Fast View Synthesis with Deep Stereo Vision

Tewodros Habtegebrial[1], Kiran Varanasi[2], Christian Bailer[2] and Didier Stricker[1,2]
[1]*Department of Computer Science, TU Kaiserslautern, Erwin-Schrdinger-Str. 1, Kaiserslautern, Germany*
[2]*German Research Center for Artificial Intelligence, Trippstadter Str. 122, Kaiserslautern, Germany*

Keywords: Novel View Synthesis, Convolutional Neural Networks, Stereo Vision.

Abstract: Novel view synthesis is an important problem in computer vision and graphics. Over the years a large number of solutions have been put forward to solve the problem. However, the large-baseline novel view synthesis problem is far from being "solved". Recent works have attempted to use Convolutional Neural Networks (CNNs) to solve view synthesis tasks. Due to the difficulty of learning scene geometry and interpreting camera motion, CNNs are often unable to generate realistic novel views. In this paper, we present a novel view synthesis approach based on stereo-vision and CNNs that decomposes the problem into two sub-tasks: *view dependent geometry* estimation and *texture inpainting*. Both tasks are structured prediction problems that could be effectively learned with CNNs. Experiments on the KITTI Odometry dataset show that our approach is more accurate and significantly faster than the current state-of-the-art.

1 INTRODUCTION

Novel view synthesis (NVS) is defined as the problem of rendering a scene from a previously unseen camera viewpoint, given other reference images of the same scene. This is an inherently ambiguous problem due to perspective projection, occlusions in the scene, and the effects of lighting and shadows that vary with a viewpoint. Due to this inherent ambiguity, this can be solved only by learning valid scene priors and hence, is an effective problem to showcase the application of machine learning to computer vision.

In the early 1990s, methods for NVS were proposed to deal with slight viewpoint changes, given images taken from relatively close viewpoints. Then NVS can be performed through view interpolation (Chen and Williams, 1993), warping (Seitz and Dyer, 1995) or rendering with stereo reconstruction (Scharstein, 1996). There are a few methods proposed over the years to solve the problem of large-baseline NVS (Chaurasia et al., 2013), (Zitnick et al., 2004), (Flynn et al., 2016), (Goesele et al., 2010), (Penner and Zhang, 2017). Some methods are based on structure from motion (*SFM*) (Zitnick et al., 2004), which can produce high-quality novel views in real time (Chaurasia et al., 2013), but has limitations when the input images contain strong noise, illumination change and highly non-planar structures like vegetation. These methods need to preform depth synthesis for poorly reconstructed areas in *SFM*, which is challenging for intricate structures. In contrast to these methods, neural networks can be trained end-to-end to render NVS directly (Flynn et al., 2016). This is the paradigm we follow in this paper.

Many recent works have addressed end-to-end training of neural networks for NVS (Dosovitskiy et al., 2015), (Tatarchenko et al., 2016), (Zhou et al., 2016), (Yang et al., 2015). These methods typically perform well under a restricted scenario, where they have to render geometrically simple scenes. The state-of-the-art large-baseline view synthesis approach that works well under challenging scenarios is *DeepStereo*, introduced by Flynn et. al. (Flynn et al., 2016). *DeepStereo* generates high-quality novel views on the KITTI dataset (Geiger et al., 2012), where older *SFM* based methods such as (Chaurasia et al., 2013) do not work at all. This algorithm uses plane-sweep volumes and processes them with a double tower CNN (with *color* and *selection* towers). However, processing plane-sweep volumes of all reference views jointly imposes very high memory and computational costs. Thus, *DeepStereo* is far slower than previous *SFM* based methods such as (Chaurasia et al., 2013).

In this work we propose a novel alternative to *DeepStereo* which is two orders of magnitude faster. Our method avoids performing expensive calculations on the combination of plane-sweep volu-

Habtegebrial, T., Varanasi, K., Bailer, C. and Stricker, D.
Fast View Synthesis with Deep Stereo Vision.
DOI: 10.5220/0007360107920799
In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2019)*, pages 792-799
ISBN: 978-989-758-354-4

