Single-View View Synthesis with Multiplane Images

Richard Tucker Noah Snavely
Google Research
richardt@google.com snavely@google.com

Abstract

A recent strand of work in view synthesis uses deep learning to generate multiplane images—a camera-centric, layered 3D representation—given two or more input images at known viewpoints. We apply this representation to ***single-view*** *view synthesis, a problem which is more challenging but has potentially much wider application. Our method learns to predict a multiplane image directly from a single image input, and we introduce* ***scale-invariant view synthesis*** *for supervision, enabling us to train on online video. We show this approach is applicable to several different datasets, that it additionally generates reasonable depth maps, and that it learns to fill in content behind the edges of foreground objects in background layers.*

Project page at https://single-view-mpi.github.io/.

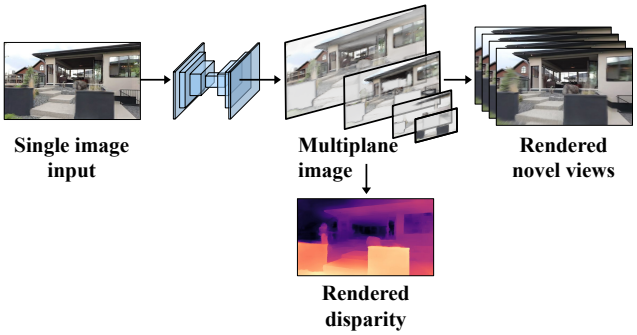

Figure 1. Our network generates a multiplane image (MPI) from a single image input. The MPI can be used to render images from novel viewpoints, and to generate a disparity map. (Video frames here and in other figures are used under Creative Commons license from Youtube user *Sona Visual*.)

1. Introduction

Taking a photograph and being able to move the camera around is a compelling way to bring photos to life. It requires understanding the 3D structure of the scene, reasoning about occlusions and what might be behind them, and rendering high quality, spatially consistent new views in real time.

We present a deep learning approach to this task which can be trained on online videos or multi-camera imagery using view synthesis quality as an objective—hence, the approach does not require additional ground truth inputs such as depth. At inference time, our method takes a single RGB image input and produces a representation of a local light field. We adopt the *multiplane image* (MPI) representation, which can model disocclusions and non-Lambertian effects, produces views that are inherently spatially consistent, is well-suited to generation by convolutional networks, and can be rendered efficiently in real time [37].

Our approach is the first to generate multiplane images directly from a *single* image input, whereas prior work has only estimated MPIs from multiple input views (anywhere from a stereo pair [37] to twelve images from a camera array [4]). Compared with multiple-input view synthesis, ours is a much more challenging task. We want the network to learn where different parts of the scene are in space without being able to observe correlations between multiple views, and without any chance to look even a tiny bit 'around the corner' of objects. A particular difficulty arises when supervising such a system using view synthesis because of the global scale ambiguity inherent in the input data. We address this with a method of *scale invariant view synthesis* which makes use of sparse point sets produced in the course of generating our training data. We also introduce an *edge-aware smoothness loss* which discourages the depth-maps derived from our predicted MPIs from being unnaturally blurry, even in the absence of depth supervision.

We train and evaluate our method using a dataset of online videos, and measure the quality of derived depth-estimates on the iBims-1 benchmark. We show the versatility of our approach by comparing it to two previous view synthesis methods using different representations: one that predicts complete 4D light fields and learns from a narrow-baseline multi-angle light field dataset, and another that predicts layered depth images and learns from wide-baseline stereo-camera imagery. Our method not only achieves higher quality view synthesis than these but is more general, not requiring light field data or known-scale inputs for training.

2. Related work

We build on work in two areas—view synthesis and depth prediction—that are themselves highly related. Learning-based methods have been applied to both of these domains.

Single-view depth prediction. There has been great interest in the task of predicting a depth map, or other suitable geometric representation, from a single RGB image. However, depth maps alone do not enable full synthesis of new views, because they do not capture content that is occluded in the reference view but visible in a desired target view. Conversely, accurate depth is also not strictly required for high-quality view synthesis—for instance, inaccurate depth in textureless regions might be imperceptible, and planarity relationships might be more perceptually important compared to strict accuracy. At the same time, depth and view synthesis are highly intertwined, and many recent depth prediction methods use view synthesis as implicit supervision [7, 9, 36]. Like these methods, we use additional views of scenes as supervision, but we explicitly focus on the application of new view synthesis, and hence use a more expressive scene representation (MPIs) compared to depth maps.

Other recent work, like ours, uses videos in the wild as a source of training data for geometric learning. For instance, Lasinger *et al*. learn a robust single-view depth predictor from a large dataset of 3D movies, by first extracting optical flow between the left and right frames as a form of pseudo-depth for supervision [15]. Chen *et al*. produce large quantities of sparse SfM-derived depth measurements from YouTube videos for use in training depth networks [2]. However, these prior methods focus on depth, while our method is the first to our knowledge to learn single-view view synthesis from videos in the wild.

Learned view synthesis. Traditionally, methods for view synthesis operated in the interpolation regime, where one is provided with multiple views of a scene, and wishes to interpolate views largely within the convex hull of their camera positions. A number of classical approaches to this problem have been explored [10, 16], including methods that involve estimation of local geometric proxies [1, 11, 23, 38].

Learning-based approaches to this interpolation problem have also been explored. Learning is an attractive tool for view synthesis because a training signal can be obtained simply from having held-out views of scenes from known viewpoints, via predicting those views and comparing to the ground truth images. Some approaches predict new views independently for each output view, leading to inconsistency from one view to the next [5, 12]. Other methods predict a single scene representation from which multiple output views can be rendered. In particular, *layered* representations are especially attractive, due to their ability to represent occluded content. For instance, *multiplane images* (MPIs), originally devised for stereo matching problems [30], have recently found success in both learned view interpolation and extrapolation from multiple input images [37, 27, 4, 21]. However, none of these methods are able to predict an MPI from a single input image.

