e. :

$$\hat{\mathbf{H}}_t = \mathbf{o}_t \odot \mathbf{C}_t \,, \quad (13)$$

$$\mathbf{H}_t = \tanh(\hat{\mathbf{H}}_t) \,. \quad (14)$$

B. Horizon Line Estimation Network

We expand our single image CNN described in Sec. III with our modified ConvLSTM presented in Sec. IV-A.1 in order to create a temporally consistent architecture. As Fig. 5 shows, two ConvLSTM layers are inserted between the last convolutional layer and the global average pooling layer of our CNN. Intuitively, applying the ConvLSTM at this stage makes most sense, as we would expect it to find temporal correlations between higher-level features which are most pertinent to the task of horizon estimation.

C. Loss Function

Our CNN has two real valued outputs: offset ω and slope θ of the predicted horizon line. We compute two loss terms; the first one is the Huber loss of ω and θ computed w.r.t. the ground truth; the second one is the maximum horizon error within the image. Combining these two losses allows us to benefit from a gain in accuracy elicited by minimising the maximum horizon error, while avoiding the instability it can cause. The Huber loss [45] is defined as:

$$L_H(x, \hat{x}) = \begin{cases} \frac{1}{2}(x - \hat{x})^2 & \text{for } |x - \hat{x}| \leq 1 \,, \\ |x - \hat{x}| - \frac{1}{2} & \text{otherwise.} \end{cases}$$

We define the first loss term as the Huber loss of ω and θ computed w.r.t. the ground truth $\hat{\omega}$ and $\hat{\theta}$:

$$L_{\omega,\theta} = L_H(\omega, \hat{\omega}) + L_H(\theta, \hat{\theta}) \,. \quad (15)$$

As this loss term does not exactly track the maximum horizon error, which is the quantity we actually seek to minimise, we have defined a second loss term. The horizon error is defined as the maximum distance between the estimated horizon $\mathbf{h}(\omega, \theta)$ (Eq. 3), and the ground truth $\mathbf{h}(\hat{\omega}, \hat{\theta})$ between the left- and rightmost borders of the image, normalised to image height H. The y-coordinate of the intersection of $\mathbf{h}$ with a vertical line at x is defined by:

$$y(\omega, \theta, x) = \left(x - \frac{W}{2}\right) \tan\theta - \omega \,. \quad (16)$$

Let $d_{y,0}$ and $d_{y,W}$ be the left- and right-most distances between the two horizons, $d_{y,x} = \left| y(\omega, \theta, x) - y(\hat{\omega}, \hat{\theta}, x) \right|$. The maximum horizon error L_e can then be defined as:

$$L_e = \begin{cases} \frac{1}{H} d_{y,0} & \text{for } d_{y,0} \geq d_{y,W} \,, \\ \frac{1}{H} d_{y,W} & \text{otherwise.} \end{cases} \tag{17}$$

While L_e directly reflects the quantity we aim to minimise, it contains singularities for $\theta = (\pi/2 + n\pi), n \in \mathbb{N}$. This causes L_e to become excessively large if θ is poorly estimated, which may be the case especially at the beginning of neural network training. We therefore use only $L_{\omega,\theta}$ at first, when estimates are still very inaccurate and noisy, and gradually switch over to L_e on a cosine schedule similar to [46]. With t being the current epoch and T being the maximum number of epochs, the schedule is defined by: $\lambda(t) = \frac{1}{2} + \frac{1}{2} \cos\left(\pi \cdot \frac{t}{T}\right)$. Using this, the final loss L is defined as:

$$L(t) = \lambda(t) \cdot L_{\omega,\theta} + (1 - \lambda(t)) \cdot L_e \,. \tag{18}$$

V. Experiments

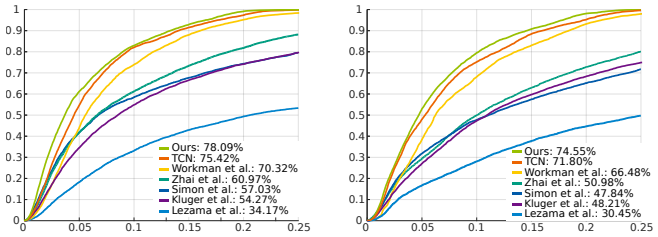
We empirically demonstrate the effectiveness of our temporally consistent horizon line estimation pipeline on the KITTI Horizon validation and test sets and compare it with state-of-the-art single-image algorithms and other temporally consistent baselines. Additional ablation studies show the importance of individual parts of this pipeline.

