

# X-Fields: Implicit Neural View-, Light- and Time-Image Interpolation

MOJTABA BEMANA, MPI Informatik, Saarland Informatics Campus  
KAROL MYSZKOWSKI, MPI Informatik, Saarland Informatics Campus  
HANS-PETER SEIDEL, MPI Informatik, Saarland Informatics Campus  
TOBIAS RITSCHEL, University College London

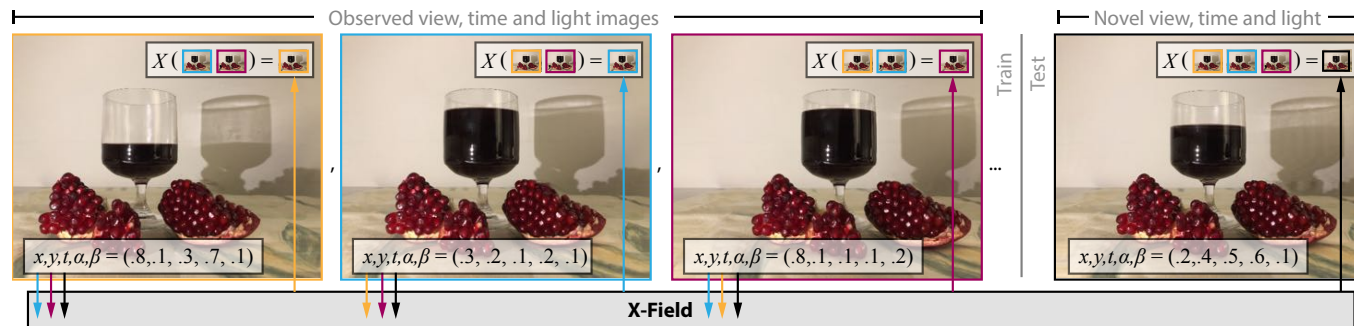


Fig. 1. To interpolate view, light and time in a set of 2D images labeled with coordinates (X-Field), we train a neural network (NN) to regress each image from all others. The first (yellow) image is the NN output (yellow up arrow) when the blue and purple observed images and their coordinates  $x, y, t, \alpha, \beta$  are input (yellow down arrows). The blue and purple observations form additional constraints, visualized as colored boxes. Provided with an unobserved coordinate (black up arrow) the NN produces, from the observed images and coordinates (black down arrow), a novel high-quality 2D image in real time.

We suggest to represent an X-Field—a set of 2D images taken across different view, time or illumination conditions, i.e., video, lightfield, reflectance fields or combinations thereof—by learning a neural network (NN) to map their view, time or light coordinates to 2D images. Executing this NN at new coordinates results in joint view, time or light interpolation. The key idea to make this workable is a NN that already knows the “basic tricks” of graphics (lighting, 3D projection, occlusion) in a hard-coded and differentiable form. The NN represents the input to that rendering as an implicit map, that for any view, time, or light coordinate and for any pixel can quantify how it will move if view, time or light coordinates change (Jacobian of pixel position with respect to view, time, illumination, etc.). Our X-Field representation is trained for one scene within minutes, leading to a compact set of trainable parameters and hence real-time navigation in view, time and illumination.

CCS Concepts: • **Computing methodologies** → **Neural networks**.

Additional Key Words and Phrases: View interpolation, Light interpolation, Time interpolation, Deep learning

## ACM Reference Format:

Mojtaba Bemana, Karol Myszkowski, Hans-Peter Seidel, and Tobias Ritschel. 2020. X-Fields: Implicit Neural View-, Light- and Time-Image Interpolation. *ACM Trans. Graph.* 39, 6, Article 257 (December 2020), 15 pages. <https://doi.org/10.1145/3414685.3417827>

Authors’ addresses: Mojtaba Bemana, MPI Informatik, Saarland Informatics Campus, mbemana@mpi-inf.mpg.de; Karol Myszkowski, MPI Informatik, Saarland Informatics Campus; Hans-Peter Seidel, MPI Informatik, Saarland Informatics Campus; Tobias Ritschel, University College London.



This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2020 Copyright held by the owner/author(s).  
0730-0301/2020/12-ART257  
<https://doi.org/10.1145/3414685.3417827>

## 1 INTRODUCTION

Current and future sensors capture images of one scene from different points (video), from different angles (light fields), under varying illumination (reflectance fields) or subject to many other possible changes. In theory, this information will allow exploring time, view or light changes in Virtual Reality (VR). Regrettably, in practice, sampling this data densely leads to excessive storage, capture and processing requirements. In higher dimensions—here we demonstrate 5D—the demands of dense regular sampling (cubature) increase exponentially. Alternatively, sparse and irregular sampling overcomes these limitations, but requires faithful interpolation across time, view and light. We suggest taking an abstract view on all those dimensions and simply denote any set of images conditioned on parameters as an “X-Field”, where X could stand for any combination of time, view, light or other dimensions like the color spectrum. We will demonstrate how the right neural network (NN) becomes a universal, compact and interpolatable X-Field representation.

While NNs have been suggested to estimate depth or correspondence across space, time or light, we here, for the first time, suggest representing the complete X-Field implicitly [Chen and Zhang 2019; Niemeyer et al. 2019; Oechsle et al. 2019; Sitzmann et al. 2019b], i.e., as a trainable architecture that implements a high-dimensional `getPixel`. The main idea is shown in Fig. 1: from sparse image observations with varying conditions and coordinates, we train a mapping that, when provided the space, time or light coordinate as an input, generates the observed sample image as an output. Importantly, when given a non-observed coordinate, the output is a faithfully interpolated image. Key to making this work is the right training and a suitable network structure, involving a very primitive (but differentiable) rendering (projection and lighting) step.

Our architecture is trained for one specific X-Field to generalize across its parameters, but not across scenes. However, per-scene training is fast (minutes), and decoding occurs at high frame rates (ca. 20 Hz) and high resolution ( $1024 \times 1024$ ). In a typical use case of the VR exploration of an X-Field, the architecture parameters only require a few additional kilobytes on top of the image samples. We compare the resulting quality to several other state-of-the-art interpolation baselines (NN and classic, specific to certain domains and general) as well as to ablations of our approach.

Our neural network implementation and training data is publicly available at <http://xfields.mpi-inf.mpg.de/>.

## 2 PREVIOUS WORK

Here we review previous techniques to interpolate across discrete sampled observations in view (light fields), time (video) and illumination (reflectance fields). Tbl. 1 summarizes this body of work along multiple axes. Tewari et al. [2020] provide further report of the state-of-the-art in Neural Rendering.

### 2.1 View Interpolation (Light Fields)

Levoy and Hanrahan [1996] as well as Gortler et al. [1996] were first to formalize the concept of a light field (LF)—the set of all images of a scene for all views—and to devise hardware to capture it. LF methods come first in Tbl. 1, where they are checked “view” as they generalize across an observer’s position and orientation. A simple solution for all interpolation, including view, is linear blending, but this leads to ghosting.

An important distinction is that a capture can be *dense* or *sparse*, denoted as “Sparse” in Tbl. 1. Sparsity depends less on the number of images, but more on the difference between captured images. Very similar view positions [Kalantari et al. 2016] as for a Lytro camera can be considered dense, while 34 views on a sphere [Lombardi et al. 2019] or 40 lights on a hemisphere [Malzbender et al. 2001] is sparse. In this paper we focus on wider baselines, with typically  $M \times N$  cameras spaced by 5–10 cm [Flynn et al. 2019], and respectively a large disparity ranging up to 250 pixels [Dabala et al. 2016; Mildenhall et al. 2019], where  $M$  and  $N$  are single-digit numbers, e.g.,  $3 \times 3$ ,  $5 \times 5$  or even  $2 \times 1$ . Depending on resources, a capture setup can be considered simple (cell phone, as we use) or more involved (light stage) as denoted in the “easy capture” column in Tbl. 1.

Early view interpolation solutions, such as Unstructured Lumigraph Rendering (ULR) [Buehler et al. 2001; Chaurasia et al. 2013], typically create proxy geometry to warp [Mark et al. 1997] multiple observations into a novel view and blend them with specific weights. More recent work has used per-view geometry [Hedman et al. 2016] and learned ULR blending weights [Hedman et al. 2018], to allow sparse input and view-dependent shading. Avoiding the difficulty of reconstructing geometry or 3D volumes has been addressed for LFs in [Du et al. 2014; Kellnhofer et al. 2017].

An attractive recent idea is to learn synthesizing novel views for LF data. Kalantari et al. [2016] indirectly learn depth maps without depth supervision to interpolate between views in a Lytro camera. Another option is to decompose LFs into multiple depth planes of the output view and construct a view-dependent plane sweep volume (PSV) [Flynn et al. 2016]. By learning how neighboring

input views contribute to the output view, the multi-plane image (MPI) representation [Zhou et al. 2018] can be built, which enables high-quality local LF fusion [Mildenhall et al. 2019].

Instead of using proxy geometry, Penner and Zhang [2017] have suggested using a volumetric occupancy representation. Inferring a good volumetric / MPI representation can be facilitated with learned gradient descent [Flynn et al. 2019], where the gradient components directly encode visibility and effectively inform the NN on the occlusion relations in the scene. MPI techniques avoid the problem of explicit depth reconstruction and allow for softer, more pleasant results. A drawback in deployment is the massive volumetric data, the difficulty of distributing occupancy therein, and finally bandwidth requirements of volume rendering itself. Our approach involves a learning route as well, but explaining the entire X-Field and using a NN to represent the scene implicitly. Deployment only requires a few additional kilobytes of NN parameters on top of the input image data, and rendering is real-time.

From yet another angle, ULR-inspired IBR creates a LF (view-dependent appearance) on the surface of a proxy geometry, i.e., a *surface light field*. Chen et al. [2018], using an MLP, as well as Thies et al. [2019], using a CNN, have proposed to represent this information using a NN defined in texture space of a proxy object. While inspired by the mechanics of sparse IBR, results are typically demonstrated for rather dense observations. Our approach does not assume any proxy to be given, but jointly represents the appearance and the geometry used to warp over many dimensions, in a single NN, trained from sparse sets of images.

Another step of abstraction is Deep Voxels [Sitzmann et al. 2019a]. Instead of storing opacity and appearance in a volume, abstract “persistent” features are derived, which are projected using learned ray-marching. The volume was learned per-scene. In this work we also use implicit functions instead of voxels, as did Sitzmann et al. [2019b], Saito et al. [2019] and Mildenhall et al. [2020]. We call such approaches “implicit” in Tbl. 1 when the NN replaces the pixel basis, i.e., the network provides a high-dimensional `getPixel(x)`. These approaches use an MLP that can be queried for occupancy [Chen and Zhang 2019; Saito et al. 2019; Sitzmann et al. 2019b], color [Mildenhall et al. 2020; Oechsle et al. 2019; Sitzmann et al. 2019b], flow [Niemeyer et al. 2019] etc. at different 3D positions along a ray for one pixel. We make two changes to this design. First, we predict texture coordinates, rather than appearance. These drive a spatial transformer [Jaderberg et al. 2015] that can copy details from the input images without representing them and do so at high speed (20 fps). Second, we train a 2D CNN instead of a 3D MLP that, for a given X-Field coordinate, will directly return a complete 2D per-pixel depth and correspondence map. For an X-Field problem, this is more efficient than ray-marching and evaluating a complex MLP at every step [Mildenhall et al. 2020; Niemeyer et al. 2019; Oechsle et al. 2019; Sitzmann et al. 2019b]. While implicit representations have so far been demonstrated to provide a certain level of fidelity when generalized across a class of simpler shapes (cars, chairs, etc.), we here make the task simpler and generalize less, but produce quality to compete with state-of-the-art view, time and light interpolation methods from computer graphics.