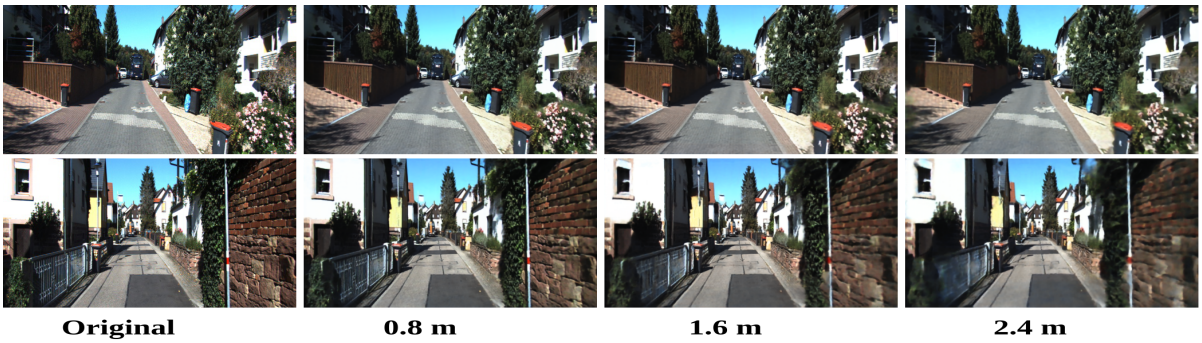


Figure 1: Two sample renderings from the KITTI dataset (Geiger et al., 2012), using our proposed method. Images are rendered using four neighbouring views. From left to right, the median baseline between consecutive input views increases from 0.8m to 2.4m.

mes of reference images. Instead, we predict a proxy scene geometry for the input views with stereo-vision. Using forward-mapping (section 3.3), we project input views to the novel view. Forward-mapped images contain a large number of pixels with unknown color values. Rendering of the target view is done by applying texture inpainting on the warped-images. Our rendering pipeline is fully-learnable from a sequence of calibrated stereo images. Compared to *DeepStereo*, the proposed approach produces more accurate results while being significantly faster. The main contributions of this paper are the following.

- We present a novel view synthesis approach based on stereo-vision. The proposed approach decomposes the problem into *proxy* geometry prediction and texture inpainting tasks. Every part of our method is fully learnable from input stereo reference images.
- Our approach provides an affordable large-baseline view synthesis solution. The proposed method is faster and even more accurate than the current *state-of-the-art* method (Flynn et al., 2016) . The proposed approach takes seconds to render a single frame while DeepStereo (Flynn et al., 2016) takes minutes.

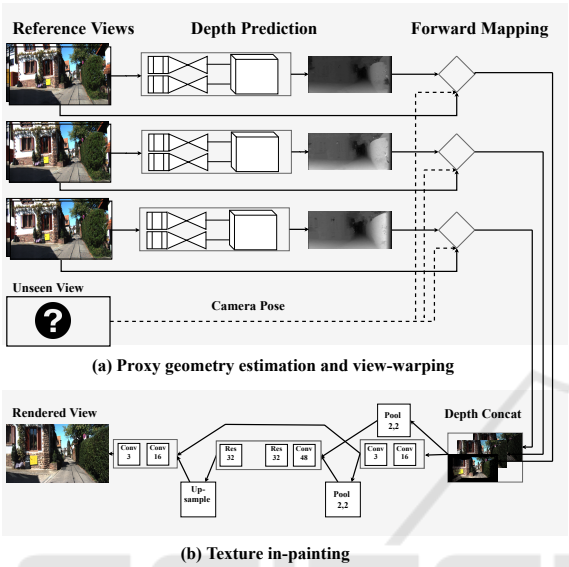
2 RELATED WORK

Image based rendering has enjoyed a significant amount of attention from the computer vision and graphics communities. Over the last few decades, several approaches of image-based rendering and modelling were introduced (Chen and Williams, 1993), (Seitz and Dyer, 1995), (Seitz and Dyer, 1996), (Adelson et al., 1991), (Scharstein, 1996). Fitzgibbon et. al. (Fitzgibbon et al., 2005) present a solution that solves view synthesis as texture synthesis by using image-based priors as regularization. Chaurasia et. al. (Chaurasia et al., 2013), presented high-quality view synthesis that utilizes 3D reconstruction. Recently, Penner at. al. (Penner and Zhang, 2017) presented a view synthesis method that uses soft 3D reconstruction via fast local stereo-matching similar to (Hosni et al., 2011) and occlusion aware depth-synthesis. Kalantari et. al. (Kalantari et al., 2016) used deep convolutional networks for view synthesis in light-fields.

Encoder-decoder Networks have been used in generating unseen views of objects with simple geometric structure, (e.g. cars, chairs, etc) (Tatarchenko et al., 2016), (Dosovitskiy et al., 2015). However, the renderings generated from encoder-decoder architectures are often blurry. Zhou et al. (Zhou et al., 2016), used encoder-decoder networks to predict appearance flow, rather than directly generating the image. Compared to direct novel view generating methods (Tatarchenko et al., 2016), (Dosovitskiy et al., 2015), the appearance-flow based method produces crispier results. Nonetheless, the appearance flow based also fails to produce any convincing results in natural scenes.