A. Implementation Details

We implemented the proposed neural network architectures using PyTorch [47]. On KITTI Horizon, all networks were trained for 160 epochs with stochastic gradient descent using a cosine annealing learning rate schedule [46] between 10^{-1} and 10^{-3}. Training was repeated ten times with different random seeds, and the model with the highest validation AUC chosen. We downscale each image by a factor of two and apply cutout [48], colour jitter, and random rotations and shifts for data augmentation. We initialise the weights of the first nine convolutional layers from a ResNet18 pretrained on ImageNet [49] while other layers are initialised randomly. Training batches always contain B sequences of S consecutive frames from the KITTI Horizon training set, with $S \cdot B = 128$.

B. Evaluation Metrics

As in [19], [21], [22], [30], [3], we compute the maximum horizon error defined in Eq. 17 for every image in the dataset. A cumulative error histogram for errors up to 0.25 is generated and its area under the curve (AUC) determined for a set of images. This horizon error AUC value gauges the overall accuracy of the estimated horizon lines. We also report the mean squared error (MSE), which is more sensitive to outliers than the AUC. For applications that rely on horizon lines estimated from a video stream, it is desirable for the estimations be accurate as well as stable. We propose another metric to measure undesirable fluctuations that do not reflect actual changes of the horizon over time: the average total variation A_{TV}. For a sequence n of length T_n of



(a) validation set (b) test set

Fig. 6: Cumulative horizon error histograms with AUC values for KITTI.

estimated horizons $\mathbf{h}_{n,t}$ and corresponding ground truth $\hat{\mathbf{h}}_{n,t}$, with $t \in [1, T_n]$ and $n \in [1, N]$, we compute the derivative $\partial L_e^{n,t} / \partial t$ of the horizon error according to Eq. 17 using second order approximation. With $M = \sum_{n=1}^{N} T_n$ being the total number of images, the mean of its absolute calculated over all sequences yields the average total variation:

$$A_{TV} = \frac{1}{M} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \left| \frac{\partial L_e^{n,t}}{\partial t} \right| \,. \tag{19}$$

This metric is invariant to constant deviations from the ground truth but sensitive to higher frequency fluctuations.

C. KITTI Horizon Results

We report all metrics on the KITTI Horizon validation and test set for following single frame algorithms: the VP based methods of Lezama et al. [21], Kluger et al. [19] and Simon et al. [22], the hybrid approach of Zhai et al. [3] and the CNN based approach of Workman et al. [30]. We also include results for our single frame CNN baseline (cf. Sec. III), trained on either HLW or KITTI, for an *average* baseline which simply always predicts the mean of the training set, a TCN [36] based temporally consistent approach with causal convolutions in the last three layers, and of course for our temporally consistent pipeline presented in Sec. IV. The results are listed in Tab. Ia and Fig. 6. As these numbers show, methods based on line segments and vanishing points [19], [21], [22], [3] are unable to deliver consistent and accurate horizon estimates on KITTI. The best performing method among them is Zhai et al. [3] with 60.97%/50.98% AUC (validation/test), which still lags behind the simplest *average* baseline (69.40%/64.18%). In addition, the very large mean squared error (MSE) and average total variation (A_{TV}) values – up to several thousand – indicate that these methods may fail catastrophically in some outlier cases. In comparison, all CNN based methods – including Workman et al. [30] and our own single frame CNN – are significantly more accurate with at least 70.32%/63.64% AUC. More importantly, the comparatively smaller MSE and A_{TV} show that these methods are much less prone to extreme outliers. If we compare the CNN of [30] with our own single-frame CNN trained on HLW, we observe that [30] performs better overall – all metrics but validation AUC are better to a relevant degree. This is unsurprising, as [30] augmented their training with an additional 500000 images