Most related to our work are methods that predict new views from *single* images. This includes work on synthesizing a full light field from a single view [28], predicting soft disparity maps [34], inferring layered representations like layered depth images (LDIs) [31, 26], or segmenting the input and predicting a 3D plane for each segment [19]. We borrow the MPI representation introduced for view interpolation and extrapolation, apply it to the *single-view* case, and show that this representation leads to higher quality results compared to light fields and LDIs.

Depth can also be used as a starting point for view synthesis, as in recent work from Niklaus *et al*. that predicts a depth map from a single view, then inpaints content behind the visible surfaces to enable high-quality single-image view synthesis [22]. However, this method requires dense, accurate depth supervision and multiple stages of post-processing. Our method learns to predict an MPI as a single stage, using only multiple views (e.g., video frames) as supervision.

3. Approach

At inference time, our method takes a single input image and generates a representation from which novel views at new camera positions can be freely generated (Fig. 1). For training, all we require is videos with static scenes and moving camera, which we process as follows.

3.1. Data

We apply SLAM (Simultaneous Localization and Mapping) and structure-from-motion algorithms to videos to identify motion sequences, estimate viewpoints, and generate a sparse point cloud. We follow the method of Zhou *et al*. [37]: the only difference is that we retain the sparse point cloud and a record of which points were tracked in each frame (referred to as *visible points*), which they do not use.

At training time, we sample pairs of frames (*source* and *target*) from the resulting sequences. Each pair gives us a source image $\mathbf{I}_s$ and a target image $\mathbf{I}_t$, together with their viewpoints v_s and v_t (camera intrinsics and extrinsics). Additionally, we extract the set of visible points for the source frame and map them into camera space, resulting in a set $\mathbf{P} = \{(x, y, d), \ldots\}$ of triples where (x, y) is the position within the source image, and d the depth of that point.

In our experiments, we apply this processing to videos from the RealEstate10K dataset [37], giving us over 70000 sequences and over 9 million frames.

3.2. Representation and rendering

We use multiplane images (MPIs) as our scene representation, which support differentiable rendering [37]. As

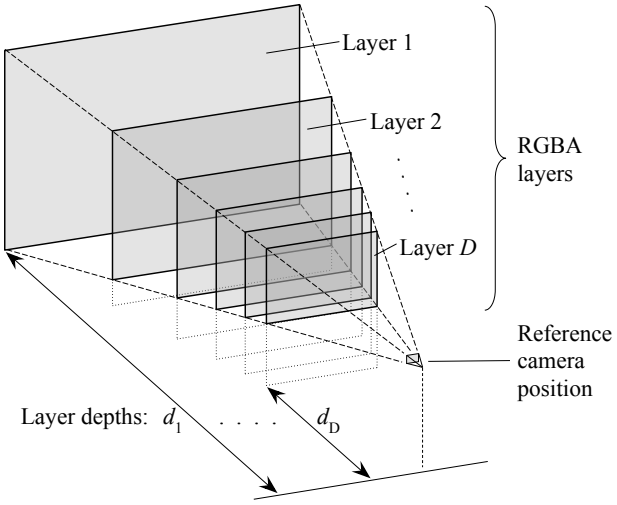

Figure 2. The multiplane image representation. See Section 3.2.

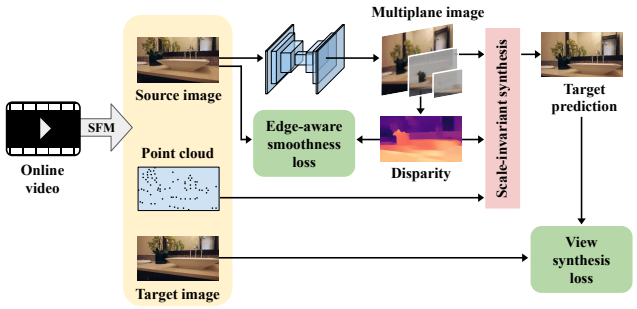

Figure 3. Our system, trained on online video, learns to predict multiplane images directly from single image inputs. Scale-invariant view synthesis allows us to apply view synthesis loss despite the global scale ambiguity in our training data.

illustrated in Fig. 2, an MPI consists of a set of D fronto-parallel planes in the frustum of a reference camera, arranged at fixed depths $d_1, \ldots, d_D$, from $d_1 = d_{\text{far}}$ to $d_D = d_{\text{near}}$, and equally spaced in disparity (inverse depth). Each plane or layer has an RGBA image: we write c_i and α_i for the color and alpha channels of layer i, each with a resolution $W \times H = N$. An MPI can also be considered as an instance of the *stack of acetates* model of Szeliski and Golland [30] with soft alpha and a specific choice of layer depths.

Given a source image $\mathbf{I}_s$ at viewpoint v_s our network f outputs an MPI whose reference camera is at v_s:

$$\{(c_1, \alpha_1), \ldots, (c_D, \alpha_D)\} = f(\mathbf{I}_s). \tag{1}$$

Warping. The first step in rendering a novel image from an MPI is to warp each layer from the source viewpoint to the desired target viewpoint v_t:

$$c'_i = \mathcal{W}_{v_s, v_t}(\sigma d_i, c_i), \quad \alpha'_i = \mathcal{W}_{v_s, v_t}(\sigma d_i, \alpha_i). \tag{2}$$

The warping operation $\mathcal{W}$ computes the color or alpha value at each pixel in its output by sampling bilinearly from the input color or alpha. To do this, it applies a homography to each target pixel's coordinates (u_t, v_t) to obtain corresponding source coordinates (u_s, v_s) at which to sample:

$$\begin{bmatrix} u_s \\ v_s \\ 1 \end{bmatrix} \sim K_s \left(R - \frac{\mathbf{t}\mathbf{n}^\mathsf{T}}{\mathbf{a}} \right) K_t^{-1} \begin{bmatrix} u_t \\ v_t \\ 1 \end{bmatrix}, \tag{3}$$

where $\mathbf{n}$ is the normal vector and $\mathbf{a}$ the distance (both relative to the target camera) to a plane that is fronto-parallel to the source camera at depth σd_i, R is the rotation and $\mathbf{t}$ the translation from v_t to v_s, and K_s, K_t are the source and target camera intrinsics.