Flynn et. al. (Flynn et al., 2016), proposed the first CNN based large baseline novel view synthesis approach. DeepStereo has a double-tower(*color and selection towers*) CNN architecture. DeepStereo takes a volume of images projected using multiple depth planes, known as *plane-sweep volume* as input. The *color-tower* produces renderings for every depth plane separately. The *selection-tower* estimates probabilities for the renderings computed for every depth plane. The output image is then computed as a weighted average of the rendered color-images. DeepStereo generates high-quality novel views from a plane sweep volume generated from few(typically 4) reference views. To the best of our knowledge, *DeepStereo* is the most accurate large-baseline method,

proven to be able to generate accurate novel views of challenging natural scenes. Recently, monocular depth prediction based view synthesis methods have been proposed (Liu et al., 2018), (Yin et al., 2018). Despite being fast, these works produce results with significantly lower quality compared to multiview approaches such as *DeepStereo* (Flynn et al., 2016).



(a) Sample view-warping

Figure 2: Illustration of our novel view synthesis approach. in the first stage(a), we begin by estimating dense depth maps from the input reference stereo pairs using an unsupervised stereo-depth prediction network. Estimated depth maps are used to project input views to the target view via Forward-mapping, shown in Equation 7. As shown in (b), the output novel view is rendered with the *texture inpainting* applied on the forward mapped views.

3 PROPOSED METHOD

In this section we discuss our proposed view synthesis approach. Our aim is to generate the image $\mathcal{X}^t$ of a scene from a novel viewpoint, using a set of reference stereo-pairs $\{\{\mathcal{X}_L^1,\mathcal{X}_R^1\},\{\mathcal{X}_L^2,\mathcal{X}_R^2\},\ldots,\{\mathcal{X}_L^V,\mathcal{X}_R^V\}\}$[1] and their poses $\{\mathcal{P}^1,\mathcal{P}^2,\ldots,\mathcal{P}^V\}$ w.r.t the target view. The proposed method has three main stages, namely *proxy scene geometry estimation*, *forward-mapping* and *texture inpainting*. As shown in Figure 2 the view synthesis is performed as follows: first, a proxy scene geometry is estimated as a dense depth map. The estimated depth map is used to forward-map the input views to the desired novel viewpoint. Forward mapping (described in section 3.3) leads to noisy images with a large number of holes. The final rendering is, therefore, generated by applying *texture inpainting* on the forward-mapped images. Both the depth estimation and texture inpainting tasks are learned via convolutional networks. Components of our view synthesis pipeline are trainable using a dataset that contains a sequence of stereo-pairs with known camera poses.

3.1 Depth Prediction Network

We train a convolutional network to estimate depth from an input stereo pair. The training set for the depth prediction CNN is generated by sampling M stereo pairs from our training set. The proposed network is composed of the following stages: *feature extraction*, *feature-volume aggregation* and *feature-volume filtering*. Architecture-wise our network is similar to GCNet (Kendall et al., 2017). However, there are several key differences, including the fact that our network is trained in an unsupervised manner. Similar to our depth prediction network, a recent work (Zhong et al., 2017) also investigates learning stereo disparity prediction without using ground truth data.

3.2 Feature Extraction

Generating robust feature descriptors is an important part of stereo matching and optical flow estimation systems. In this work we extract image-features using a convolutional network. Applying the fully convolutional network (Table 1) on *left* and *right* stereo-images, we extract features $\mathcal{F}_L$ and $\mathcal{F}_R$.

Feature-volume Generation

Features $\mathcal{F}_L$ and $\mathcal{F}_R$ are aggregated into feature-volumes $\mathcal{V}_L$ and $\mathcal{V}_R$ for the left and right images, respectively. Feature volumes are data structures that are convenient for matching features. Feature volume $\mathcal{V}_L$ is created by concatenating $\mathcal{F}_L$ with $\mathcal{F}_R$ translated at different disparity levels $d_i \in \{d_1,d_2,\ldots d_D\}$. $\mathcal{V}_L = [(\mathcal{F}_L,\mathcal{F}_R^{d_1}),(\mathcal{F}_L,\mathcal{F}_R^{d_2}),\ldots,(\mathcal{F}_L,\mathcal{F}_R^{d_D})]$[2] Similarly, $\mathcal{V}_R$ is created by translating $\mathcal{F}_L$ and concatenating it with $\mathcal{F}_R$.

[1] subscripts L and R indicate left and right views, respectively

[2] We used (X,Y) to denote *concatenating* X and Y across the first dimension and we use [X and Y] to denote *stacking* X and Y, which creates a new first dimension

Table 1: **Details of the feature extraction stage.** In the first entry of the third row, we use *res_1 to res_9*, as a shorthand notation for a stack of 9 identical residual layers.