		AUC		MSE $\times 10^{-3}$		$A_{TV} \times 10^{-3}$	
		val	test	val	test	val	test
Lezama et al. [21]		34.17%	30.45%	>1000	>1000	2397	1537
Kluger et al. [19]		54.27%	48.21%	>1000	>1000	188.6	206.4
Simon et al. [22]		57.03%	47.84%	84.26	224.0	65.94	88.71
Zhai et al. [3]		60.97%	50.98%	>1000	>1000	91.56	1575
Workman et al. [30]		70.32%	66.48%	9.208	11.19	6.893	8.430
Average baseline		69.40%	64.18%	8.800	12.20	6.091	5.123
single frame (Sec. III)	trained on HLW	71.10%	63.64%	10.41	14.31	13.90	15.71
	trained on KITTI-H	77.42%	74.08%	6.024	7.025	5.061	5.585
	exp. smoothing	77.44%	74.11%	5.986	6.987	4.337	4.687
TCN [36]	(3-3-5)	75.42%	71.80%	6.392	8.318	4.945	**4.937**
temporally consistent (Sec. IV)		**78.09%**	**74.55%**	**5.427**	**6.731**	**4.619**	4.984
	exp. smoothing	**78.11%**	**74.68%**	**5.405**	**6.712**	**4.159**	**4.404**

(a)

	AUC	MSE $\times 10^{-3}$	A_{TV} $\times 10^{-3}$
Huber loss (Sec. V-D.1)	71.96%	7.851	7.051
non-temporal (Sec. V-D.2)	74.36%	7.266	5.699
w/o residual (Sec. V-D.3)	64.29%	11.60	5.279
naïve residual (Sec. V-D.3)	74.01%	7.009	**4.967**
Ours (Sec. IV)	**74.55%**	**6.731**	4.984

(b)

TABLE I: (a) Horizon estimation results on the KITTI Horizon (Sec. II) validation and test sets using the metrics described in Sec. V-B. AUC: higher is better; MSE and A_{TV}: lower is better. Refer to Sec. V-C for a detailed discussion. (b) Ablation study (Sec. V-D) results on the KITTI Horizon test set.

sampled from Google Street View, while we just used HLW. Naturally, if trained on the KITTI Horizon dataset, the accuracy of our single frame CNN increases significantly: from 71.10%/63.64% to 77.42%/74.08%, which is a 21.8%/28.7% relative increase. Best results on all metrics are obtained with our temporally consistent approach (Sec. IV), with relative improvements upon the single frame CNN between 1.8% (test AUC) and 12.1% (test A_{TV}). While the smoothness A_{TV} of the single frame CNN improves measurably without diminishing overall accuracy if we additionally apply an exponential smoothing filter (Eq. 4, $\alpha = 0.5$), similar gains can be achieved when applied to the temporally consistent CNN as well. We also trained a TCN [36] based on our single frame CNN with causal temporal convolutions of widths 3, 3, and 5 in the last three layers and a receptive field of nine frames. Surprisingly, it performs worse than the single frame CNN on all metrics but A_{TV}. We suspect that the TCN is more susceptible to overfitting, as it achieved a lower training loss, but higher validation loss compared to our other CNNs. Compared to our ConvLSTM based network, it is on par w.r.t. A_{TV} on the test set, but measurably worse otherwise.

D. Ablation Studies

1) Loss function: In order to investigate whether our new loss defined in Sec. IV-C had the desired effect on estimation accuracy, we also trained our main CNN model described in Sec. IV-B using just the Huber loss defined in Eq. 15 and also used by [30]. As Tab. Ib shows, we report an AUC of 71.96% and an MSE of $7.851 \cdot 10^{-3}$ on the test set. Using our newly defined loss, however, we achieve an AUC of 74.55% and an MSE of $6.731 \cdot 10^{-3}$, which marks a considerable relative improvement of 9.2% and 14.3% respectively.