Inspired by Nguyen Phuoc et al. [2018], some work [Nguyen Phuoc et al. 2019; Sitzmann et al. 2019a,b] learns the differentiable

Table 1. Comparison of space, time, and illumination interpolation methods (rows) in respect to capabilities (columns), with an emphasis on deep methods. (<sup>1-3,5</sup>Similar-class scenes demonstrated, e.g., cars, chairs, urban city views; <sup>4</sup>LF sparse in time; <sup>6</sup>Clothed humans demonstrated; <sup>7</sup>Human faces shown; <sup>8</sup>Only structured grids shown. Should support unstructured, as long as the transformation between views are known.)

Method	Citation	Scene	View	Time	Light	Sparse	Unstructured	Real-time	Easy capture	Learned	Implicit	Diff. render.	Remarks
		Generalize	Interface	Implem.									
Unstructured Lumigraph	Buehler et al. [2001]	✓	✓	×	×	✓	✓	×	✓	×	×	×	IBR
Soft 3D	Penner and Zhang [2017]	✓	✓	×	×	✓	✓	×	✓	×	×	×	MPI
Inside Out	Hedman et al. [2016]	✓	✓	×	×	✓	✓	✓	✓	×	×	×	IBR; SfM; Per-view geometry
Deep Blending	Hedman et al. [2018]	✓	✓	×	×	✓	✓	✓	✓	✓	×	×	IBR; SfM; Learned fusion
Learning-based View Interp.	Kalantari et al. [2016]	✓	✓	×	×	×	×	×	✓	✓	×	×	Lytro; Learned disparity and fusion
Local LF Fusion	Mildenhall et al. [2019]	✓	✓	×	×	✓	✓	✓	✓	✓	×	×	MPI
DeepView	Flynn et al. [2019]	×	✓	×	×	✓	✓	×	×	×	×	×	MPI
Deep Surface LFs	Chen et al. [2018]	×	✓	×	×	×	×	✓	×	✓	×	×	Texture; Lumitexel; MLPs
NeRF	Mildenhall et al. [2020]	×	✓	×	×	×	✓	×	×	✓	✓	✓	MLPs, ray-marching
Neural Textures	Thies et al. [2019]	×	✓	×	×	×	×	✓	×	✓	✓	✓	Texture; Lumitexel; CNNs
DeepVoxels	Sitzmann et al. [2019a]	×	✓	×	×	×	✓	×	×	✓	✓	✓	3D CNN
HoloGAN	Nguyen Phuoc et al. [2019]	✓	✓	×	×	×	×	×	✓	✓	×	✓	Adversarial; 3D representation
Appearance Flow	Zhou et al. [2016]	✓ <sup>1</sup>	✓	×	×	×	×	×	×	×	✓	✓	App. Flow; Fixed views
Multi-view App. Flow	Sun et al. [2018a]	✓ <sup>2</sup>	✓	×	×	×	×	×	×	×	✓	✓	App. Flow; Learned fusion; Fixed views
Spatial Trans. Net IBR	Chen et al. [2019]	✓ <sup>3</sup>	✓	×	×	×	✓	✓	×	✓	✓	✓	App. Flow; Per-view geometry; Free views
Moving Gradients	Mahajan et al. [2009]	✓	×	✓	×	×	×	×	✓	×	×	×	Gradient domain
Super SlowMo	Jiang et al. [2018]	✓	×	✓	×	✓	×	×	✓	✓	×	×	Occlusions: Learns visibility maps
MEMC-Net	Bao et al. [2019b]	✓	×	✓	×	✓	×	×	✓	✓	×	×	Occlusions: Learns visibility maps
Depth-aware Frame Int.	Bao et al. [2019a]	✓	×	✓	×	✓	×	×	✓	✓	×	×	Occlusions: Learns depth maps
Video-to-video	Wang et al. [2018]	✓	×	✓	×	×	×	×	✓	✓	×	×	Adversarial; Segmented content editing
Puppet Dubbing	Fried and Agrawala [2019]	×	×	✓	×	×	×	✓	✓	✓	✓	✓	Visual and sound sync.
Layered Representation	Zitnick et al. [2004]	✓	✓	✓	×	✓	✓	✓	×	×	×	×	MVS reconst.; Layered Depth Images
Video Array	Wilburn et al. [2005]	✓	✓	✓	×	×	×	×	×	×	×	×	Optical flow
Virtual Video	Lipski et al. [2010]	✓	✓	✓	×	✓	✓	✓	✓	×	×	×	Structure from Motion (SfM)
Hybrid Imaging	Wang et al. [2017]	✓	✓	✓	×	✓ <sup>4</sup>	×	✓	✓	✓	×	×	Lytro+DSLR camera system
Neural Volumes	Lombardi et al. [2019]	×	✓	✓	×	×	×	×	×	✓	✓	✓	3D CNN; Lightstage; Fixed time (video)
Scene Represent. Net	Sitzmann et al. [2019b]	✓ <sup>5</sup>	✓	✓	×	×	×	×	×	✓	✓	✓	3D MLP
Pixel-aligned Implicit Funct.	Saito et al. [2019]	✓ <sup>6</sup>	✓	✓	×	✓	✓	×	✓	✓	✓	✓	3D MLP
Polynomial Textures	Malzbender et al. [2001]	×	×	✓	✓	×	×	✓	×	×	✓	×	Lightstage
Neural Relighting	Ren et al. [2015]	×	×	×	✓	×	✓	×	×	✓	✓	×	MLP; Lightstage and hand-held lighting
Sparse Sample Relighting	Xu et al. [2018]	✓	×	×	✓	✓	✓	✓	×	✓	×	×	Optimized light positions
Sparse Sample View Synth.	Xu et al. [2019]	✓	✓	×	✓	✓	✓	✓	×	✓	×	×	Optimized lights as in [Xu et al. 2018]
Multi-view Relighting	Philip et al. [2019]	✓	✓	×	✓	✓	✓	✓	✓	✓	×	×	Geometry proxy; Auxiliary 2D buffers
Deep Reflectance Fields	Meka et al. [2019]	✓ <sup>7</sup>	✓	✓	✓	×	×	×	×	✓	✓	×	Lightstage
The Relightables	Guo et al. [2019]	✓	✓	✓	✓	×	×	✓	×	✓	×	×	Lightstage
Ours		×	✓	✓	✓	✓	×	✓ <sup>8</sup>	✓	✓	✓	✓	

tomographic rendering step, while other work has shown how it can be differentiated directly [Henzler et al. 2019; Lombardi et al. 2019; Mildenhall et al. 2020]. Our approach avoids tomography and works with differentiable warping [Jaderberg et al. 2015] with consistency handling inspired by unsupervised depth reconstruction [Godard

et al. 2017; Zhou et al. 2017]. Avoiding volumetric representations allows for real-time playback, while at the same generalizing from view to other dimensions such as time and light.

The Appearance Flow work of Zhou et al. [2016] suggests to combine the idea of warping pixels with learning how to warp.

While Zhou et al. [2016] typically consider a single input view, Sun et al. [2018a] employ multiple views to improve warped view quality. Both works use an implicit representation of the warp field, i.e., a NN that for every pixel in one view predicts from where to copy its value in the new view. While those techniques worked best for fixed camera positions that are used in training, Chen et al. [2019] introduces an implicit NN of per-pixel depth that enables an arbitrary view interpolation. All these methods require an extensive training for specific classes of scenes such as cars, chairs, or urban city views. We take this line of work further by constructing an implicit NN representation that generalizes jointly over complete geometry, motion, and illumination changes. Our task on the one hand is simpler, as we do not generalize across different scenes, yet on the other hand it is also harder, as we generalize across many more dimensions and provide state-of-the-art visual quality.

## 2.2 Time (Video)

Videos comprise discrete observations, and hence are also a sparse capture of the visual world. To get smooth interpolation, e.g., for slow-motion (individual frames), motion blur (averaging multiple frames) images need to be interpolated [Mahajan et al. 2009], potentially using NNs [Bao et al. 2019a,b; Jiang et al. 2018; Sun et al. 2018b; Wang et al. 2018]. More exotic domains of video re-timing, which involve annotation of a fraction of frames and one-off NN training, include the visual aspect in sync with spoken language [Fried and Agrawala 2019].

## 2.3 Space-Time

Warping can be applied to space or time, as well as to both jointly [Manning and Dyer 1999], resulting in LF video [Lipski et al. 2010; Wang and Yang 2005; Wang et al. 2017; Zitnick et al. 2004].

Recent work has extended deep novel-view methods into the time domain [Lombardi et al. 2019], and is closest to our approach. They also use warping, but for a very different purpose: deforming a pixel-basis 3D representation over time in order to avoid storing individual frames (motion compensation). Both methods Sitzmann et al. [2019a] and Lombardi et al. [2019] are limited by the spatial 3D resolution of volume texture and the need to process it, while we work in 2D depth and color maps only. As they learn the tomographic operator, this limit in resolution is not a classic Nyquist limit, e.g., sharp edges can be handled, but results typically are on isolated, dominantly convex objects, while we target entire scenes. Ultimately, we do not claim depth maps to be superior to volumes per se. Instead, we suggest that 3D volumes have their strength for seeing objects from all views (at the expense of resolution), whereas our work, using images, is more for observing scenes from a “funnel” of views, but at high 2D resolution. No work yet is able to combine high resolution and arbitrary views, not to mention time.

## 2.4 Light Interpolation (Reflectance Fields)

While a LF is specific to one illumination, a *reflectance field* (RF) [Debevec et al. 2000] is a generalization additionally allowing for relighting, often just for a fixed view. Dense sampling for individually controlled directional lights can be performed using Lightstage [Debevec et al. 2000], which leads to hundreds of captured images.

The number of images can be reduced by employing specially designed illumination patterns [Fuchs et al. 2007; Peers et al. 2009; Reddy et al. 2012] to exploit various forms of coherence in the light transport function. Our capture is from uncalibrated sets of flash images of mobile phones. For interpolation, the signal is frequently separated, such as into highlights, reflectance or shadows [Chen and Lensch 2005]. We also found such a separation to help. Angular coherence in incoming lighting leads to an efficient reflectance field representation as polynomial texture maps [Malzbender et al. 2001], which can be further improved by neural networks whose expressive power enables one to capture non-linear spatial coherence [Ren et al. 2015], or generalize across views [Maximov et al. 2019]. Xu et al. [2018] directly regresses images of illumination from an arbitrary light direction when given five images from specific other light directions. The innovation is in optimizing what should be input at test time, but the setup requires custom capture dome equipment, as well as input images taken from those five, very specific, directions. For scenes captured under controlled illumination for multiple sparse views, generalization across views can be achieved by concatenation with a view synthesis method [Xu et al. 2019]. While the results are compelling on synthetic scenes, the method exhibits difficulties in handling complex or non-convex geometry, as well as high frequency details such as specularities and shadows [Meka et al. 2019]. An approximate geometry proxy and extensive training over rendered scenes might compensate for inaccuracies in derived shadows and overall relighting quality [Philip et al. 2019].

Specialized systems for relighting human faces and characters remove many such limitations, including fixed view and static scene assumptions, using advanced Lightstage hardware that enables capturing massive data for CNN training [Meka et al. 2019] and complex optimizations that are additionally fed with multiple depth sensors’ data [Guo et al. 2019]. As only two images for an arbitrary face or character under spherical color gradient lighting are required at the test time, real-time dynamic performance capturing is possible. CNN-based, LF-style view interpolation is performed in Meka et al. [2019], whereas Guo et al. [2019] capture complete 3D models with textures and can easily change viewpoint as well.

Meka et al. [2019] and Guo et al. [2019] generalize over similar scenes (faces) while our approach is fixed to one specific scene. On the other hand, we remove the requirements for massive training data and costly capturing hardware, while our lightweight network enables real-time rendering of animated scenes under interpolated dynamic lighting and view position.

## 3 BACKGROUND

Two main observations motivate our approach: First, representing information using NNs leads to interpolation. Second, this property is retained, if the network contains more useful layers, such as a differentiable rendering step. Both will be discussed next:

*Deep representations help interpolation.* It is well-known that deep representations suit interpolation of 2D images [Radford et al. 2015; Reed et al. 2015; White 2016], audio [Engel et al. 2017] or 3D shape [Dosovitskiy et al. 2015] much better than the pixel basis.