This procedure and the compositing that follows are the same as in Zhou *et al*. [37], except for the introduction of the scale factor σ in Eq. 2 and Eq. 3. The layer depths d_i are multiplied by σ, scaling the whole MPI up or down correspondingly. As described in Section 3.3, choosing the right scale σ allows us to overcome the scale ambiguity inherent to SfM models and achieve scale-invariant synthesis.

Compositing. The warped layers (c'_i, α'_i) are composited using the *over* operation [24] to give the rendered image $\hat{\mathbf{I}}_t$:

$$\hat{\mathbf{I}}_t = \sum_{i=1}^{D} \left(c'_i \alpha'_i \prod_{j=i+1}^{D} (1 - \alpha'_j) \right). \tag{4}$$

We can also synthesize a disparity-map $\hat{\mathbf{D}}_s$ from an MPI, by compositing the layer disparities (i.e. inverse depths):

$$\hat{\mathbf{D}}_s = \sum_{i=1}^{D} \left(d_i^{-1} \alpha_i \prod_{j=i+1}^{D} (1 - \alpha_j) \right). \tag{5}$$

Note that although layer depths are discrete, the disparity-map can be smooth because α_i blends softly between layers.

3.3. Scale-invariant synthesis

Visual SLAM and structure-from-motion have no way of determining absolute scale without external information: each of our training sequences is therefore equally valid if we scale the world (including the sparse point sets and the translation part of the camera poses) up or down by any constant factor. This is not an issue when dealing with multiple-image input since the relative pose between the inputs resolves the scale ambiguity, but it poses a challenge for learning any sort of 3D representation from a *single* input. To address this ambiguity, prior work on single-view depth prediction typically employs a *scale-invariant depth loss* [3, 32] or more recently even a *scale-and-shift invariant depth loss* [15]. These methods can be seen as finding the

scale factor which minimizes a scale-dependent loss, and rely on there being a closed-form solution for this scale factor.

View-synthesis losses are subject to the same problem: the scale of each training instance is arbitrary, but without the correct scale, rendered images cannot match ground truth. We could try to minimize a view synthesis loss over all possible scale factors, but the rendering operations (Eqs. 2–4) make this not amenable to a closed-form solution, and direct optimization for scale at run-time will be prohibitively slow.

We observe that although scale is unknown, the camera poses v_s, v_j and the point set $\mathbf{P}_s$ do have a *consistent* scale for each training example. Therefore we can use the point set to compute a scale factor to apply in rendering. We compute the scale factor σ which minimizes log-squared error between the predicted disparity $\hat{\mathbf{D}}_s$ and the point set $\mathbf{P}_s$:

$$\sigma = \exp\left[\frac{1}{|\mathbf{P}_s|}\sum_{(x,y,d)\in\mathbf{P}_s}(\ln\hat{\mathbf{D}}_s(x,y) - \ln(d^{-1}))\right] \quad (6)$$

where $\hat{\mathbf{D}}_s(x,y)$ denotes bilinear sampling from the disparity map at position (x,y). The scale factor σ thus obtained is applied in Eqs. 2 and 3, ensuring that the rendered image $\hat{\mathbf{I}}_t$ no longer varies with the scale of the input viewpoints and point set. $\hat{\mathbf{I}}_t$ is therefore suitable for use in training with view-synthesis losses.

3.4. Losses

Our overall loss combines a view synthesis loss, a smoothness loss on the synthesized disparity, and a sparse depth supervision loss:

$$\mathcal{L} = \lambda_{\mathsf{p}}\mathcal{L}^{\mathsf{pixel}} + \lambda_{\mathsf{s}}\mathcal{L}^{\mathsf{smooth}} + \lambda_{\mathsf{d}}\mathcal{L}^{\mathsf{depth}} \quad (7)$$

We now describe each of these in turn.

Synthesis. To encourage the rendered image at the target viewpoint to match the ground truth, we use an L^1 per-pixel loss:

$$\mathcal{L}^{\mathsf{pixel}} = \sum_{\text{channels}}\frac{1}{N}\sum_{(x,y)}|\hat{\mathbf{I}}_t - \mathbf{I}_t|. \quad (8)$$

We can optionally add an image gradient term to this, but we did not find it to be consistently helpful.

Edge-aware smoothness. For natural images, depth discontinuities are typically accompanied by discontinuities in the image itself (though the reverse is not the case) [6]. This idea has been used in classical computer vision, notably in stereo correspondence [25], and also in a variety of different smoothness losses for learning depth prediction [9, 17, 33]. These losses work by encouraging depth to be smooth wherever the input image is smooth.

We apply this idea as follows. First, let $\mathbf{G}$ be the sum over all channels of the L^1 norm of the gradient of an image (we use Sobel filters to compute the gradient):

$$\mathbf{G}(\mathbf{I}) = \sum_{\text{channels}}\left\|\nabla\mathbf{I}\right\|_1 \quad (9)$$

We define a *source edge mask* $\mathbf{E}_s$ which is 1 wherever the source image gradient is at least a fraction e_{min} of its maximum over the image.

$$\mathbf{E}_s = \min\left(\frac{\mathbf{G}(\mathbf{I}_s)}{e_{\mathsf{min}} \times \max_{(x,y)}\mathbf{G}(\mathbf{I}_s)}, 1\right) \quad (10)$$

Our *edge-aware smoothness loss* then penalizes gradients higher than a threshold g_{min} in the predicted disparity map, but only in places where the edge mask is less than one:

$$\mathcal{L}^{\mathsf{smooth}} = \frac{1}{N}\sum_{(x,y)}\left(\max(\mathbf{G}(\hat{\mathbf{D}}_s) - g_{\mathsf{min}}, 0)\odot(1-\mathbf{E}_s)\right), \quad (11)$$

where $\odot$ is the Hadamard product. As with our synthesis loss, $\mathcal{L}^{\mathsf{smooth}}$ is an average over all pixels. In practice we set $e_{\mathsf{min}} = 0.1$ and $g_{\mathsf{min}} = 0.05$.