Layers	Kernel size	Stride	Input channels	Output channels	Nonlinearity
conv_0	5x5	2x2	32	32	ReLU
res_1 to res_9	3x3	1x1	32	32	ReLU + BN
conv_10	3x3	1x1	32	32	-

Feature-volume Filtering

Generated feature volumes aggregate left and right image features at different disparity levels. This stage is posed with the task of computing pixel-wise probabilities over disparities from feature volumes. Since $\mathcal{V}_L$ and $\mathcal{V}_R$ are composed of features vectors spanning 3-dimensions (disparity, image-height, and image-width) it is convenient to use 3D convolutional layers. 3D convolutional layers are able to utilize neighborhood information across the above three axes. The 3D-convolutional encoder-decoder network used in this work is similar to Kendal et. al. (Kendall et al., 2017).

Let's denote the output of applying feature-volume filtering on $\mathcal{V}_L$ and $\mathcal{V}_R$ as $\mathcal{C}_L$ and $\mathcal{C}_R$, respectively. $\mathcal{C}_L$ and $\mathcal{C}_R$, are tensors represent pixel-wise disparity "confidences". In order to convert confidences into pixel-wise disparity maps, a *soft-argmin* function is used. $\mathcal{C}_L$ and $\mathcal{C}_R$ represent negative of confidence, hence, we use *soft-argmin*, instead of *soft-argmax*. The *soft-argmin* operation (Equation 1) is a differentiable alternative to *argmin* (Kendall et al., 2017). For every pixel location, *soft-argmin* first normalizes the depth confidences into a proper probability distribution function. Then, disparities are computed as the expectation of the disparity under the normalized distributions. Thus applying *soft-argmin* on $\mathcal{C}_L$ and $\mathcal{C}_R$ gives disparity maps $\mathcal{D}_L$ and $\mathcal{D}_R$, respectively.

$$\begin{aligned} \mathcal{P}_L(i,x,y) &= \frac{e^{-\mathcal{C}_L(i,x,y)}}{\sum_{j=1}^{D} e^{-\mathcal{C}_L(j,x,y)}} \\ \mathcal{D}_L(x,y) &= \sum_{i=1}^{D} disp(i) * \mathcal{P}_L(i,x,y) \end{aligned} \tag{1}$$

The estimated disparities $\mathcal{D}_L, \mathcal{D}_R$ are disparities that encode motions of pixels between the left and right images. We convert the predicted disparities into a sampling grid in order to warp left image to the right image (and vice versa). Once sampling grid is created we apply bi-linear sampling (Jaderberg et al., 2015) to perform the warping. Denoting the bi-linear sampling operation as Φ, the warped left and right images could be expressed as $\tilde{\mathcal{X}}_L = \Phi(\mathcal{X}_R, \mathcal{D}_L)$ and $\tilde{\mathcal{X}}_R = \Phi(\mathcal{X}_L, \mathcal{D}_R)$, respectively.

Disparity Estimation Network Training Objective

Our depth prediction network is trained in unsupervised manner by minimizing a loss term which mainly depends on the photometric discrepancy between the input images $\{\mathcal{X}_L,\mathcal{X}_R\}$ and their respective re-renderings $\{\tilde{\mathcal{X}}_L,\tilde{\mathcal{X}}_R\}$. Our network minimizes, the loss term $\mathcal{L}_T$ which has three components: a photometric discrepancy term $\mathcal{L}_P$, smoothness term $\mathcal{L}_S$ and left-right consistency term $\mathcal{L}_{LR}$, with different weighting parameters $\lambda_0, \lambda_1,$ and λ_2.

$$\mathcal{L}_T = \lambda_0 \mathcal{L}_P + \lambda_1 \mathcal{L}_{LR} + \lambda_2 \mathcal{L}_S \tag{2}$$

Photometric discrepancy term $\mathcal{L}_P$ is a sum of an $L1$ loss and a structural dissimilarity term based on SSIM (Wang et al., 2004) with multiple window sizes. In our experiments we use $S = \{3,5,7\}$. N in Equation 3 is the total number of pixels.

$$\begin{aligned} \mathcal{L}_P = &\frac{\lambda_1^p}{N}(||\tilde{\mathcal{X}}_L - \mathcal{X}_L||^1 + ||\tilde{\mathcal{X}}_R - \mathcal{X}_R||^1) + \\ &\frac{\lambda_s^p}{N}\sum_{s\in S}(SSIM_s(\tilde{\mathcal{X}}_L, \mathcal{X}_L) + SSIM_s(\tilde{\mathcal{X}}_R, \mathcal{X}_R)) \end{aligned} \tag{3}$$

Depth prediction from photo-consistency is an ill-posed inverse problem, where there are a large number of photo-consistent geometric structures for a given stereo pair. Imposing regularization terms encourages the predictions to be closer to the physically valid solutions. Therefore, we use an edge-aware smoothness regularization term $\mathcal{L}_S$ in Equation 5 and left-right consistency loss (Godard et al., 2016) Equation 4. $\mathcal{L}_{LR}$ is computed by warping disparities and comparing them to the original disparities predicted by the network.