Consider the blue and orange bumps in Fig. 2, a; these are observed. They represent flat-land functions of appearance (vertical

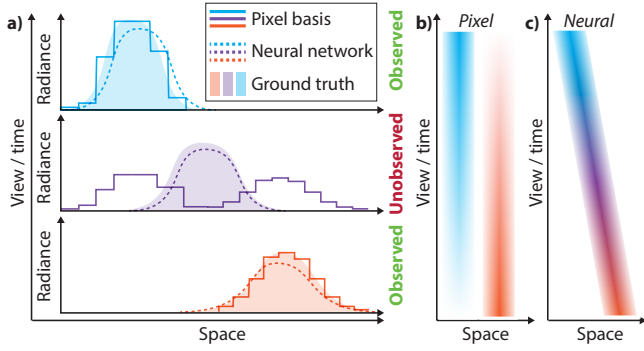


Fig. 2. NN and pixel interpolation: **a)** Flatland interpolation in the pixel (lines) and the NN representations (dotted lines) compared to a reference (solid) for a 1D field (vertical axis angle; horizontal axis space). The top and bottom are observed and the middle is unobserved, i.e., interpolated. **b,c)** Comparing the continuous interpolation in the pixel and the NN representation visualized as a (generalized) epi-polar image. Note that the NN leads to smooth interpolation, while the pixel representation causes undesired fade-in/fade-out transitions.

axis), depending on some abstract domain (horizontal axis), that later will become space, time, reflectance etc. in an X-Field. We wish to interpolate something similar to the unobserved violet bump in the middle. Linear interpolation in the pixel basis (solid lines) will fade both in, resulting in two flat copies. Visually this would be unappealing and distracting ghosting. This difference is also seen in the continuous setting of Fig. 2, b that can be compared to the reference in Fig. 2, c. When representing the bumps as NNs to map coordinates to color (dotted lines), we note that they are slightly worse than the pixel basis and might not match the NNs. However, the interpolated, unobserved result is much closer to the reference, and this is what matters in X-Field interpolation.

To benefit from interpolation, typically, substantial effort is made to construct latent codes from images, such as auto-encoders [Hinton and Salakhutdinov 2006], variational auto-encoders [Kingma and Welling 2013] or adversarial networks [Goodfellow et al. 2014]. We make the simple observation that this step is not required in the common graphics task of image (generalized) interpolation. In our problem we already have the latent space given as beautifully laid-out space-time X-Field coordinates and only need to learn to decode these into images.

*(Differentiable) rendering is just another non-linearity.* The second key insight is that the above property holds for any architecture as long as all units are differentiable. In particular, this allows for a primitive form of rendering (projection, shading and occlusion units). These units do not even have learnable parameters. Their purpose instead is to free the NN from learning basic concepts like occlusion, perspective, etc.

Figure 2 shows interpolation of colors over space. Consider regression of appearance using a multi-layer perceptron (MLP) [Chen and Zhang 2019; Oechsle et al. 2019; Sitzmann et al. 2019b] or convolutional neural network (CNN). CNNs without the coord-conv trick [Liu et al. 2018] are particularly weak at such spatially-conditioned

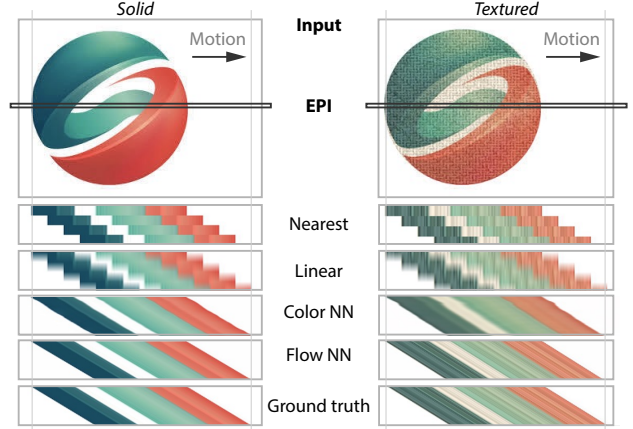


Fig. 3. Validation experiment: Different interpolation (rows), for two variants (columns) of a right-moving SIGGRAPH Asia 2020 logo (a 1D X-Field). For each method we show the same epipolar slice (i.e., space on the horizontal axis; time on the vertical axis) marked in the input image. Nearest and linear sampling show either blur or step artifacts. A NN to interpolate solid color depending on time succeeds, but lacks capacity to reproduce textured details, where the fine diagonal stripes are missing. A NN to interpolate flow instead, also captures the textured stripes.

generation. But even with coord-conv, this complex function is unnecessarily hard to find and slow to fit.

In contrast, methods that sample the observations using warping [Jaderberg et al. 2015] are much more effective to change the view [Zhou et al. 2016]. Figure 3 shows a validation experiment, that compares classic pixel-basis interpolation and neural interpolation of color and warping. Using a NN provides smooth epipolar lines, using warping, adds the details. We will now detail our work, motivated by those observations.

## 4 OUR APPROACH

We will first give a definition of the function we learn, followed by the architecture we choose for implementing it.

### 4.1 Objective

We represent the X-Field as a non-linear function:

$$L_{\text{out}}^{(\theta)}(\mathbf{x}) \in \mathcal{X} \rightarrow \mathbb{R}^{3 \times n_p},$$

with trainable parameters  $\theta$  to map from an  $n_d$ -dimensional X-Field coordinate  $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^{n_d}$  to 2D RGB images with  $n_p$  pixels. The X-Field dimension depends on the capture modality: A 4D example would be two spatial coordinates, one temporal dimension and one light angle. Parametrization can also be as simple as scalar 1D time for video interpolation. The symbol  $L_{\text{out}}$  is chosen as images are in units of radiance with a subscript to denote them as output.

We denote as  $\mathcal{Y} \subset \mathcal{X}$  the subset of *observed* X-Field coordinates for which an image  $L_{\text{in}}(\mathbf{y})$  was captured at the known coordinate  $\mathbf{y}$ . Typically  $|\mathcal{Y}|$  is sparse, i.e., small, like  $3 \times 3$ ,  $5 \times 5$  for view changes or even  $2 \times 1$  for stereo magnification. We find this mapping  $L_{\text{out}}$  by

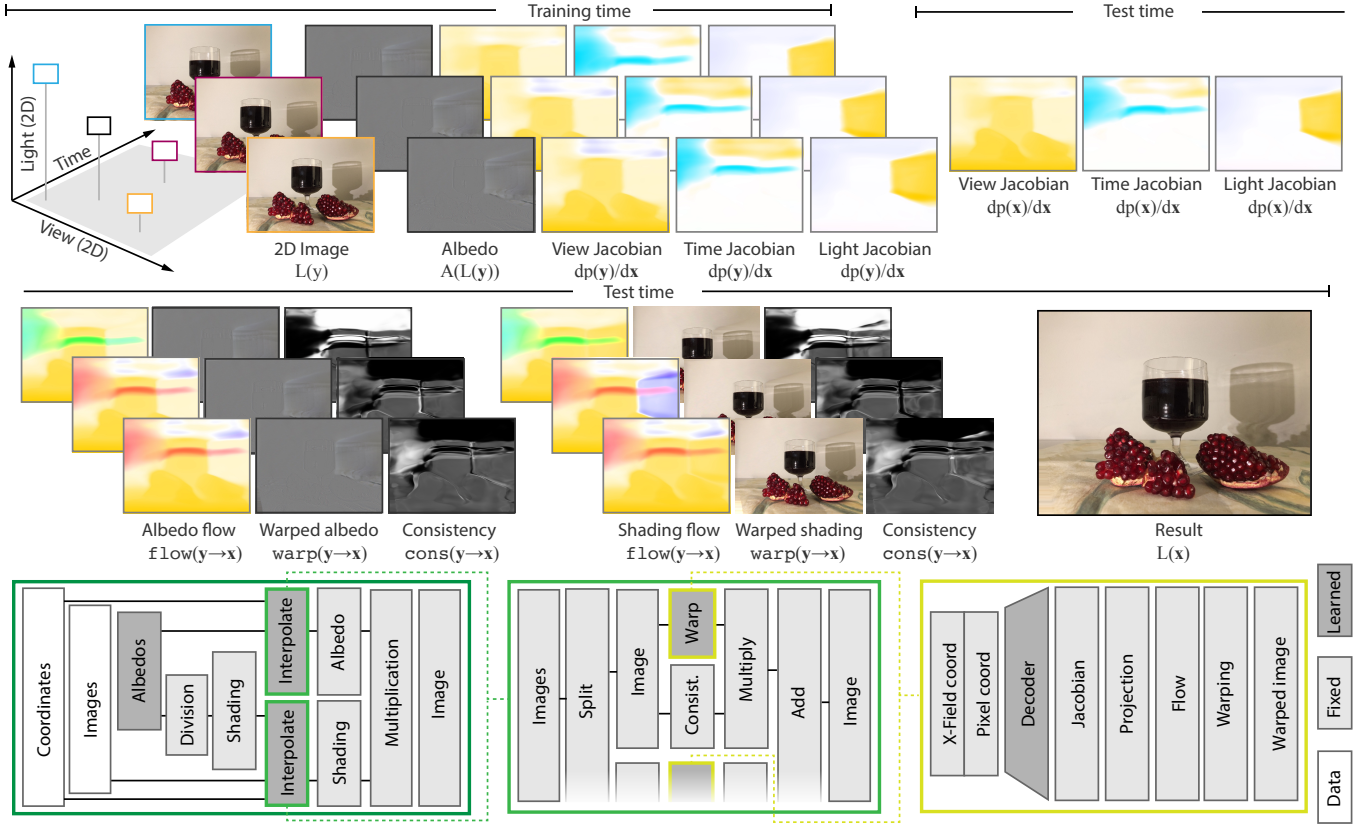


Fig. 4. Data flow for an example with three dimensions (one view, one light, one temporal) and three samples, denoted as colors, as in Fig. 1 and stacked vertically in each column. In the first row, the  $2 \times 3$  Jacobian matrix is always visualized as separate channels i.e., as three columns with two dimensions each. Values are 2D-vectors, hence visualized as false colors. At test time, the Jacobians are evaluated at the output X-Field coordinate only; hence, only a single row is shown. In the second row, each observation is separately warped for shading and albedo, leading to  $2 \times 3$  flow, result and weight images. The last row shows the flow of information as a diagram. *Learned* is a tunable, *Fixed* a non-tunable step (i.e., without learnable parameters). *Data* denotes access to inputs.

optimizing for

$$\theta = \arg \min_{\theta'} \mathbb{E}_{\mathbf{y} \sim \mathcal{Y}} \|L_{\text{out}}^{(\theta')}(\mathbf{y}) - L_{\text{in}}(\mathbf{y})\|_1,$$

where  $\mathbb{E}_{\mathbf{y} \sim \mathcal{Y}}$  is the expected value across all the discrete and sparse X-Field coordinates  $\mathcal{Y}$ . In prose, we train an architecture  $L_{\text{out}}$  to map vectors  $\mathbf{y}$  to captured images  $L_{\text{in}}(\mathbf{y})$  in the hope of also getting plausible images  $L_{\text{out}}(\mathbf{x})$  for unobserved vectors  $\mathbf{x}$ . We aim for interpolation;  $\mathcal{X}$  is a convex combination of  $\mathcal{Y}$  and does not extend beyond.

Note that training never evaluates any X-Field coordinate  $\mathbf{x}$  that is not in  $\mathcal{Y}$ , as we would not know what the image  $L_{\text{in}}(\mathbf{x})$  at that coordinate would be.

## 4.2 Architecture

We model  $L_{\text{out}}$  using three main ideas. First, appearance is a combination of appearance in observed images. Second, appearance is assumed to be a product of shading and albedo. Third, we assume the unobserved shading and albedo at  $\mathbf{x}$  to be a warped version of the observed shading and albedo at  $\mathbf{y}$ . These assumptions do not strictly need to hold, in particular not for splitting albedo and

shading: when they are not fulfilled, the NN just has a harder time capturing the relationship of coordinates and images.