As noted earlier, there are many possible formulations of such a loss. Our $\mathcal{L}^{\mathsf{smooth}}$ is one that we found creates qualitatively better depth maps in our system, by allowing gradual changes in disparity while encouraging discontinuities to be accurately aligned to image edges.

Sparse depth supervision. The point set $\mathbf{P}_s$ allows us to apply a form of direct but sparse depth supervision. We adopt the L^2 loss of Eigen *et al.* on log disparity [3] (as noted in Section 3.3, σ is the scale factor that minimizes this loss—it is equivalent to the variable α in Eigen *et al.*'s scale-invariant loss, under $\ln\sigma = \alpha$):

$$\mathcal{L}^{\mathsf{depth}} = \frac{1}{|\mathbf{P}_s|}\sum_{(x,y,d)\in\mathbf{P}_s}\left(\ln\frac{\hat{\mathbf{D}}_s(x,y)}{\sigma} - \ln(d^{-1})\right)^2 \quad (12)$$

3.5. Implementation

Network. We use a DispNet-style network [20], specified in Table 1. We pad the input (the single RGB image $\mathbf{I}_s$) to a multiple of 128 in height and width, and crop the output correspondingly. The first $D-1$ channels of the output give us $\alpha_2, \ldots, \alpha_D$. The back layer is always opaque, so $\alpha_1 = 1$ and need not be output from the network. When initializing our network for training, we set the bias weights on the last convolutional layer so that the mean of the initial output distribution corresponds to an initial alpha value of $1/i$ in layer i. This *harmonic bias* helps ameliorate an issue during training in which layers which are not near the front of the MPI volume are heavily occluded and have very small gradients with respect to our losses.

Input	k_1	c_1	k_2	c_2	Output
$\mathbf{I}_s$	7	32	7	32	$conv_1$
$MP_2(conv_1)$	5	64	5	64	$conv_2$
$MP_2(conv_2)$	3	128	3	118	$conv_3$
$MP_2(conv_3)$	3	256	3	256	$conv_4$
$MP_2(conv_4)$	3	512	3	512	$conv_5$
$MP_2(conv_5)$	3	512	3	512	$conv_6$
$MP_2(conv_6)$	3	512	3	512	$conv_7$
$MP_2(conv_7)$	3	512	3	512	$conv_8$
$Up_2(conv_8)$ + $conv_7$	3	512	3	512	$conv_9$
$Up_2(conv_9)$ + $conv_6$	3	512	3	512	$conv_{10}$
$Up_2(conv_{10})$ + $conv_5$	3	512	3	512	$conv_{11}$
$Up_2(conv_{11})$ + $conv_4$	3	512	3	512	$conv_{12}$
$Up_2(conv_{12})$ + $conv_3$	3	128	3	128	$conv_{13}$
$Up_2(conv_{13})$ + $conv_2$	3	64	3	64	$conv_{14}$
$Up_2(conv_{14})$ + $conv_1$	3	64	3	64	$conv_{15}$
$conv_{15}$	3	64	3	64	$conv_{16}$
$conv_{16}$	3	34	-	-	output

Table 1. Our network architecture. Each row describes *two* convolutional layers in sequence: $\mathbf{k}_1, \mathbf{k}_2$ are the kernel sizes and $\mathbf{c}_1, \mathbf{c}_2$ the numbers of output channels. **Input** shows the input to the first layer, where MP_2 denotes maxpooling with a pool size of 2 (thus halving the size), Up_2 denotes nearest-neighbour upscaling by a factor of 2, and $+$ is concatenation. Each layer is followed by ReLU activation. The final row shows a single convolutional layer, which is instead followed by sigmoid activation. For details of how the outputs are translated into MPI layers, see Section 3.5.

We follow Zhou *et al*. [37] and model each layer's color as a per-pixel blend of the input image with a predicted global *background image* $\hat{\mathbf{I}}_{bg}$. In that work, blend weights are predicted for each pixel in each MPI layer. We reason instead that content that is visible (from the source viewpoint) should use the foreground image, and content that is fully occluded should use the background image. Therefore we can derive the blend weights w_i from the alpha channels as follows:

$$w_i = \prod_{j>i}(1-\alpha_j), \tag{13}$$

$$c_i = w_i \mathbf{I}_s + (1-w_i)\hat{\mathbf{I}}_{bg}. \tag{14}$$

The background image is determined by the remaining three channels of the network output. Because it is difficult for the network to learn to predict α_i and $\hat{\mathbf{I}}_{bg}$ simultaneously, during training we set $\hat{\mathbf{I}}_{bg}$ to be a linear interpolation between $\mathbf{I}_s$ and the network output, with the contribution of the network increasing gradually over the first s_{bg} training steps.

Training. In our experiments, D (the number of MPI planes) is 32, $s_{\mathsf{bg}} = 100{,}000$, and our losses are weighted as follows: $\lambda_{\mathsf{p}} = 1$, $\lambda_{\mathsf{s}} = 0.5$, $\lambda_{\mathsf{p}} = 0.1$. We train using the Adam Optimizer [13] with a learning rate of 0.0001.

4. Experiments

We present quantitative and qualitative evaluations of our method on the RealEstate10K dataset, depth evaluations with the iBims-1 benchmark, and comparisons with previous view synthesis methods on the Flowers and KITTI datasets. Because of the very visual nature of the view synthesis task, we strongly encourage the reader to view the additional examples, including animations, in our supplementary video.