2) Temporal information: As Tab. Ia shows, our temporally consistent approach based on ConvLSTMs is able to achieve more accurate horizon estimates with less variance. In order to verify that this is due to the ConvLSTM utilising temporal correlations, and not simply due other architecture changes that arose as a result, we retrained our main CNN model with temporal connections disabled, i.e. we reset the LSTM states at every time step. On the test set, this yields an AUC of 74.36% and A_{TV} of $5.699 \cdot 10^{-3}$. When we enable the temporal connections of the LSTM, overall accuracy increases moderately but A_{TV} decreases noticeably, to $4.984 \cdot 10^{-3}$, which is a relative improvement of 12.6%. We conclude that the ConvLSTM is indeed able to retain temporal consistency in a meaningful way.

3) ConvLSTM Architecture: We compare our ConvLSTM architecture described in Sec IV-A.1 against a ConvLSTM using a naïve residual path implementation and a ConvLSTM without the residual path. As Tab. Ib shows, the naïve residual path already increases accuracy dramatically, from 64.29% to 74.01% AUC, and is evidently crucial for deep LSTM networks. On par w.r.t. A_{TV}, our proposed ConvLSTM improves AUC and MSE upon the naïve implementation, yielding a relative improvement of 2.1% and 4.0% respectively. While both approaches are able to generate smooth trajectories, our improved ConvLSTM is measurably more accurate on average.

VI. Conclusion

The horizon line is an important geometric feature which can be used in many computer vision tasks, such as camera pose and ground plane estimation. Due to their importance, horizon lines have received considerable attention in recent years. Nonetheless, neither has any other work focused on temporal consistency, nor are there appropriate datasets available. In this work, an extension of the well-known KITTI dataset is presented that adds horizon line annotations to 72 sequences. We furthermore propose a neural network for temporally consistent horizon line estimation in video sequences. It utilises an improved convolutional LSTM and an adaptive loss function that yields more accurate horizon line estimates and ensures stable training. The experimental evaluation demonstrates that the proposed architecture achieves superior performance for a diverse set of metrics which measure accuracy and smoothness of trajectories.

Acknowledgement: This work was supported by German Research Foundation (DFG) grant Ro 2497 / 12-2.

REFERENCES

[1] S. M. Ettinger, M. C. Nechyba, P. G. Ifju, and M. Waszak, "Vision-guided flight stability and control for micro air vehicles," *Advanced Robotics*, vol. 17, no. 7, pp. 617–640, 2003.

[2] Y. Hold-Geoffroy, K. Sunkavalli, J. Eisenmann, M. Fisher, E. Gambaretto, S. Hadap, and J.-F. Lalonde, "A perceptual measure for deep single image camera calibration," in *CVPR*, 2018.

[3] M. Zhai, S. Workman, and N. Jacobs, "Detecting vanishing points using global image context in a non-manhattan world," in *CVPR*, 2016.

[4] A. Criminisi, I. Reid, and A. Zisserman, "Single view metrology," *IJCV*, vol. 40, no. 2, pp. 123–148, 2000.

[5] H. Lee, E. Shechtman, J. Wang, and S. Lee, "Automatic upright adjustment of photographs," in *CVPR*, 2012.

[6] J. M. Alvarez, T. Gevers, and A. M. Lopez, "3d scene priors for road detection," in *CVPR*, 2010.

[7] A. Geiger, "Monocular road mosaicing for urban environments," in *IV*, 2009.

[8] D. Tananaev, H. Zhou, B. Ummenhofer, and T. Brox, "Temporally consistent depth estimation in videos with recurrent architectures," in *ECCV*, 2018.

[9] P. Bertholet, A.-E. Ichim, and M. Zwicker, "Temporally consistent motion segmentation from rgb-d video," in *Computer Graphics Forum*, vol. 37, no. 6, 2018, pp. 118–134.

[10] A. Hanson, P. Koutilya, S. Krishnagopal, and L. Davis, "Bidirectional convolutional lstm for the detection of violence in videos," in *ECCV*, 2018.

[11] M. Y. Yang, W. Liao, Y. Cao, and B. Rosenhahn, "Video event recognition and anomaly detection by combining gaussian process and hierarchical dirichlet process models," *Photogrammetric Engineering & Remote Sensing*, 2018.