Our pipeline  $L_{\text{out}}$ , depicted in Fig. 4, implements this in four steps: decoupling shading and albedo (Sec. 4.2.1), interpolating images (Sec. 4.2.2) as a weighted combination of warped images (Sec. 4.2.3), representing flow using a NN (Sec. 4.2.4) and resolving inconsistencies (Sec. 4.2.5).

**4.2.1 De-light.** De-lighting splits appearance into a combination of shading, which moves in one way in response to changes in X-Field coordinates, e.g., highlights move in response to view changes or shadows move with respect to light changes, and albedo, which is attached to the surface and will move with geometry, i.e., textures.

To this end, every observed image is decomposed as  $L_{\text{in}}(\mathbf{y}) = E(\mathbf{y}) \odot A(\mathbf{y})$ , a per-pixel (Hadamard) product  $\odot$  of a shading image  $E$  and an albedo image  $A$ . This is done by adding one parameter to  $\theta$  for every observed pixel channel in  $E$ , and computing  $A$  from  $L_{\text{in}}$  by division as  $E(\mathbf{y}) = L_{\text{in}}(\mathbf{y}) \odot A(\mathbf{y})^{-1}$ . Both shading and albedo are interpolated independently:

$$L_{\text{out}}(\mathbf{x}) = \text{int}(A(L_{\text{in}}(\mathbf{y})), \mathbf{y} \rightarrow \mathbf{x}) \odot \text{int}(E(L_{\text{in}}(\mathbf{y})), \mathbf{y} \rightarrow \mathbf{x}) \quad (1)$$

and recombined into new radiance at an unobserved location  $\mathbf{x}$  by multiplication. We will detail the operator `int`, working the same way on both shading  $E(L_{in})$  and albedo  $A(L_{in})$ , next.

**4.2.2 Interpolation.** Interpolation warps all observed images and merges the individual results. Both warp and merge are performed completely identically for shading  $E$  and albedo  $A$ , which we neutrally denote  $I$ , as in:

$$\text{int}(I, \mathbf{y} \rightarrow \mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{Y}} (\text{cons}(\mathbf{y} \rightarrow \mathbf{x}) \odot \text{warp}(I(\mathbf{y}), \mathbf{y} \rightarrow \mathbf{x})). \quad (2)$$

The result is a weighted combination of deformed images. Warping (Sec. 4.2.3) models how an image changes when X-Field coordinates change by deforming it, and a per-pixel weight is given to this result to handle flow consistency (Sec. 4.2.5).

**4.2.3 Warping.** Warping deforms an observed into an unobserved image, conditioned on the observed and the unobserved X-Field coordinates:

$$\text{warp}(I, \mathbf{y} \rightarrow \mathbf{x}) \in \mathcal{I} \times \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{I}. \quad (3)$$

We use a spatial transformer (STN) [Jaderberg et al. 2015] with bilinear filtering, i.e., a component that computes all pixels in one image by reading them from another image according to a given flow map. STNs are differentiable, do not have any learnable parameters and are efficient to execute at test time. The key question is, (Fig. 5) from which position  $\mathbf{q}$  should a pixel at position  $\mathbf{p}$  read when the image at  $\mathbf{x}$  is reconstructed from the one at  $\mathbf{y}$ ?

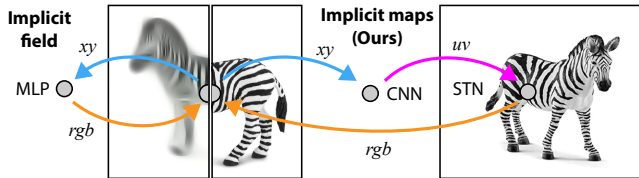


Fig. 5. Implicit maps: implicit fields (left) typically use an MLP to map 3D position to color, occupancy etc. We (right) add an indirection and map pixel position to texture coordinates to look up another image.

To answer this question, we look at the Jacobians of the mapping from X-Field coordinates to pixel positions. Here, Jacobians capture, for example, how a pixel moves in a certain view and light if time is changed, or its motion for one light, time and view coordinate if light is moved, and so forth. Formally, for a specific pixel  $\mathbf{p}$ , the Jacobian is:

$$\text{flow}_{\partial}(\mathbf{x})[\mathbf{p}] = \frac{\partial \mathbf{p}(\mathbf{x})}{\partial \mathbf{x}} \in \mathcal{X} \rightarrow \mathbb{R}^{2 \times n_d}, \quad (4)$$

where  $[\cdot]$  denotes indexing into a discrete pixel array. This is a Jacobian matrix with size  $2 \times n_d$ , which holds all partial derivatives of the two image pixel coordinate dimensions (horizontal and vertical) with respect to all  $n_d$ -dimensional X-Field coordinates. A Jacobian is only differential and does not yet define the finite position  $\mathbf{q}$  to read for at a pixel position  $\mathbf{p}$  as required by the STN.

To find  $\mathbf{q}$  we will now *project* the change in X-Field coordinate  $\mathbf{y} \rightarrow \mathbf{x}$  to 2D pixel motion using finite differences:

$$\text{flow}_{\Delta}(\mathbf{y} \rightarrow \mathbf{x})[\mathbf{p}] = \mathbf{p} + \Delta(\mathbf{y} \rightarrow \mathbf{x})\text{flow}_{\partial}(\mathbf{x})[\mathbf{p}] = \mathbf{q}. \quad (5)$$

Here, the finite delta in X-Field coordinates ( $\mathbf{y} \rightarrow \mathbf{x}$ ), an  $n_d$ -dimensional vector, is multiplied with an  $n_d \times 2$  matrix, and added to the start position  $\mathbf{p}$ , producing an absolute pixel position  $\mathbf{q}$  used by the STN to perform the warp. In other words, Eq. 4 specifies how pixels move for an infinitesimal change of X-Field coordinates, while Eq. 5 gives a finite pixel motion for a finite change of X-Field coordinates. We will now look into a learned representation of the Jacobian,  $\text{flow}_{\partial}$ , the core of our approach.

**4.2.4 Flow.** Input to the flow computation is only the X-Field coordinate  $\mathbf{x}$  and output is the Jacobian (Eq. 4). We implement this function using a CNN, in particular.

*Implementation.* Our implementation starts with a fully connected operation that transforms the coordinate  $\mathbf{x}$  into a  $2 \times 2$  image with 128 channels. The coord-conv [Liu et al. 2018] information (the complete  $\mathbf{x}$  at every pixel) is added at that stage. This is followed by as many steps as it takes to arrive at the output resolution, reducing the number of channels to produce at  $n_d$  output channels. For some input, it can be acceptable to produce a flow map at a resolution lower than the image resolution and warp high-resolution images using low-resolution flow, which preserves details in color, but not in motion.

*Compression.* Changes in some X-Field dimension can only change the pixel coordinates in a limited way. One example is view: all changes of pixel motion with respect to known camera motion can be explained by disparity [Forsyth and Ponce 2002]. So instead of modeling a full 2D motion to depend on all view parameters, we only generate per-pixel disparity and compute the flow Jacobian from disparity in closed form using reprojection. For our data, this is only applicable to depth, as no such constraints are in place for derivatives of time or light.

*Discussion.* It should also be noted that no pixel-basis RGB observation  $L_{in}(\mathbf{y})$  ever is input to  $\text{flow}_{\partial}$ , and hence, all geometric structure is encoded in the network. Recalling Sec. 3, we see this as both a burden, but also required to achieve the desired interpolation property: if the geometry NN can explain the observations at a few  $\mathbf{y}$ , it can explain their interpolation at all  $\mathbf{x}$ . This also justifies why  $\text{flow}_{\partial}$  is a NN and we do not directly learn a pixel-basis depth-motion map: it would not be interpolatable.

An apparent alternative would be to learn  $\text{flow}'_{\partial}(\mathbf{x}, \mathbf{y})$  to depend on both  $\mathbf{y}$  and  $\mathbf{x}$ , so as not to use a Jacobian, but allow any mapping. Regrettably, this does not result in interpolation. Consider a 1D view alone: Using  $\text{flow}_{\partial}(\mathbf{x})$  has to commit to one value that just minimizes image error after soft blending. If a hypothetical  $\text{flow}'_{\partial}(\mathbf{x}, \mathbf{y})$  can pick any different value for every pair  $\mathbf{x}$  and  $\mathbf{y}$ , it will do so without incentive for a solution that is valid in between them.

Finally, it should be noted, that using skip connections is not applicable to our setting, as the decoder input is a mere three numbers without any spatial meaning.

**4.2.5 Consistency.** To combine all observed images warped to the unobserved X-Field coordinate, we weight each image pixel by its *flow consistency*. For a pixel  $\mathbf{q}$  to contribute to the image at  $\mathbf{p}$ , the flow at  $\mathbf{q}$  has to map back to  $\mathbf{p}$ . If not, evidence for not being an occlusion is missing and the pixel needs to be weighted down.

Formally, consistency of one pixel  $\mathbf{p}$  when warped to coordinate  $\mathbf{x}$  from  $\mathbf{y}$  is the partition of unity of a weight function:

$$\text{cons}(\mathbf{y} \rightarrow \mathbf{x})[\mathbf{p}] = w(\mathbf{y} \rightarrow \mathbf{x})[\mathbf{p}] \left( \sum_{\mathbf{y}' \in \mathcal{Y}} w(\mathbf{y}' \rightarrow \mathbf{x})[\mathbf{p}] \right)^{-1}. \quad (6)$$

The weights  $w$  are smoothly decreasing functions of the 1-norm of the delta of the pixel position  $\mathbf{p}$  and the backward flow at the position  $\mathbf{q}$  where  $\mathbf{p}$  was warped to:

$$w(\mathbf{y} \rightarrow \mathbf{x})[\mathbf{p}] = \exp(-\sigma \|\mathbf{p} - \text{flow}_{\Delta}(\mathbf{x} \rightarrow \mathbf{y})[\mathbf{q}]\|_1). \quad (7)$$

Here  $\sigma = 10$  is a bandwidth parameter chosen manually. No benefit was observed when making  $\sigma$  a vector or learning it.

*Discussion.* In other work, consistency has been used in a loss, asking for consistent flow for unsupervised depth [Godard et al. 2017; Zhou et al. 2017] and motion [Zou et al. 2018] estimation. Our approach does not have consistency in the loss during training, but inserts it into the image compositing of the architecture, i.e., also to be applied at test time. In other approaches—that aim to produce depth, not images—consistency is not used at test time. Our flow can be, and for our problem has to be inconsistent: for very sparse images such as three views, many occlusions occur, leading to inconsistencies. Also flow due to, e.g., caustics or shadows probably has a fundamentally different structure compared to multi-view flow, that has been not explored in the literature we are aware of.

The graphics question answered here is, however, what to do with inconsistencies. To this end, instead of a consistency loss, we allow the architecture to apply multiple flows, such that the combined result is plausible when weighting down inconsistencies. In the worst case, no flow is consistent with any other and  $w$  has similar but small values for large cons which lead to equal weights after normalization, i.e., linear blending.

## 5 RESULTS

Here we will provide a comparison to other work (Sec. 5.1), evaluation of scalability (Sec. 5.2), and a discussion of applications (Sec. 5.3).

Please see the supplemental materials for an interactive WebGL demo to explore different X-Field data sets using our method, as well as a supplemental video to document temporal coherence.

### 5.1 Comparison

We compare our approach to other *methods*, following a specific *protocol* and by different *metrics* to be explained now:

*Methods.* We consider the following methods: OURS, BLENDING, WARPING, KALANTARIETAL, Local light-field fusion (LLFF), SUPER-SLOWMO, and three ablations of our approach: NOCORDCONV, as well as NOWARPING and NOCONSISTENCY.

Linear BLENDING is not a serious method, but documents the sparsity: plagued by ghosting for small baselines, we see our baseline/sparsity poses a difficult interpolation task, far from linear. It is applicable to all dimensions.