4.1. View synthesis on RealEstate10K

To investigate the effects of our different losses and MPI background prediction, we train several versions of our method on videos from the RealEstate10K dataset [37]:

- **full**: Our full method, as described in Section 3.
- **nodepth**: As **full**, but with no depth loss, i.e. $\lambda_{\mathsf{d}} = 0$.
- **noscale**: As **full**, but with no depth loss and no scale-invariance, i.e. $\lambda_{\mathsf{d}} = 0$, $\sigma = 1$.
- **nosmooth**: As **full**, but with no edge-aware disparity smoothness loss, i.e. $\lambda_{\mathsf{s}} = 0$.
- **nobackground**: As **full**, but with no background prediction. Instead, all MPI layers take their color from the input, i.e. $c_i = \mathbf{I}_s$.

To compare these methods, we measure the accuracy of synthesized images using the LPIPS perceptual similarity metric [35] and PSNR and SSIM metrics, on a held-out set of 300 test sequences, choosing source and target frames to be 5 or 10 frames apart. At test time, we use the point set to compute the scale factor σ in the same way as we do during training—for a fair comparison, we also do this for the **noscale** model. Results are in Table 2 ($LPIPS_{all}$, $PSNR_{all}$ and $SSIM_{all}$ columns).

We observe that **nodepth** performs a little worse than **full**, and **noscale** performs considerably worse still. This shows that direct depth supervision—although sparse—is of some benefit, but that the improvement from our *scale-invariant synthesis* is more significant. As expected, for all variants, performance decreases with larger camera movement.

Somewhat unintuitively, the **nosmooth** and **nobackground** models outperform the **full** model on PSNR and SSIM metrics. But at larger distances the LPIPS metric, which attempts to measure perceptual similarity, shows benefits from our smoothness loss and from allowing the network to predict the background layer, with the **full** model performing best.

Qualitatively, the **nobackground** model introduces unpleasant artefacts at the edges of foreground objects, whereas the **full** model is able to use the background layer to predict the appearance of some disoccluded content, as shown in Fig. 4. To quantify this effect, we first compute a *disocclusion mask* $\mathbf{M}_t$ for each image by warping and compositing the blend weights w_i used in the **full** model:

Method	$\text{LPIPS}_{\text{all}} \downarrow$		$\text{PSNR}_{\text{all}} \uparrow$		$\text{SSIM}_{\text{all}} \uparrow$		$\text{PSNR}_{\text{disocc}} \uparrow$		$\text{SSIM}_{\text{disocc}} \uparrow$	
	$n = 5$	$n = 10$	$n = 5$	$n = 10$	$n = 5$	$n = 10$	$n = 5$	$n = 10$	$n = 5$	$n = 10$
full	0.103	**0.155**	26.4	23.5	0.859	0.795	**19.7**	17.9	0.513	0.480
nodepth	0.120	0.178	26.2	23.4	0.854	0.791	19.2	18.0	0.525	0.496
noscale	0.149	0.221	25.4	22.8	0.837	0.771	18.5	17.3	0.496	0.470
nosmooth	0.104	0.159	26.4	23.6	0.860	0.798	19.6	**18.4**	**0.540**	**0.527**
nobackground	**0.099**	0.162	**26.8**	**23.7**	**0.867**	**0.802**	18.7	17.7	0.509	0.499

Table 2. Ablation studies on images from RealEstate10K video sequences. n indicates the number of frames between source and target in the video sequence. '*all*' metrics are computed on the whole image (with a 5% crop), '*disocc*' metrics on disoccluded pixels only, i.e. those where $\mathbf{M}_t > 0.6$. We observe that scale-invariance gives a large benefit, depth supervision a smaller one, and that predicting background content does not clearly help overall but does improve performance for disoccluded pixels and on perceptual similarity. See Section 4.1.

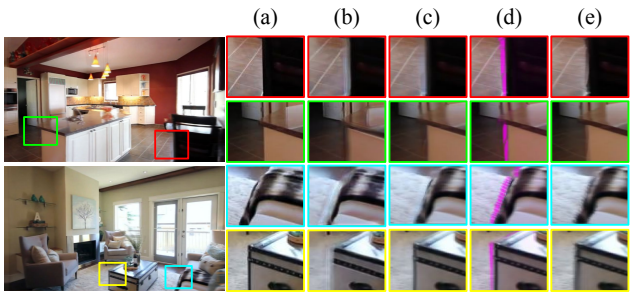

Figure 4. Use of the background image. For each region we show (a) the input image $\mathbf{I}_s$, (b) the predicted background $\hat{\mathbf{I}}_{bg}$, (c) a rendering generated by our **full** method from a slightly different viewpoint, (d) a visualization of where $\hat{\mathbf{I}}_{bg}$ is used in rendering: pixels whose value comes 90% or more from the background are highlighted, (e) the same region rendered by our **nobackground** model. Comparing (c) and (e), images rendered by our **full** model show cleaner edges with fewer artefacts than those rendered by the **nobackground** model. Comparing (a) and (b), the network has learned to erode the edges of foreground objects and to predict what color may be behind them, although some artefacts remain.

$$w'_i = \mathcal{W}_{v_s,v_t}(\sigma d_i, w_i),$$
$$\mathbf{M}_t = 1 - \sum_{i=1}^{D}\Big(w'_i\alpha'_i \prod_{j=i+1}^{D}(1-\alpha'_j)\Big). \qquad (15)$$

$\mathbf{M}_t$ tells us how much of the composited value at each pixel comes from the background image. We use it to compute metrics only on dioscclude pixels, i.e. those where $\mathbf{M}_t$ is greater than some threshold. Results are in Table 2 ($\text{PSNR}_{\text{disocc}}$ and $\text{SSIM}_{\text{disocc}}$ columns). Although **nobackground** achieves slightly better scores than **full** over the whole image, it performs worse on these disoccluded areas.