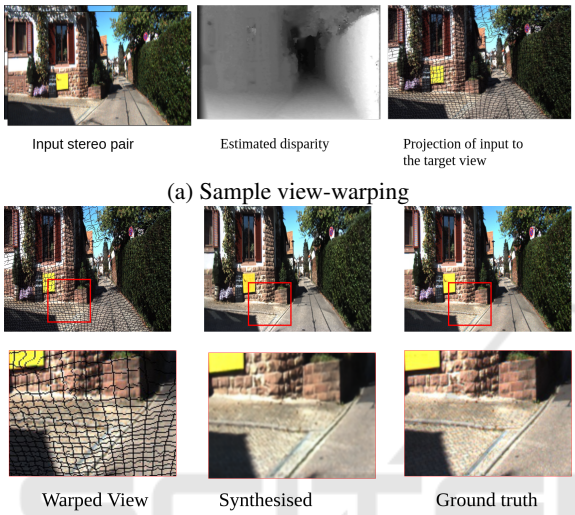
$$\begin{aligned} &\mathcal{L}_{\mathcal{LR}} = ||\tilde{\mathcal{D}_L} - \mathcal{D}_L||^1 + ||\tilde{D_R} - \mathcal{D}_R||^1 \\ &\text{where } \tilde{\mathcal{D}_L} = \Phi(\mathcal{D}_R, \mathcal{D}_L) \text{ and } \tilde{\mathcal{D}_{\mathcal{R}}} = \Phi(\mathcal{D}_L, \mathcal{D}_R) \end{aligned} \tag{4}$$

Smoothness term $\mathcal{L}_S$ forces disparities $\mathcal{D}_L$ and $\mathcal{D}_{\mathcal{R}}$ to have small gradient magnitudes in smooth image regions. However, the term allows large disparity gradients in regions where there are strong image gradients.

$$\begin{aligned} \mathcal{L}_S = &\nabla_x \mathcal{D}_L e^{(-\nabla_x \mathcal{X}_L)} + \nabla_y \mathcal{D}_L e^{(-\nabla_y \mathcal{X}_L)} + \\ &\nabla_x \mathcal{D}_R e^{(-\nabla_x \mathcal{X}_R)} + \nabla_y \mathcal{D}_R e^{(-\nabla_y \mathcal{X}_R)} \end{aligned} \tag{5}$$

Bilinear sampling, Φ allows back-propagation of error (sub-)gradients from the output such as $\tilde{\mathcal{X}}s$ back to the input images $\mathcal{X}$s and disparities $\mathcal{D}$s. Therefore, it is possible to use standard back-propagation to train our network. Formal derivations for the back-propagation of gradients via a bilinear sampling module could be found in (Jaderberg et al., 2015).

3.3 Forward Mapping



(a) Sample view-warping

(b) Close-up view of warped and target images

Figure 3: View-warping with forward mapping.

We apply our trained stereo depth predictor on the $\mathcal{V}$ input stereo pairs $\{\{\mathcal{X}_L^1,\mathcal{X}_R^1\},\{\mathcal{X}_L^2,\mathcal{X}_R^2\},\ldots,\{\mathcal{X}_L^V,\mathcal{X}_R^V\}\}$ and generate disparities for the left camera input views, $\{\mathcal{D}_L^1,\mathcal{D}_L^2,\ldots\mathcal{D}_L^V\}$. The right stereo views are used only for estimating disparities. Forward mapping and texture inpainting are done only on the images from the left camera. In this section, unless specified otherwise, we refer the images from left camera as the input images/views and we drop the subscripts L and R.

Forward-mapping projects input views to the target-view using their respective depth-maps. The predicted disparities of the input views could be converted to depth values. Depth $\mathcal{Z}_{w,h}^i$ for a pixel at location (w,h) in the $i-th$ input view, can be computed from the corresponding disparity $\mathcal{D}_{w,h}^i$, as follows:

$$\mathcal{Z}_{w,h}^i = \frac{f_x * B}{\mathcal{D}_{w,h}^i} \tag{6}$$

where K is the intrinsic camera matrix, B is the baseline, and f_x is focal length.