WARPING and SUPER-SLOWMO first estimate the correspondence in image pairs [Sun et al. 2018b] or light field data [Dabala et al. 2016] and later apply warping [Mark et al. 1997] with ULR-style weights [Buehler et al. 2001]. Note how ULR weighting accounts

for occlusion. Warping is applicable to time (Jiang et al. [2018]) and view interpolation (Dabala et al. [2016]).

KALANTARIETAL and LLFF are the publicly available implementations of Kalantari et al. [2016] and Mildenhall et al. [2019]. Both are applicable to and tested on lightfields, i.e., view interpolation, only.

To evaluate other work in higher dimensions, we further explore their hypothetical combinations, such as first using LLFF for view interpolation followed by SUPER-SLOWMO for time interpolation.

Finally, we compare three ablations of our method. The first, NOCORDCONV, regresses without coord-conv, i.e., will produce spatially invariant fields. The second, NOWARPING, uses direct regression of color values without warping. The third, NOCONSISTENCY, does not perform occlusion reasoning but averages directly. These are applicable to all dimensions.

As we did not have access to the reference implementation of Soft3D [Penner and Zhang 2017], we test on their data and encourage qualitative comparison by inspecting our results in Fig. 6 and their supplemental video.

*Protocol.* Success is quantified as the expected ability of a method to predict a set of held-out LF observed coordinates  $\mathcal{H}$  when trained on  $\mathcal{Y} - \mathcal{H}$ , i.e.,  $\mathbb{E}_{\mathbf{h} \sim \mathcal{H}} L_{\text{out}}(\mathbf{h}) \ominus_m L_{\text{in}}(\mathbf{h})$ , where  $\ominus_m$  is one of the metrics to be defined below.

For dense LF the held-out protocol follows Kalantari et al. [2016]: four corner views as an input. Sparse LF interpolation is on  $5 \times 5$ , holding out the center one. For time, interpolation triplets are used, i.e., we train on past and future frames, withholding the middle one.

*Metrics.* For comparing the predicted to the held-out view we use the  $L_2$ , SSIM and VGG [Zhang et al. 2018] metrics.

*Data.* We use the publicly available LF data from [Levoy and Hanrahan 1996], [Penner and Zhang 2017], [Dabala et al. 2016], and [Kalantari et al. 2016], LF video data from [Sabater et al. 2017], sequences from [Butler et al. 2012], relighting data from Xu et al. [2018] as well as custom captured reflectance field video. For aggregate statistics, we use 5 LFs, three videos and one view-time-light X-Field.

Our own data is captured using a minimalist setup: a pair of mobile phones. The first image takes the photo; the second one provides the light source. Both are moved with one, two or three degrees of freedom, depending on the scene. All animation is produced by stop motion. We have captured several X-Fields, but only one that has additional reference views to quantify quality.

*Results.* Tbl. 2 summarizes the outcome of the main comparison. We see that our method provides the best quality in all tasks according to all metrics on all domains.

For example images corresponding to the plots in Tbl. 2, please see Fig. 6 for interpolation in space, Fig. 7 and Fig. 10 for time, Fig. 11 for light, Fig. 8 for space-time and Fig. 9 for view-time-light results. In each figure we document the input view and multiple insets that show the results from all competing methods.

Figure 6 shows results for view interpolation. Here, WARPING produces crisp images, but pixel-level outliers that are distracting in motion, e.g., for the bench. KALANTARIETAL and LLFF do not capture the tip of the grass (top row). Instead, ghosted copies are observed. KALANTARIETAL is not supposed to work for larger baselines [Kalantari et al. 2016] and only shown for completeness on the



Table 2. Results of different methods (columns) for different dimensions (rows) according to different metrics. Below, the same data as diagrams. Colors encode methods. The best method according to one metric for one class of X-Field is denoted in bold font (for  $L_2$  and VGG less is better, for SSIM more is better). <sup>1</sup>For view-time interpolation, combined with LLFF.

View Time Light	LINEAR			WARPING			KALANTARI			LLFF			SuSLoMo <sup>1</sup>			NoWARP			NoCC			NoCONS			OURS					
	VGG	MSE	SSIM	VGG	MSE	SSIM	VGG	MSE	SSIM	VGG	MSE	SSIM	VGG	MSE	SSIM	VGG	MSE	SSIM	VGG	MSE	SSIM	VGG	MSE	SSIM	VGG	MSE	SSIM			
✓	421	221	.662	210	2.28	0.929	351	20.39	.769	223	2.78	.919	—	—	—	330	11.78	.768	421	6.45	.806	175	2.25	.941	<b>151</b>	<b>1.79</b>	<b>.951</b>			
✓	359	71	.723	—	—	—	—	—	—	—	—	—	224	3.90	.867	315	5.43	.778	497	7.63	.706	147	1.45	.935	<b>147</b>	<b>1.46</b>	<b>.935</b>			
✓	116	9	.940	—	—	—	—	—	—	—	—	—	120	.784	.947	119	0.95	.941	302	5.25	.848	111	0.68	.948	<b>111</b>	<b>0.66</b>	<b>.948</b>			
✓✓	620	176	.558	—	—	—	—	—	—	—	—	—	269	1.99	.892	388	7.67	.775	571	14.97	.632	273	2.61	.888	<b>252</b>	<b>2.00</b>	<b>.896</b>			
✓✓✓	522	209	.584	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	523	20.60	.595	493	10.10	.692	419	7.09	.719	<b>247</b>	<b>2.19</b>	<b>.827</b>

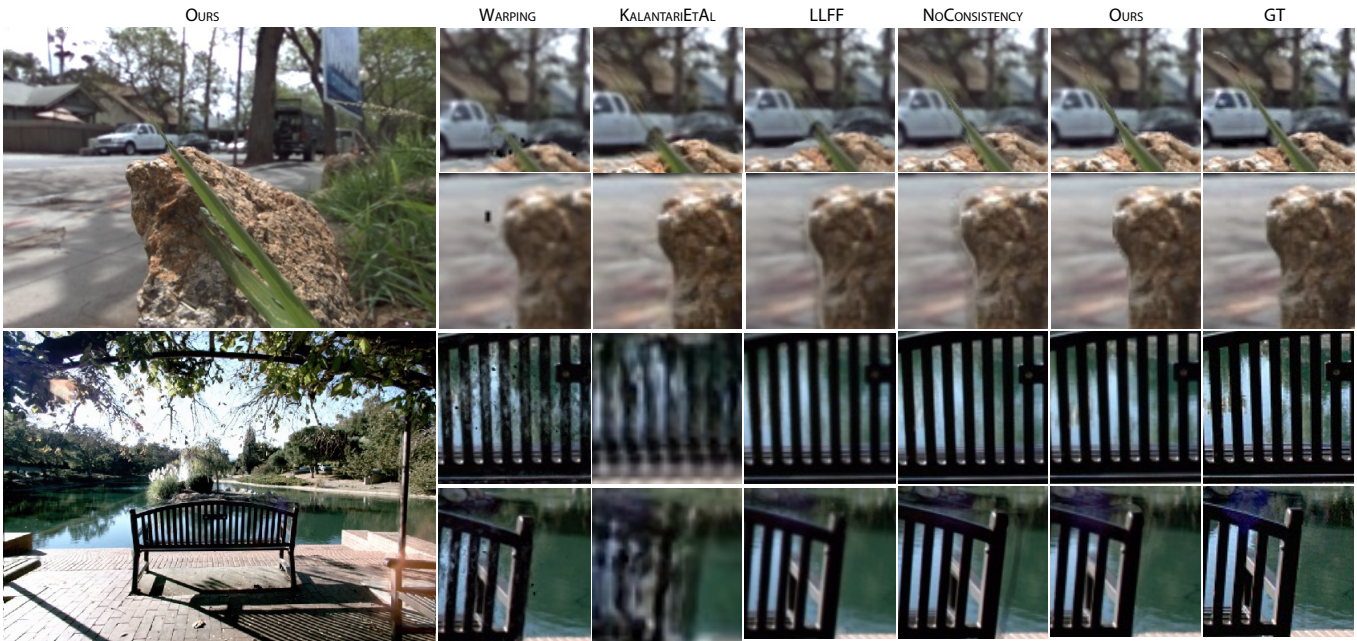
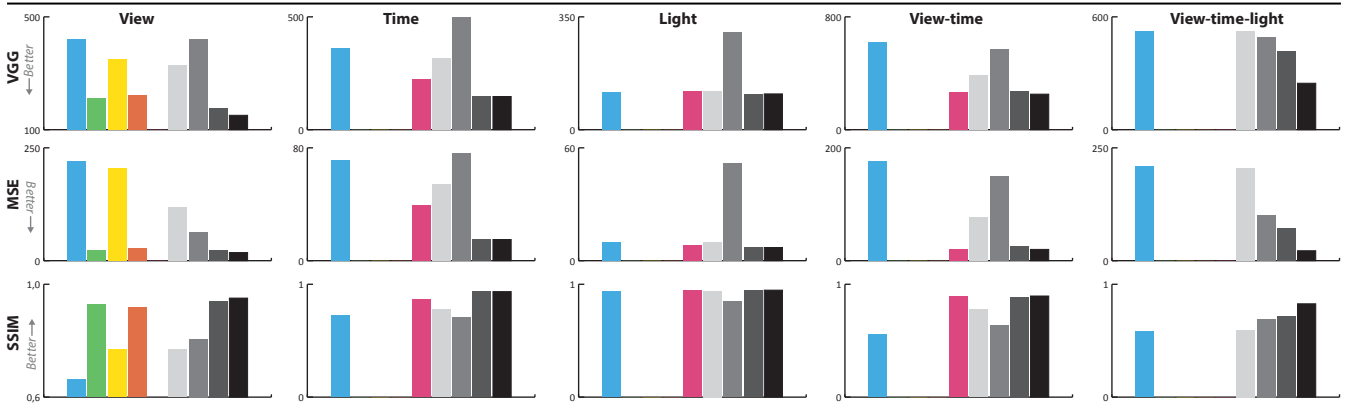


Fig. 6. Comparison of our approach for view interpolation to other methods for two scenes (rows). The top scene, from [Kalantari et al. 2016] is a dense LF; the one below, from [Penner and Zhang 2017], is sparse. Columns show, left to right, Ours at the position of the withheld reference, the results from (WARPING, KALANTARI EtAL, LLFF, and NoCONSISTENCY and Ours), as well as the ground truth as insets.

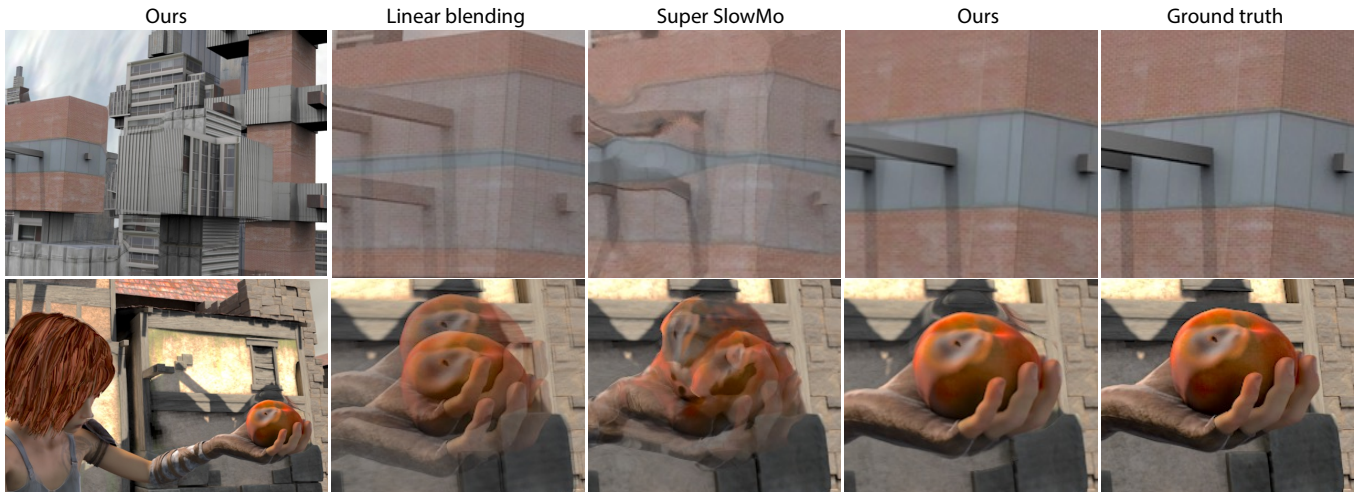


Fig. 7. Temporal interpolation for two scenes (rows) using different methods (columns). See Sec. 5.1 for a discussion.