The **nosmooth** model achieves plausible view synthesis results, but this is not the only potential application of MPIs. For other tasks, such as editing or object insertion, it is desirable to have accurate depth maps. As shown in Fig. 5, **nosmooth** performs significantly worse than our full model

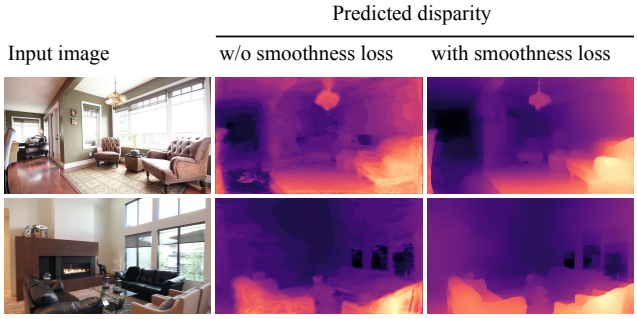

Figure 5. Effect of smoothness loss on predicted disparity. As shown in these examples, our edge-aware smoothness loss encourages the predicted disparity to be smooth where the input image is smooth, and consequently also encourages it to have sharp edges aligned with visible object boundaries.

in this regard: it both lacks sharp edges where depth should be discontinuous, and introduces discontinuities where depth should be smooth. Our point set depth data is insufficient to evaluate depth accuracy on the RealEstate10K dataset, so for a quantitative measurement we turn to another benchmark.

4.2. Depth evaluation

While our objective is view synthesis not depth prediction, we can conveniently synthesize disparity maps from our MPIs, and use them to evaluate depth performance. Here we measure this using the iBims-1 benchmark [14], which has ground truth depth for a variety of indoor scenes. As in Niklaus *et al.*, we scale and bias depth predictions to minimize the (L^2) depth error before evaluation [22]. In Table 3, we compare performance with three depth-prediction methods: MegaDepth [18], Depth in the Wild [29] and the recent "3D Ken Burns Effect" system [22]. Of our models, the **full** version performs the best, at a level comparable to MegaDepth, despite that method's much heavier reliance on explicit depth supervision. As we would expect, removing depth supervision and/or scale invariance leads to worse performance. Our **nosmooth** model performs the worst, con-

Method	rel ↓	log10 ↓	RMS ↓	σ_1 ↑	σ_2 ↑	σ_3 ↑
DIW	0.25	0.10	1.00	0.61	0.86	0.95
MegaDepth (Mega)	0.23	0.09	0.83	0.67	0.89	0.96
MegaDepth (Mega + DIW)	0.20	0.08	0.78	0.70	0.91	0.97
3DKenBurns	**0.10**	**0.04**	**0.47**	**0.90**	**0.97**	**0.99**
Ours: **full**	0.21	0.08	0.85	0.70	0.91	0.97
nodepth ($\lambda_{\text{depth}} = 0$)	0.23	0.09	0.90	0.67	0.89	0.96
noscale ($\sigma = 1$)	0.23	0.09	0.89	0.65	0.89	0.97
nosmooth ($\lambda_{\text{smooth}} = 0$)	0.24	0.09	0.94	0.65	0.87	0.96
nobackground ($c_i = \mathbf{I}_s$)	0.22	0.09	0.90	0.67	0.90	0.97

Table 3. Measuring depth prediction quality with the iBims-1 benchmark [14]. While not state of the art in terms of depth prediction, our method is comparable to other systems that use explicit depth supervision, even when we use no depth supervision at all. We reran the MegaDepth model to ensure consistency; results for other methods are as reported by Niklaus *et al*. [22]. See Section 4.2.

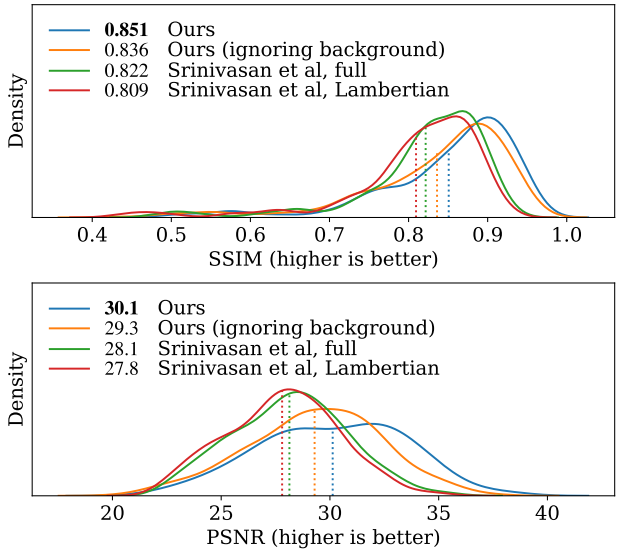

Figure 6. Results of running our method on the Flowers dataset, inputting a central view and synthesizing views from the four corner angles. We visualise the distribution of SSIM and PSNR metrics across the test set of 100 light fields (dotted lines and numbers in the legend show the mean values). See Section 4.3.

firming that our edge-aware smoothness loss is valuable in learning to predict MPIs that correspond to good depth-maps.

4.3. View synthesis on Flowers light fields

We now apply our method to other datasets. Srinivasan *et al*. introduced a dataset of light field photos of flowers, and a method for predicting the entire light field from a single image [28]. This dataset consists of over 3000 photographs, each one capturing a narrow-baseline 14 × 14 grid of light field angles. This dataset has no point cloud data for determining scale, so we cannot apply our scale-independent view synthesis approach. However, the scale is constant across the whole dataset so we can simply set $\sigma = 1$ and rely on the network to learn the appropriate scale. For this task, we add a gradient term to our synthesis loss. We train our model on the Flowers dataset by picking source and target images randomly from the 8 × 8 central square of light field angles, and evaluate the results on a held-out set of 100 light fields.