[3] ∇_x is gradient w.r.t x, similarly ∇_y is gradient w.r.t y

The goal of forward mapping is to project the input views to the target view, t. Given the relative pose between the input-view i and target-view as a transformation matrix $P^i = [R^i|T^i]$, pixel $p_{h,w}^i$ (pixel location $\{h,w\}$ in view i) will be forward mapped as follows, to a pixel location $p_{x,y}^t$ on the target view:

$$\begin{aligned} [x',y',z'] &\sim KP^i Z_{h,w}^i K^{-1}[h,w,1]^T \\ x &= \lfloor x'/z' \rfloor \text{ and } y = \lfloor y'/z' \rfloor \end{aligned} \tag{7}$$

Following a standard forward projection Equation 7, the reference input frames $\{\mathcal{X}^1,\mathcal{X}^2,\ldots,\mathcal{X}^V\}$ are warped to the target view $\{\mathcal{W}^1,\mathcal{W}^2,\ldots,\mathcal{W}^V\}$. As shown in Figure 3, forward-mapped views have a large number of pixel locations with unknown color values. These holes are created for various reasons. First, forward-mapping is a one-to-one mapping of pixels from the input views to the target view. This creates holes as it doesn't account for zooming in effects of camera movements, which could lead to one-to-many mapping. Moreover, occlusion and rounding effects lead to more holes. In addition to holes, some warped pixels have wrong color values due to inaccuracies in depth prediction.

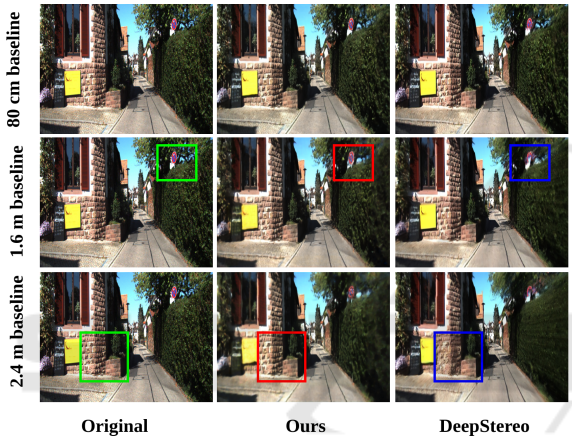
3.4 Texture Inpainting

The goal of our texture inpainting network is to learn generating the target view $\mathcal{X}^t$ from the set of warped input views $\{\mathcal{W}^1,\mathcal{W}^2,\ldots,\mathcal{W}^V\}$. This is a structured prediction task where the input and output images are aligned. Due to the effects mentioned above in section 3.3, texture mapping results in noisy warped views, see Figure 3. Forward-mapped images $\mathcal{W}^i s$ contain two kinds of pixels: *noisy-pixels*, those with unknown (or wrong) color values and *good-pixels*, those with correct color value. Ideally we would like to hallucinate the correct color value for the noisy pixels while maintaining the *good* pixels.

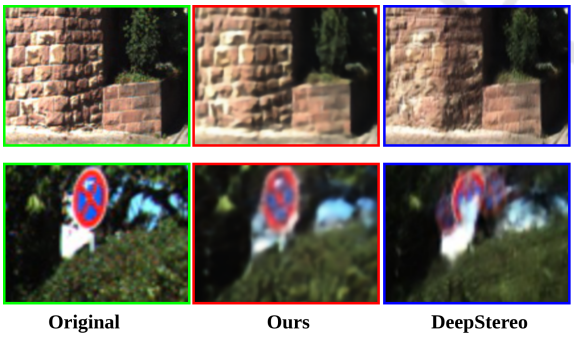
The architecture of our proposed *inpainting* network is inspired by Densely Connected (Huang et al., 2017) and Residual (He et al., 2016) network architectures. Details of the network architecture are presented in Table 2. The network has residual layers with long range skip-connections that feed features from early layers to the top layers, similar to DenseNets (Huang et al., 2017). The architecture is designed to facilitate flow of activations as well as gradients through the network. The texture inpainting network is trained by minimizing *L1* loss between the predicted novel views and the original images.

Table 2: **Texture inpainting network architecture.** The network is mainly residual, except special convolution layers which could be used as input and output layers.

Block	Layer	Input	Input,Out. channels	Output size
Block 0	conv_0	warped views	4*4,32	H, W
Block 0	res_1	conv_0	32,32	H, W
Block 1	res_2	pool(res_1), pool(warped views)	32+16, 48	H/2, W/2
Block 1	res_2 to res_7 (repeat res_2, 5 times)	res_2	48 ,48	H/2, W/2
Block 1	res_8	res_7	48 ,48	H/2, W/2
Block 2	conv_9	upsample(res_8), res_1	48+32,48	H, W
Block 2	res_10	conv_9	48, 48	H, W
Block 2	res_11	res_10	48, 48	H,W
Block 3	conv_12	res_11	48, 16	H,W
Block 3	output	conv_12	16, 3	H,W



(a) Rendering with our approach and DeepStereo (Flynn et al., 2016)



(b) Close-up view of warped and target images

Figure 4: Rendering of a sample scene for qualitative evaluation. Close-up views show that preserves the geometric structure better than *DeepStereo* and textitDeepStereo has ghosting where the traffic sign is replicated multiple times. Our rendering resembles the target except for slight amount of blur.