Fig. 8. Results for view-time interpolation. The input was a  $2 \times 2 \times 2$  X-Field:  $2 \times 2$  sparse view observations with two frames.

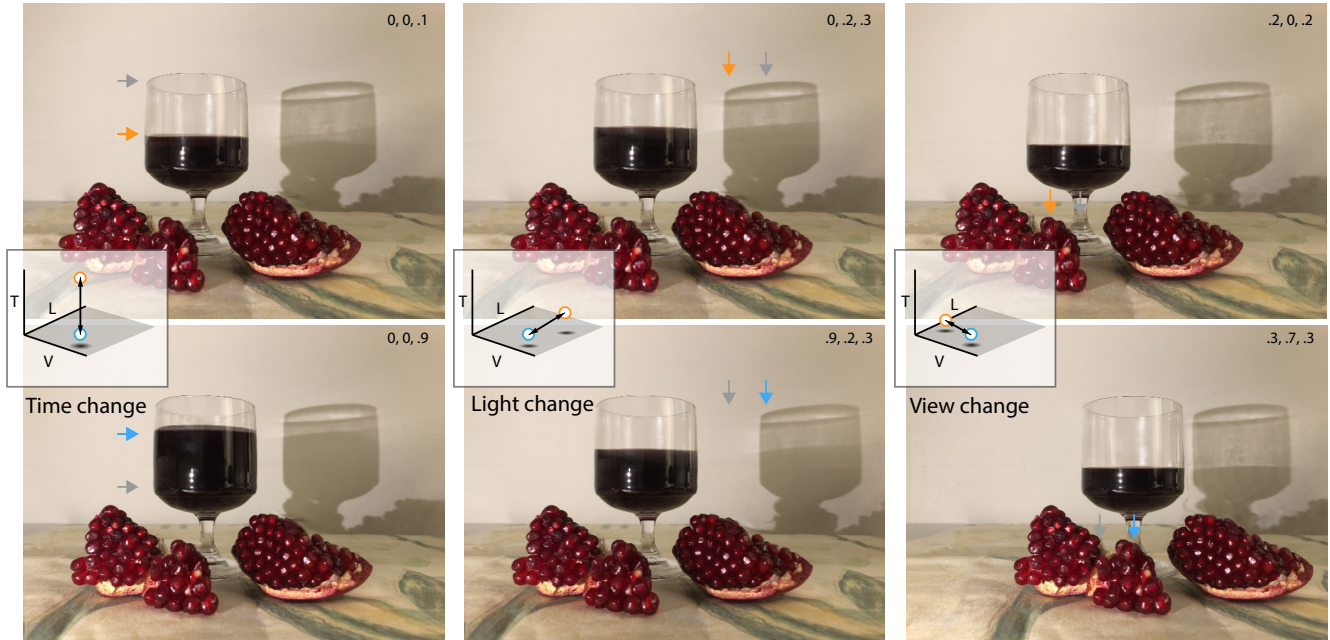


Fig. 9. Exploring a view-time-light X-Field. In each column, we show a change in dominant X-Field dimension. The input was a  $3 \times 3 \times 3$  X-Field. All images are at unobserved intermediate coordinates. Colored arrows indicate how image features have moved in response.

bench scene. LLFF produces slightly blurrier results for the sparse bench scene. Our NoCONSISTENCY shows the tip of the grass, but on top of ghosting. OURS has details, plausible motion and is generally most similar to the ground truth.

The temporal interpolation comparison in Fig. 7 indicates similar conclusions: BLENDING is not a usable option; not handling occlusion, also in time, creates ghosting due to overlap. SUPERSLowMo fails for both scenes as the motion is large. The motion size can be seen from the linear blending. Ultimately, OURS is similar to the ground truth. The motion smoothness is best seen in the slow-mo application of the supplemental video.

Interpolation between triplets of images can represent strong, non-rigid changes involving transparency, scattering, etc. (Fig. 10).

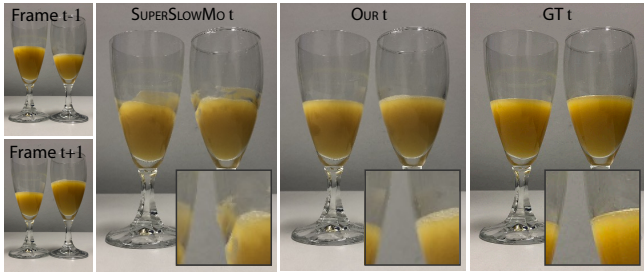


Fig. 10. Interpolation of two frames (shown left) compared to a reference using our approach and state-of-the-art SuperSlowMo [Jiang et al. 2018].

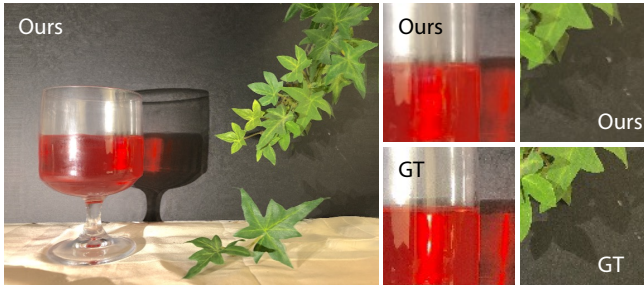


Fig. 11. Interpolation in the light dimension. We note that the image is plausible, even in the presence of cast shadows or caustics and transparency, maybe at the slight expense of blurring highlights and ghosting shadows.

Interpolation across light is seen in Fig. 11.

For light interpolation, Xu et al. [2018] is an extension of our ablation DIRECT, by an additional optimization over sample placement when assuming a capture dome. We will here compare to their implementation on their data. Please note that their method cannot be applied to our data as it requires a custom capture setup. Figure 12 shows a comparison from interpolating across a neighborhood of  $3 \times 3$  images out of the 541 dome images, covering a baseline of approximately 20 degrees. We see that direct regression blurs both the shadows and the highlights, while our method deforms the image, retaining sharpness. Tbl. 3 quantifies this result as the average across their test images “Dinosaur”, “Jewel” and “Angel”. Besides the 10-degree column corresponding to Fig. 12, we also include

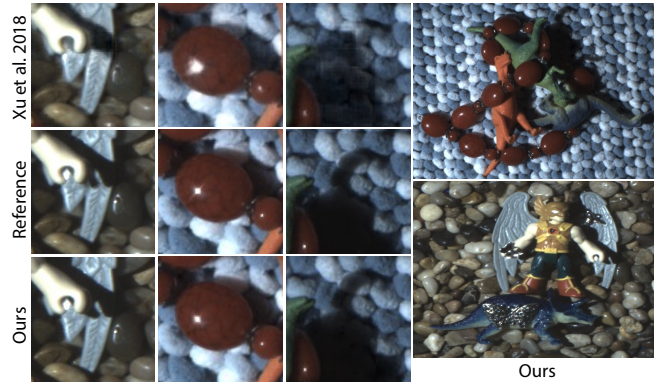


Fig. 12. Comparison between Xu et al. [2018] (top), the GT (middle) and our approach (bottom) for a 10 degree baseline.

Table 3. Relighting comparison to Xu et al. [2018] for different baselines.

Method	20° Baseline			30° Baseline			45° Baseline		
	VGG	MSE	SSIM	VGG	MSE	SSIM	VGG	MSE	SSIM
● XuETAL	192	.1424	.954	194	.1561	.950	196	.1580	.950
● OURS	93	.0335	.989	134	.0718	.970	169	.1220	.958

other baselines. We see that for wider baselines, Xu et al. [2018] both methods converge in quality.

When interpolating across view and time as in Fig. 8, ghosting effects get stronger for others, as images get increasingly different. OURS can have difficulties where deformations are not fully rigid, as seen for faces, but compensates for this to produce plausible images.

We conclude that both numerically and visually our approach can produce state-of-the-art interpolation in view and time in high spatial resolution and at high frame rates. Next, we look into evaluating the dependency of this success on different factors.

## 5.2 Evaluation

We evaluate our approach in terms of scalability with training effort and observation sparsity, speed and detail reproduction. These tests are performed on the view interpolation only.

*Analysis of albedo splitting.* Figure 13 shows an example of a scene that benefits from albedo splitting for a light interpolation. We find that splitting albedo and shading is critical for shadows cast on textured surfaces.

*Observation sparsity.* We interpolate from extremely sparse data. In Tbl. 4 we evaluate the quality of our interpolation depending on the number of training exemplars, also seen in Fig. 14.

*Speed.* At deployment, our method requires no more than taking a couple of numbers and passing them

Table 4. Error for the *Crystal Ball* scene with resolution  $512 \times 512$  using different metrics (columns) for different view counts (rows).

	LF	VGG19	L2	SSIM
3×3	140	.005	.90	
5×5	119	.003	.93	
9×9	102	.002	.95	

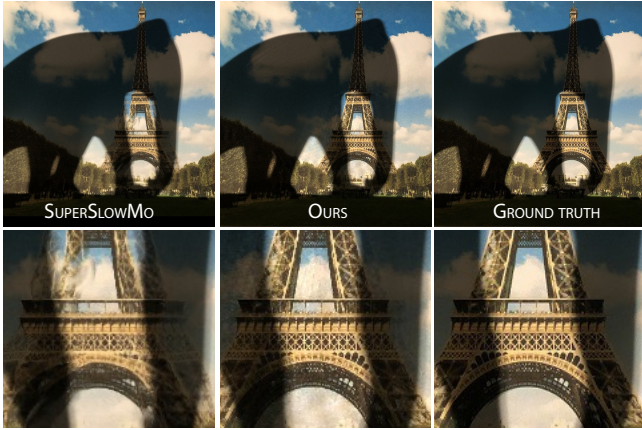


Fig. 13. Splitting albedo and shading: When the elephant’s shadow meets a texture of the Eiffel Tower unprepared, a single-layer method such as SUPER-SLOWMo cannot find a unique flow and produces artifacts. Our approach leaves both shadow and texture structures mostly intact.

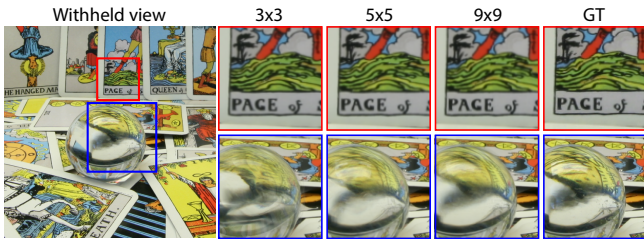


Fig. 14. Visual quality of our approach as a function of increasing (left to right) training set size for view interpolation.

through a decoder for each observation, followed by warping and a weighting. The end-speed for view navigation is around 20 Hz (on average 46 ms per frame) at  $1024 \times 1024$  for a  $5 \times 5$  LF on an Nvidia 1080Ti with 12 GB RAM.

*Training effort.* Our approach needs to be trained again for every LF. Typical training time is listed in Tbl. 5. Figure 15 shows progression of interpolation quality over learning time. We see that even after little training, results can be acceptable. Overall, we see that training the NN requires a workable amount of time, compared approaches trained in the order of many hours or days.

*Smoothness.* The depth and flow map we produce are smooth in view and time and may lack detail. It would be easy to add skip connections to get the details from the appearance. Regrettably, this would only work on the input image, and that needs to be withheld at training, and is unknown at test time. An example of this is seen in Fig. 16. This smoothness is one source of artifacts.

Table 5. Training time (minutes) and network parameters for different resolutions for a  $5 \times 5$  LF array and spatial interpolation.