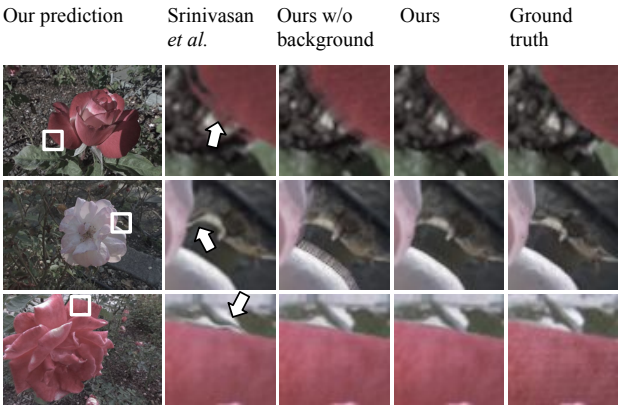

Figure 7. Comparisons on Flowers light fields. Views rendered by our method without using the predicted background show blurriness and repeated edge artefacts at depth boundaries, which the predicted background ameliorates. Our predictions improve on those of Srinivasan *et al*. by avoiding occasional large 'floating' foreground artefacts, and by reducing the distortion of background elements. See Section 4.3.

	PSNR ↑		SSIM ↑	
Method	all	disocc.	all	disocc.
Evaluated at 384 × 128 *pixels*				
Tulsiani *et al*. [1]	16.5	15.0	0.572	0.523
Ours ('low res')[2]	19.3	17.2	0.723	0.631
Ours (full)[3]	**19.5**	**17.5**	**0.733**	**0.639**
w/o background	19.3	16.9	0.731	0.627
Evaluated at 1240 × 375 *pixels*				
Ours (full)	**19.3**	**17.4**	**0.696**	**0.651**
w/o background	19.1	16.7	0.690	0.634

[1] Their method predicts layers at 768 × 256 resolution but renders at 384 × 128 to avoid cracks.
[2] For a fair comparison, our 'low res' model is trained to predict layers at 768 × 256.
[3] Our full method predicts layers at 1240 × 375.

Table 4. Evaluation on city sequences from the KITTI dataset. We compute PSNR and SSIM metrics over all pixels and over 'disoccluded' pixels only. The upper part of the table compares results at a rendering resolution of 384 × 128, for comparison with the model of Tulsiani *et al*. The lower part shows that our model also performs well when evaluated at full resolution. The 'w/o background' rows show the result of taking our MPI and ignoring the background image by replacing each layer's color with the input $\mathbf{I}_s$. Especially on disoccluded pixels, using the background image leads to a substantial improvement. See Section 4.4.

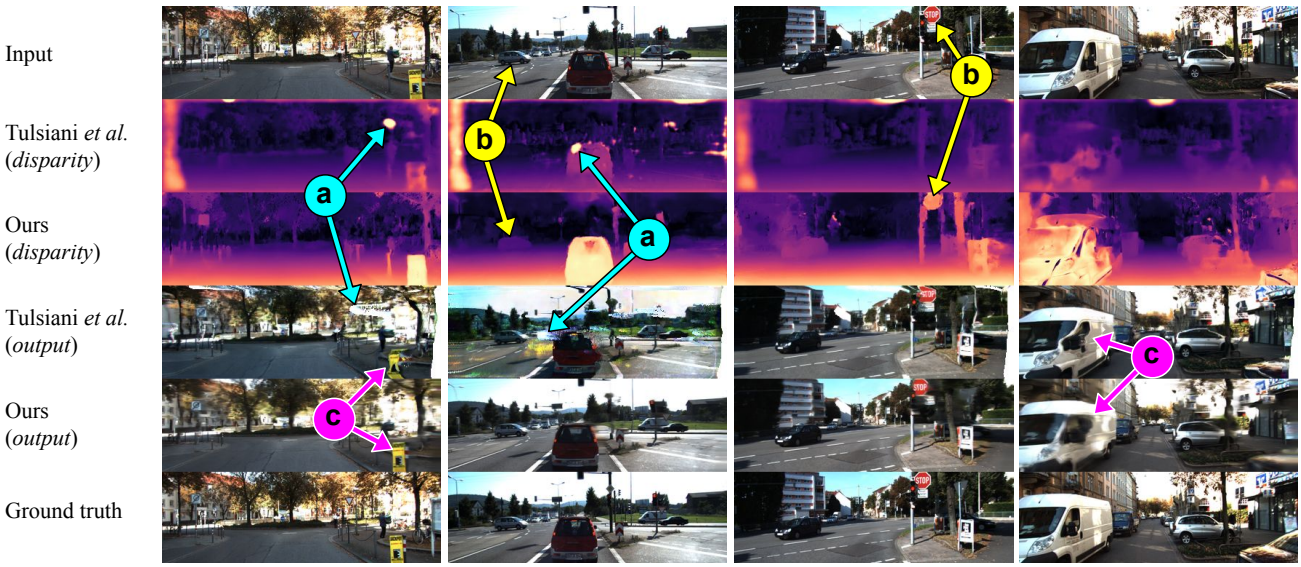

Figure 8. Comparison on KITTI city sequences. (a) Erroneous bright spots in Tulsiani *et al.*'s depth-maps lead to unpleasant visual artefacts. (b) Comparing depth maps shows structures (the car, the Stop sign) identified by our method but missing from theirs. (c) In challenging areas their method produces sharp but distorted output, ours tends to produce blurrier output. See Section 4.4.

For comparison, we retrained the model of Srinivasan *et al.* [28] using their publicly available code.

As shown in Fig. 6, our method improves on that of Srinivasan *et al.* Even if we eliminate the background image during testing (i.e. set color $c_i = \mathbf{I}_s$ throughout) we achieve higher PSNR and SSIM measures (along with lower absolute error) on the test set; using the predicted background image we see an additional small improvement. Our method has other advantages over theirs: we do not require complete light field data for training, and our representation can be rerendered at arbitrary novel viewpoints without further inference steps. Fig. 7 shows some qualitative comparisons.

4.4. View Synthesis on KITTI

Instead of sampling source and target viewpoints from a sequence or light field, we can also apply our model to data where only left-right stereo pairs are available, such as KITTI [8]. Tulsiani *et al.* showed the possibility of using view synthesis as a proxy task to learn a two-layer layered depth image (LDI) representation from such data [31]. We train our model on the same data using 22 of the *city* category sequences in the 'raw' KITTI dataset, randomly taking either the left or the right image as the source (the other being the target) at each training step. Because the cameras are fixed, the relative pose is always a translation left or right by about 0.5m. Again, the scale is constant so we set $\sigma = 1$, and again we add a gradient term to the synthesis loss. We compare our synthesized views on 1079 image pairs in the 4 test sequences with those produced by their pre-trained model.