[12] Y. Huang, W. Wang, and L. Wang, "Video super-resolution via bidirectional recurrent convolutional networks," *TPAMI*, vol. 40, no. 4, pp. 1015–1028, 2018.

[13] R. Henschel, Y. Zou, and B. Rosenhahn, "Multiple people tracking using body and joint detections," in *CVPRW*, 2019.

[14] B. Wandt, H. Ackermann, and B. Rosenhahn, "3d reconstruction of human motion from monocular image sequences," *TPAMI*, 2016.

[15] M. Reso, J. Jachalsky, B. Rosenhahn, and J. Ostermann, "Temporally consistent superpixels," in *ICCV*, 2013.

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

[17] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *IJRR*, 2013.

[18] O. Barinova, V. Lempitsky, E. Tretiak, and P. Kohli, "Geometric image parsing in man-made environments," in *ECCV*, 2010.

[19] F. Kluger, H. Ackermann, M. Y. Yang, and B. Rosenhahn, "Deep learning for vanishing point detection using an inverse gnomonic projection," in *GCPR*, 2017.

[20] J. Košecká and W. Zhang, "Video compass," in *ECCV*, 2002.

[21] J. Lezama, R. Grompone von Gioi, G. Randall, and J.-M. Morel, "Finding vanishing points via point alignments in image primal and dual domains," in *CVPR*, 2014.

[22] G. Simon, A. Fond, and M.-O. Berger, "A-contrario horizon-first vanishing point detection using second-order grouping laws," in *ECCV*, 2018.

[23] J.-P. Tardif, "Non-iterative approach for fast and accurate vanishing point detection," in *ICCV*, 2009.

[24] A. Vedaldi and A. Zisserman, "Self-similar sketch," in *ECCV*, 2012.

[25] H. Wildenauer and A. Hanbury, "Robust camera self-calibration from monocular images of manhattan worlds," in *CVPR*, 2012.

[26] Y. Xu, S. Oh, and A. Hoogs, "A minimum error vanishing point detection approach for uncalibrated monocular images of man-made environments," in *CVPR*, 2013.

[27] C. Rother, "A new approach to vanishing point detection in architectural environments," *Image and Vision Computing*, vol. 20, no. 9, pp. 647–655, 2002.

[28] J. M. Coughlan and A. L. Yuille, "Manhattan world: Compass direction from a single image by bayesian inference," in *ICCV*, 1999.

[29] P. Denis, J. H. Elder, and F. J. Estrada, "Efficient edge-based methods for estimating manhattan frames in urban imagery," in *ECCV*, 2008.

[30] S. Workman, M. Zhai, and N. Jacobs, "Horizon lines in the wild," in *BMVC*, 2016.

[31] J.-T. Lee, H.-U. Kim, C. Lee, and C.-S. Kim, "Semantic line detection and its applications," in *ICCV*, 2017.

[32] D. K. Prasad, D. Rajan, L. Rachmawati, E. Rajabally, and C. Quek, "Video processing from electro-optical sensors for object detection and tracking in a maritime environment: a survey," *T-ITS*, vol. 18, no. 8, pp. 1993–2016, 2017.

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.

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

[35] R. G. Brown, *Smoothing, forecasting and prediction of discrete time series*, 2004.

[36] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv:1803.01271*, 2018.

[37] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv:1609.03499*, 2016.

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[39] A. Graves, N. Jaitly, and A.-r. Mohamed, "Hybrid speech recognition with deep bidirectional lstm," in *ASRU*, 2013.

[40] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *INTERSPEECH*, 2014.

[41] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv:1609.08144*, 2016.

[42] 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.

[43] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," in *IJCNN*, vol. 3, 2000, pp. 189–194.

[44] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *CVPR*, 2017.

[45] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in statistics*, 1992, pp. 492–518.

[46] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv:1608.03983*, 2016.

[47] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS Autodiff Workshop*, 2017.

[48] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv:1708.04552*, 2017.

[49] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR*, 2009.