	$512^2$	$1024^2$	$1764^2$
Time	28	60	172
Params	482 k	492 k	492 k

Overcoming this, e.g., using an adversarial design, is left to future work.

*Coherence.* Visual coherence across dimensions, when traversing the X-space smoothly, is best assessed from the supplemental video. Our method might miss details or over-smooth, but is coherent, as first, we never regress colors that flicker, only texture coordinates; second, Jacobians are multiplied with view differentials in a linear operation, and hence smooth; third as the NNs to produce Jacobians are smooth functions and, finally, soft occlusion is smooth.

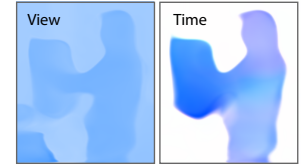


Fig. 16. Correspondence for Fig. 8.

### 5.3 Applications

Figure 17 demonstrates motion blur (time interpolation), depth-of-field (view interpolation), and both (interpolating both).

## 6 DISCUSSION / LIMITATIONS

We find the success of our method to largely depend on three factors, Data, Model and Capacity, which we will discuss next. Please also see the supplemental video and Fig. 19 for examples of such limitations.

*Data.* We train from very sparse observations, often only a dozen images. It is clear that information not present in any image, will not be reconstructed. Even parts observed in only one image can be problematic (Fig. 18, left). A classic example is occlusion: if only three different views are available and two occlude an area that is not occluded in one view, this area will be filled in. However, this fill-in will occur in the domain we learn, the X-Field Jacobian. Hence, disoccluded pixels will change their position similarly to their spatial neighbours. Artifacts manifest as rubber-like stretches between the disoccluding and the occluding object. The chair example from the supplemental video and Fig. 19 shows artifacts resulting from lack of data. Similarly, the foam in the supplemental video and Fig. 19 is stochastic and different in every image, and hence unable to form fine-scale correspondence. The consistency weighting typically removes them. Future work might overcome this limitation by training on more than one scene.

*Model.* We combine a primitive, hard-wired image formation model with a learned scene representation. As long as the data roughly follows this model, this is a winning combination. Scenes that are entirely beyond the model’s scope might fail and will do so independently of the amount of data or the representation capacity.

Our key assumption is that changes are explained by flow. This is not a reasonable assumption with dominant transparency [Kopf et al. 2013]. Changes in brightness due to casual capture with auto-exposure can cause variation that our deformation model fails to explain. In an X-Field non-unique flow is common: after one bounce, multiple indirect shadows might overlap and move differently.

We address this by processing the signal, so a unique flow becomes more applicable: by splitting shading and albedo, by representing the full X-Field Jacobian, by learning a non-linear inverse flow instead of linearly interpolating a forward flow, etc. Finally, if

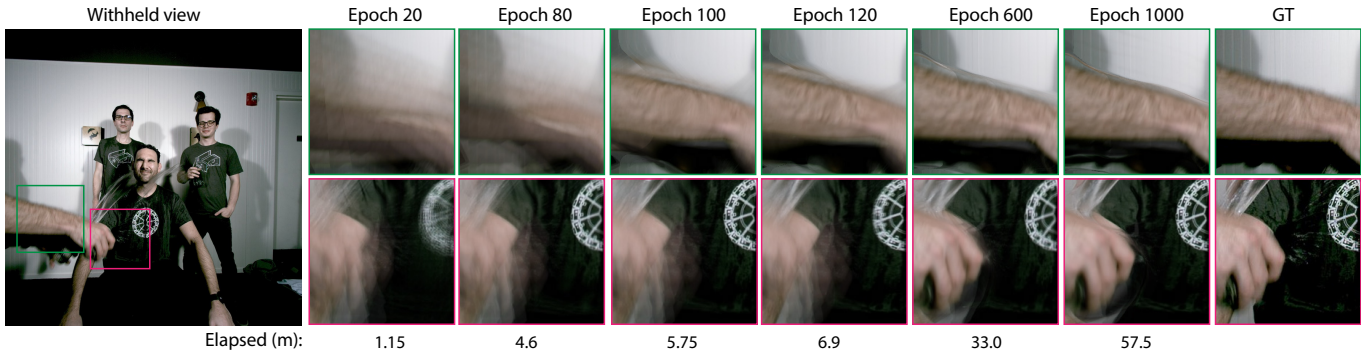


Fig. 15. Progression of visual fidelity for different training efforts (horizontal axis) for two insets (vertical axis) in one scene. After 500 epochs (ca. 30 minutes) the result is usable, and it converges after 1000 epochs (ca. 1h). Note that epochs are short as we only have 5×5 training examples.

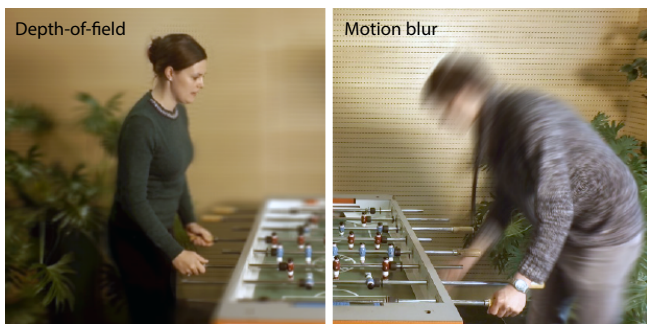


Fig. 17. Two LF video-enabled effects, computed using view interpolation: Depth-of-field (**left**) and motion blur (**right**). For both, we generate and average many images at X-Field coordinates covering a lens resp. shutter.

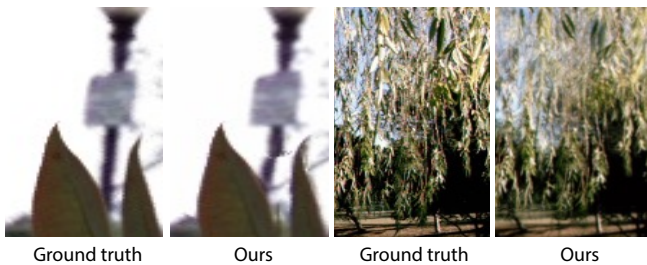


Fig. 18. Two failure cases of our method, documenting, left, insufficient data (the lamp post is only visible in one view and happens to become attached to the foreground leaf) and right, insufficiency of the capacity (the depth structure of the twigs is too complex to be represented by an architecture we can train at this points). Please see the supplemental video for changes of view point to best appreciate the effect.

all flows were wrong, consistency weighting degenerates to linear blending. Future work could learn layered flow [Sun et al. 2012].

*Capacity.* Finally, even if all data is available, the model is perfect and the model assumptions are fulfilled, the NN needs to have the capacity to represent the input to the model. Naturally, any finite model can only be an approximation, and hence, the flow, and consequently shape, illumination and motion is smooth. The NN allows for some level of sharpness via non-linearities as in other

implicit representations [Chen and Zhang 2019; Niemeyer et al. 2019; Oechsle et al. 2019] but the amount of information is finite (Fig. 18, right). Capturing sharp silhouettes is clearly possible, but to represent a scene with stochastic variation, stochasticity should be inserted [Karras et al. 2019] in combination with a style loss.

## 7 CONCLUSION

We have demonstrated representing an X-Field as a NN that produces images, conditioned on view, time and light coordinates. The interpolation is high-quality and high-performance, outperforming several competitors for dynamic changes of advanced light transport (all BRDFs, (soft) shadows, GI, caustics, reflections, transparency), as well as fine spatial details (plant structures), both for single objects (still-life scenes) and entire scenes (tabletop soccer, parks).

The particular structure of a network that combines a learnable view-time geometry model, combined with warping and reasoning on consistency, has shown to perform better than direct regression of color or warping without handling occlusion and state-of-the-art domain-adapted solutions.

We want to reiterate that, partly, this success is possible because we changed a general task to a much simpler one: instead of interpolating all possible combinations of images, we only interpolate a fixed set. Strong generalization is a worthwhile and exciting scientific goal, in particular from an AI perspective. But, depending on the use case, it might not be required in applied graphics: With our approach, after 20 minutes of pre-calculation, we can deploy an X-Field in a VR application to play back at interactive rates. A user enjoying this high-quality visual experience might not ask if the same network could generalize to a different scene or not.

In future work, other data such as data from Lightstages or sparse and unstructured capture, as well as extrapolation, should be explored. We aim to further reduce training time (eventually using learned gradient descent [Flynn et al. 2019]), and explore interpolation along other domains such as wavelength or spatial audio [Engel et al. 2017], as well as reconstruction from even sparser observations.

## REFERENCES

- Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. 2019a. Depth-aware Video Frame Interpolation. In *Proc. CVPR*.  
 Wenbo Bao, Wei-Sheng Lai, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. 2019b. MEMC-Net: Motion Estimation and Motion Compensation Driven Neural

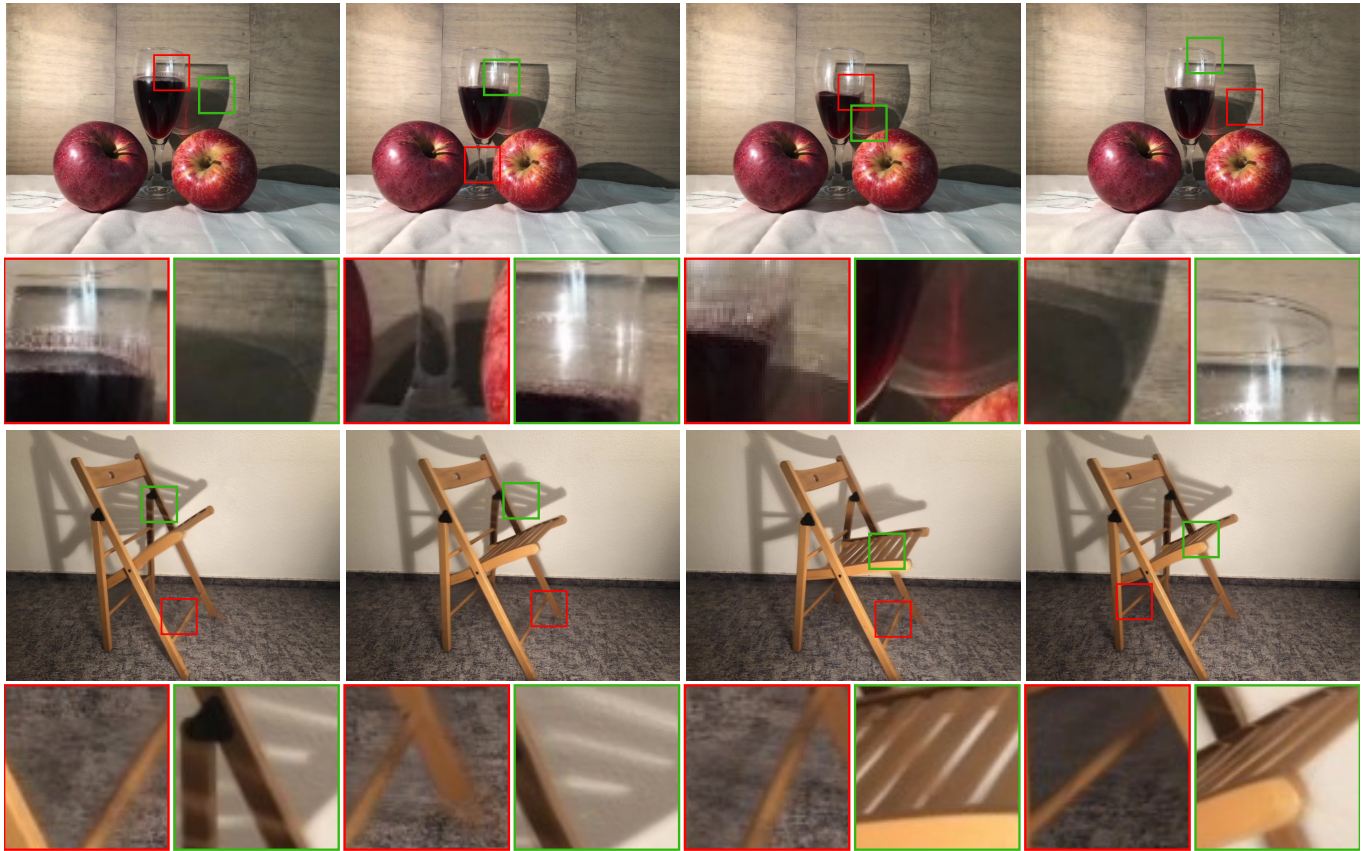


Fig. 19. Our interpolation results for two scenes (“Apple” and “Chair”) from the supplemental video: Insets in red identify regions where artifacts appeared, and insets in green indicate challenging examples which our method interpolated successfully. Artifacts mainly happened due to lack of training data; in the apple scene (top), a  $3 \times 3 \times 3$  X-Field capture, the caustics in the shadow shows in only one view, and the foam is stochastic and different at each level of the liquid. In these regions, appearance does not properly interpolate but fades in and out, leading to ghosting or blurring, best seen in the supplemental video. In the chair scene (bottom), which is a  $5 \times 5 \times 5$  X-Field, the texture on the carpet beneath the chair gets blurry as this part of the carpet becomes visible only in one view due to occlusion caused by the chair and its shadow. However, our method could handle soft shadow casting on a textured background or when there is a moving shadow of complex object occluded with the object itself.