The representation used by Tulsiani *et al.* is less capable of high-quality view synthesis than our MPIs, because lacking alpha it cannot model soft edges, and because its splat-based rendering generates low-resolution output to avoid cracks. Both methods exhibit many artefacts at the image edges, so we crop 5% of the image away at all sides, and then compute PSNR and SSIM metrics on all pixels, and also on 'disoccluded' pixels only (as estimated by a multi-view stereo algorithm). For a fair comparison with Tulsiani *et al.*, our 'low res' model matches theirs in resolution; we also train a 'full' model at a higher resolution. Both models achieve improvements over theirs. The effect of our predicted background is small over the whole image, but larger when we consider only disoccluded pixels. Results are shown in Table 4, and qualitative comparisons in Fig. 8.

5. Conclusion

We demonstrate the ability to predict MPIs for view synthesis from single image inputs without requiring ground truth 3D or depth, and we introduce a scale-invariant approach to view synthesis that allows us to train on data with scale ambiguity, such as that derived from online video. Our system is able to use the predicted background image to 'inpaint' what lies behind the edges of foreground objects even though there is no explicit inpainting step in our system—although we typically do not see inpainting of more than a few pixels at present. A possible future direction would be to pair MPI prediction with adversarial losses to see if more, and more realistic, inpainting can be achieved.

Acknowledgements. This work benefited from helpful discussions with Jon Barron, Tali Dekel, John Flynn, Graham Fyffe, Angjoo Kanazawa, Andrew Liu and Vincent Sitzmann.

References

[1] Gaurav Chaurasia, Sylvain Duchêne, Olga Sorkine-Hornung, and George Drettakis. Depth synthesis and local warps for plausible image-based navigation. *Trans. on Graphics*, 32:30:1–30:12, 2013. 2

[2] Weifeng Chen, Shengyi Qian, and Jia Deng. Learning single-image depth from videos using quality assessment networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[3] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NeurIPS*, 2014. 3, 4

[4] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1, 2

[5] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. DeepStereo: Learning to predict new views from the world's imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[6] Ed Gamble and Tomaso Poggio. Visual integration and detection of discontinuities: The key role of intensity edges. A.I. Memo 970, AI Lab, MIT, 1987. 4

[7] Ravi Garg, Vijay Kumar BG, Gustavo Carneiro, and Ian Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *ECCV*, 2016. 2

[8] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *IJRR*, 2013. 8

[9] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 4

[10] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *Proceedings of SIGGRAPH 96*, Annual Conference Series, 1996. 2

[11] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. In *ACM Transactions on Graphics (SIGGRAPH Asia)*, 2018. 2

[12] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Trans. Graph.*, 35(6):193:1–193:10, 2016. 2

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 5

[14] Tobias Koch, Lukas Liebel, Friedrich Fraundorfer, and Marco Körner. Evaluation of CNN-based single-image depth estimation methods. In Laura Leal-Taixé and Stefan Roth, editors, *European Conference on Computer Vision Workshop (ECCV-WS)*, pages 331–348. Springer International Publishing, 2018. 6, 7

[15] Katrin Lasinger, René Ranftl, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *arXiv preprint arXiv:1907.01341*, 2019. 2, 3

[16] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of SIGGRAPH 96*, Annual Conference Series, 1996. 2

[17] Zhengqi Li, Tali Dekel, Forrester Cole, Richard Tucker, Noah Snavely, Ce Liu, and William T. Freeman. Learning the depths of moving people by watching frozen people. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 4

[18] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 6

[19] Miaomiao Liu, Xuming He, and Mathieu Salzmann. Geometry-aware deep network for single-image novel view synthesis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[20] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. arXiv:1512.02134. 4

[21] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019. 2

[22] Simon Niklaus, Long Mai, Jimei Yang, and Feng Liu. 3D Ken Burns effect from a single image. *ACM Transactions on Graphics (TOG)*, 2019. 2, 6, 7

[23] Eric Penner and Li Zhang. Soft 3D reconstruction for view synthesis. *ACM Trans. Graph.*, 36(6):235:1–235:11, 2017. 2

[24] Thomas Porter and Tom Duff. Compositing digital images. *SIGGRAPH Comput. Graph.*, 18(3):253–259, 1984. 3

[25] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42, 2002. 4

[26] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *Proceedings of SIGGRAPH 98*, 1998. 2

[27] Pratul P. Srinivasan, Richard Tucker, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[28] Pratul P. Srinivasan, Tongzhou Wang, Ashwin Sreelal, Ravi Ramamoorthi, and Ren Ng. Learning to synthesize a 4D RGBD light field from a single image. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2, 7, 8

[29] Lech Świrski, Christian Richardt, and Neil A. Dodgson. Layered photo pop-up. In *SIGGRAPH Posters*, Aug. 2011. Winner of the ACM SIGGRAPH Student Research Competition. 6

[30] Richard Szeliski and Polina Golland. Stereo matching with transparency and matting. *Int. J. Comput. Vision*, 32(1):45–61, Aug. 1999. 2, 3

[31] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3D scene inference via view synthesis. In *The European Conference on Computer Vision (ECCV)*, September 2018. 2, 8

[32] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 3

[33] Chaoyang Wang, José Miguel Buenaposada, Rui Zhu, and Simon Lucey. Learning depth from monocular videos using direct methods. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 4

[34] Junyuan Xie, Ross Girshick, and Ali Farhadi. Deep3D: Fully automatic 2D-to-3D video conversion with deep convolutional neural networks. In *European Conference on Computer Vision*, pages 842–857. Springer, 2016. 2

[35] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 5

[36] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[37] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Trans. Graph.*, 37(4):65:1–65:12, 2018. 1, 2, 3, 5

[38] C. Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. *ACM Trans. Graph.*, 23(3):600–608, 2004. 2