4 EXPERIMENTS

We tested our proposed approach on the KITTI (Geiger et al., 2012) public computer vision benchmark. Our approach has lower rendering error than the current state-of-the-art (Flynn et al., 2016) (see Table 3). Qualitative evaluation also show that out method better preserves the geometric details of the scene, shown in Figure 4.

Network Training and Hyper-parameters. The depth predictor loss term has three weighting parameters: λ_0 for photometric loss, λ_1 for left-right consistency loss and λ_2 for local smoothness regularization. The weighting parameters have to be set properly, wrong weights might lead trivial solutions such as a constant disparity output for every pixel. The weighting parameters used in our experiments are the following: $\lambda_0 = 5$, $\lambda_1 = 0.01$, $\lambda_2 = 0.0005$. The photometric loss is also a weighted combination of l1 loss and SSIM losses with different window sizes. These weighting factors are set to be $\lambda_P^1 = 0.2$, $\lambda_P^3 = 0.8$, $\lambda_P^5 = 0.2$ and $\lambda_P^7 = 0.2$. As discussed in section 3. the depth-prediction and texture inpainting networks are trained separately. We train the networks with back-propagation using Adam optimizer (Kingma and Ba, 2014), with mini-batch size 1 and learning rate of 0.0004. The depth prediction network is trained for 200,000 iterations and the inpainting network is trained for 1 Million iterations.

Dataset

We evaluate our proposed method on the publicly available KITTI Odometry Dataset (Geiger et al., 2012). KITTI is an interesting dataset for large-baseline novel view synthesis. The dataset has a large number of frames recorded in outdoors sunny urban environment. The dataset contains variations in the scene structure (vegetation, buildings, etc), strong illumination changes and noise which makes KITTI a challenging benchmark.

The dataset contains around 23000 stereo pairs divided into 11 sequences (*00* to *10*). Each sequence

Table 3: **Quantitative evaluation of our method against *DeepStereo***. We used our texture inpainting network with residual blocks as shown in Table 2. Our approach outperforms *DeepStereo* (Flynn et al., 2016) in all cases. Each row shows the performance of a method that is trained on a specific input camera spacing and tested on all three spacings.

Spacing	Method	Test 0.8 m	Test 1.6 m	Test 2.4 m
Train 0.8 m	Ours	**6.66**	**8.90**	**12.14**
	DeepStereo	7.49	10.41	13.44
Train 1.6 m	Ours	**6.92**	**8.47**	**10.38**
	DeepStereo	7.60	10.28	12.97
Train 2.4 m	Ours	**7.47**	**8.73**	**10.28**
	DeepStereo	8.06	10.73	13.37

contains a set of images captured by a stereo camera, mounted on top of car driving through a city. The stereo camera captures frames every ≈ 80 *cms*. In order to make our results comparable to *DeepStereo* (Flynn et al., 2016), we hold out *sequence 04* for validation, *sequence 10* for test and used the rest for training. In our experiments, 5 consecutive images are used, with the middle one is held out as target and the other 4 images are used as reference. Tests are done at different values of spacing between consecutive images. Taking consecutive frames gives spacing of around 0.8 *m*. Sampling every other frame give a spacing of $\approx 1.6m$ and every third frame gives $\approx 2.4m$ spacing. The error metric used to evaluate performance of rendering methods a mean absolute brightness error, computed per pixel per color channel.

Results

Table 3 shows the results of our proposed approach compared to DeepStereo (Flynn et al., 2016), on the KITTI dataset. Our approach outperforms *DeepStereo* in all spacings of the input views, while being significantly faster. In Figure 4, sample renderings are shown for a qualitative evaluation of our approach and *DeepStereo*. As shown in the lowest two rows of Figure 4, renderings of *DeepStereo* suffer from ghosting effects due to cross-talk between different depth planes, which led to the replicated traffic sign and noisy bricks(as could be seen in the close-up views).

We performed tests to compare our texture inpainting architecture against the so called UNet architecture (Ronneberger et al., 2015). Furthermore, in order to evaluate the significance of the texture inpainting stage, we measured the accuracy of our system when the texture inpainting stage is replaced by simple median filtering scheme applied on the warped views. In an ablation test, we found out that our inpainting network achieves lower error than a commonly used architecture called UNet (Ronneberger et al., 2015)(error 6.66 vs 7.08). A test performed to investigate the effect of using residual layers instead of convolutional layers shows that residual layers give slightly better performance (error 6.66 vs 6.70).

Timing. During the test phase, at a resolution of 528×384, our depth prediction stage and the forward mapping take 6.08 seconds and 2.60 seconds (for four input neighbouring views), respectively. The texture rendering takes takes 0.05 seconds. Thus, the total time adds up to 8.73 seconds, per frame. This is much faster than *DeepStereo* which takes 12 minutes to render a single frame of resolution 500×500. Our experiments are performed on a multi-core cpu machine with single Tesla V100 GPU.

5 CONCLUSION AND FUTURE WORK

We presented a fast and accurate novel view synthesis pipeline based on stereo-vision and convolutional networks. Our method decomposes novel view synthesis into, view-dependent geometry estimation and texture inpainting problems. Thus, our method utilizes the power of convolutional neural networks in learning structured prediction tasks.

Our proposed method is tested on a challenging benchmark, where most existing approaches are not able to produce reasonable results. The proposed approach is significantly faster and more accurate than the current state-of-the-art. As part of a future work, we would like to explore faster architectures to achieve real-time performance.

ACKNOWLEDGEMENTS

This work was partially funded by the BMBF project VIDIETE(01IW18002).

REFERENCES

Adelson, E. H., Bergen, J. R., et al. (1991). The plenoptic function and the elements of early vision.

Chaurasia, G., Duchene, S., Sorkine-Hornung, O., and Drettakis, G. (2013). Depth synthesis and local warps

for plausible image-based navigation. *ACM Transactions on Graphics (TOG)*, 32(3):30.

Chen, S. E. and Williams, L. (1993). View interpolation for image synthesis. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 279–288. ACM.

Dosovitskiy, A., Tobias Springenberg, J., and Brox, T. (2015). Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1538–1546.

Fitzgibbon, A., Wexler, Y., and Zisserman, A. (2005). Image-based rendering using image-based priors. *International Journal of Computer Vision*, 63(2):141–151.

Flynn, J., Neulander, I., Philbin, J., and Snavely, N. (2016). Deepstereo: Learning to predict new views from the world's imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5515–5524.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE.

Godard, C., Mac Aodha, O., and Brostow, G. J. (2016). Unsupervised monocular depth estimation with left-right consistency. *arXiv preprint arXiv:1609.03677*.

Goesele, M., Ackermann, J., Fuhrmann, S., Haubold, C., Klowsky, R., Steedly, D., and Szeliski, R. (2010). Ambient point clouds for view interpolation. In *ACM Transactions on Graphics (TOG)*, volume 29, page 95. ACM.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hosni, A., Bleyer, M., Rhemann, C., Gelautz, M., and Rother, C. (2011). Real-time local stereo matching using guided image filtering. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, pages 1–6. IEEE.

Huang, G., Liu, Z., Weinberger, K. Q., and van der Maaten, L. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3.

Jaderberg, M., Simonyan, K., Zisserman, A., et al. (2015). Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025.

Kalantari, N. K., Wang, T.-C., and Ramamoorthi, R. (2016). Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics (TOG)*, 35(6):193.

Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P., Kennedy, R., Bachrach, A., and Bry, A. (2017). End-to-end learning of geometry and context for deep stereo regression. *arXiv preprint arXiv:1703.04309*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Liu, M., He, X., and Salzmann, M. (2018). Geometry-aware deep network for single-image novel view synthesis. *arXiv preprint arXiv:1804.06008*.

Penner, E. and Zhang, L. (2017). Soft 3d reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)*, 36(6):235.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Scharstein, D. (1996). Stereo vision for view synthesis. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*, pages 852–858. IEEE.

Seitz, S. M. and Dyer, C. R. (1995). Physically-valid view synthesis by image interpolation. In *Representation of Visual Scenes, 1995.(In Conjuction with ICCV'95), Proceedings IEEE Workshop on*, pages 18–25. IEEE.

Seitz, S. M. and Dyer, C. R. (1996). View morphing. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 21–30. ACM.

Tatarchenko, M., Dosovitskiy, A., and Brox, T. (2016). Multi-view 3d models from single images with a convolutional network. In *European Conference on Computer Vision*, pages 322–337. Springer.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612.

Yang, J., Reed, S. E., Yang, M.-H., and Lee, H. (2015). Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. In *Advances in Neural Information Processing Systems*, pages 1099–1107.

Yin, X., Wei, H., Wang, X., Chen, Q., et al. (2018). Novel view synthesis for large-scale scene using adversarial loss. *arXiv preprint arXiv:1802.07064*.

Zhong, Y., Dai, Y., and Li, H. (2017). Self-supervised learning for stereo matching with self-improving ability. *arXiv preprint arXiv:1709.00930*.

Zhou, T., Tulsiani, S., Sun, W., Malik, J., and Efros, A. A. (2016). View synthesis by appearance flow. *arXiv preprint arXiv:1605.03557*.

Zitnick, C. L., Kang, S. B., Uyttendaele, M., Winder, S., and Szeliski, R. (2004). High-quality video view interpolation using a layered representation. In *ACM transactions on graphics (TOG)*, volume 23, pages 600–608. ACM.