- Network for Video Interpolation and Enhancement. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019).
- Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. 2001. Unstructured Lumigraph Rendering. In *Proc. SIGGRAPH*.
- D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. 2012. A naturalistic open source movie for optical flow evaluation. In *European Conf. on Computer Vision (ECCV) (Part IV, LNCS 7577)*, A. Fitzgibbon et al. (Eds.) (Ed.). Springer-Verlag, 611–625.
- Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. 2013. Depth Synthesis and Local Warps for Plausible Image-based Navigation. *ACM Trans. Graph.* 32, 3 (2013).
- Anpei Chen, Minye Wu, Yingliang Zhang, Nianyi Li, Jie Lu, Shenghua Gao, and Jingyi Yu. 2018. Deep Surface Light Fields. *Proc. i3D* 1, 1 (2018), 14.
- Billy Chen and Hendrik PA Lensch. 2005. Light Source Interpolation for Sparsely Sampled Reflectance Fields. In *Proc. Vision, Modeling and Visualization*. 461–469.
- Xu Chen, Jie Song, and Otmar Hilliges. 2019. Monocular Neural Image Based Rendering with Continuous View Control. In *ICCV*.
- Zhiqin Chen and Hao Zhang. 2019. Learning Implicit Fields for Generative Shape Modeling. *CVPR* (2019).
- Lukasz Dabala, Matthias Ziegler, Piotr Didyk, Frederik Zilly, Joachim Keinert, Karol Myszkowski, Hans-Peter Seidel, Przemyslaw Rokita, and Tobias Ritschel. 2016. Efficient Multi-image Correspondences for On-line Light Field Video Processing. *Comp. Graph. Forum (Proc. Pacific Graphics)* (2016).
- Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. 2000. Acquiring the Reflectance Field of a Human Face. In *Proc. SIGGRAPH*. 145–156.
- Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. 2015. Learning to Generate Chairs with Convolutional Neural Networks. In *CVPR*.
- Song-Pei Du, Piotr Didyk, Frédo Durand, Shi-Min Hu, and Wojciech Matusik. 2014. Improving Visual Quality of View Transitions in Automultiscopic Displays. *ACM Trans. Graph. (Proc. SIGGRAPH)* 33, 6 (2014).
- Jesse Engel, Cijnon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan. 2017. Neural Audio Synthesis of Musical Notes with Wavenet Autoencoders. In *JMLR*.
- John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. 2019. DeepView: View Synthesis With Learned Gradient Descent. In *CVPR*.
- John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. 2016. DeepStereo: Learning to Predict New Views From the World’s Imagery. In *CVPR*.
- David A Forsyth and Jean Ponce. 2002. *Computer Vision: a Modern Approach*. Prentice-Hall Professional Technical Reference.
- Ohad Fried and Maneesh Agrawala. 2019. Puppet Dubbing. In *Proc. EGSR*.
- Martin Fuchs, Volker Blanz, Hendrik P.A. Lensch, and Hans-Peter Seidel. 2007. Adaptive Sampling of Reflectance Fields. *ACM Trans. Graph.* 26, 2, Article 10 (2007).
- Clément Godard, Oisín Mac Aodha, and Gabriel J. Brostow. 2017. Unsupervised Monocular Depth Estimation with Left-Right Consistency. In *CVPR*.
- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Proc. NIPS*.
- Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. 1996. The Lumigraph. In *SIGGRAPH*.

- Kaiwen Guo et al. 2019. The Relightables: Volumetric Performance Capture of Humans with Realistic Relighting. *ACM Trans. Graph. (Proc SIGGRAPH Asia)* 38, 5 (2019).
- Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel J. Brostow. 2018. Deep Blending for Free-Viewpoint Image-Based Rendering. *ACM Trans. Graph. (Proc. SIGGRAPH)* 37, 6 (2018).
- Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. 2016. Scalable Inside-Out Image-Based Rendering. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 35, 6 (2016).
- Philipp Henzler, Niloy J Mitra, , and Tobias Ritschel. 2019. Escaping Plato's Cave: 3D Shape From Adversarial Rendering. (2019).
- Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the Dimensionality of Data with Neural Networks. *Science* 313, 5786 (2006).
- Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. 2015. Spatial Transformer Networks. In *Proc. NIPS*.
- Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. 2018. Super SloMo: High Quality Estimation of Multiple Intermediate Frames for Video Interpolation. In *CVPR*.
- Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. 2016. Learning-based View Synthesis for Light Field Cameras. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 35, 6 (2016).
- Tero Karras, Samuli Laine, and Timo Aila. 2019. A Style-based Generator Architecture for Generative Adversarial Networks. In *CVPR*. 4401–4410.
- Petr Kellnhofer, Piotr Didyk, Szu-Po Wang, Pitchaya Sitthi-Amorn, William Freeman, Fredo Durand, and Wojciech Matusik. 2017. 3DTV at Home: Eulerian-Lagrangian Stereo-to-Multiview Conversion. *ACM Trans. Graph. (Proc. SIGGRAPH)* 36, 4 (2017).
- Diederik P Kingma and Max Welling. 2013. Auto-encoding Variational Bayes. In *Proc. ICLR*.
- Johannes Kopf, Fabian Langguth, Daniel Scharstein, Richard Szeliski, and Michael Goesele. 2013. Image-Based Rendering in the Gradient Domain. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 32, 6 (2013).
- Marc Levoy and Pat Hanrahan. 1996. Light field rendering. In *SIGGRAPH*.
- Christian Lipski, Christian Linz, Kai Berger, Anita Sellent, and Marcus Magnor. 2010. Virtual Video Camera: Image-Based Viewpoint Navigation Through Space and Time. *Computer Graphics Forum* 29, 8 (Dec 2010).
- Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. 2018. An Intriguing Failing of Convolutional Neural Networks and the CoordConv Solution. In *Proc. NIPS*.
- Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. 2019. Neural Volumes: Learning Dynamic Renderable Volumes from Images. *ACM Trans. Graph. (Proc. SIGGRAPH)* 38, 4 (2019).
- Dhruv Mahajan, Fu-Chung Huang, Wojciech Matusik, Ravi Ramamoorthi, and Peter Belhumeur. 2009. Moving Gradients: a Path-based Method for Plausible Image Interpolation. In *ACM Trans. Graph.*, Vol. 28. 42.
- Tom Malzbender, Dan Gelb, and Hans Wolters. 2001. Polynomial Texture Maps. In *Proc. SIGGRAPH*.
- Russell A. Manning and Charles R. Dyer. 1999. Interpolating View and Scene Motion by Dynamic View Morphing. In *CVPR*, Vol. 1. 388–394.
- William R Mark, Leonard McMillan, and Gary Bishop. 1997. Post-rendering 3D Warping. In *Proc. 3D*.
- Maxim Maximov, Laura Leal-Taixé, Mario Fritz, and Tobias Ritschel. 2019. Deep Appearance Maps. In *Proc. ICCV*.
- Abhimitra Meka et al. 2019. Deep Reflectance Fields - High-Quality Facial Reflectance Field Inference From Color Gradient Illumination. *ACM Trans. Graph. (Proc SIGGRAPH)* 38, 4 (2019).
- Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. 2019. Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines. *ACM Trans. Graph. (Proc. SIGGRAPH)* 38, 4, Article 29 (2019).
- Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *ArXiv 2003.08934* (2020).
- Thu Nguyen Phuoc, Chuan Li, Stephen Balaban, and Yongliang Yang. 2018. RenderNet: A deep Convolutional Network for Differentiable Rendering from 3D Shapes. (2018).
- Thu Nguyen Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yongliang Yang. 2019. HoloGAN: Unsupervised Learning of 3D Representations From Natural Images. In *ICCV*.
- Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. 2019. Occupancy Flow: 4D Reconstruction by Learning Particle Dynamics. In *Proc. ICCV*.
- Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. 2019. Texture Fields: Learning Texture Representations in Function Space. *ICCV* (2019).
- Pieter Peers, Dhruv K. Mahajan, Bruce Lamond, Abhijeet Ghosh, Wojciech Matusik, Ravi Ramamoorthi, and Paul Debevec. 2009. Compressive Light Transport Sensing. *ACM Trans. Graph.* 28, 1, Article 3 (2009).
- Eric Penner and Li Zhang. 2017. Soft 3D Reconstruction for View Synthesis. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 36, 6 (2017).
- Julien Philip, Michaël Gharbi, Tinghui Zhou, Alexei A. Efros, and George Drettakis. 2019. Multi-View Relighting Using a Geometry-Aware Network. *ACM Trans. Graph.* 38, 4, Article Article 78 (2019).
- Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *Arxiv 1511.06434* (2015).
- Dikpal Reddy, Ravi Ramamoorthi, and Brian Curless. 2012. Frequency-Space Decomposition and Acquisition of Light Transport under Spatially Varying Illumination. In *Proc. ECCV*. 596–610.
- Scott E Reed, Yi Zhang, Yuting Zhang, and Honglak Lee. 2015. Deep Visual Analogymaking. In *NIPS*.
- Peiran Ren, Yue Dong, Stephen Lin, Xin Tong, and Baining Guo. 2015. Image Based Relighting Using Neural Networks. *ACM Trans. Graph. (Proc. SIGGRAPH)* 34, 4 (2015).
- Neus Sabater, Guillaume Boisson, Benoit Vandame, Paul Kerbirou, Frederic Babon, Matthieu Hog, Tristan Langlois, Remy Gendrot, Olivier Bureller, Arno Schubert, and Valerie Allie. 2017. Dataset and Pipeline for Multi-View Light-Field Video. In *CVPR Workshops*.
- Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. 2019. PIFu: Pixel-Aligned Implicit Function for High-Resolution Clothed Human Digitization. *Proc. ICCV* (2019), 2304–2314.
- Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. 2019a. DeepVoxels: Learning Persistent 3D Feature Embeddings. In *CVPR*.
- Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. 2019b. Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations. In *NeurIPS*.
- D. Sun, E. B. Sudderth, and M. J. Black. 2012. Layered Segmentation and Optical Flow Estimation Over Time. In *CVPR*. 1768–1775.
- Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. 2018b. PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. In *CVPR*.
- Shao-Hua Sun, Minyoung Huh, Yuan-Hong Liao, Ning Zhang, and Joseph J Lim. 2018a. Multi-view to Novel View: Synthesizing Novel Views with Self-learned Confidence. In *Proc. ECCV*.
- A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, R. Pandey, S. Fanello, G. Wetzstein, J.-Y. Zhu, C. Theobalt, M. Agrawala, E. Shechtman, D. B Goldman, and M. Zollhöfer. 2020. State of the Art on Neural Rendering. *Comp. Graph. Forum (EG STAR 2020)* (2020).
- Justus Thies, Michael Zollhöfer, and Matthias Nießner. 2019. Deferred Neural Rendering: Image Synthesis using Neural Textures. *ACM Trans. Graph. (Proc. SIGGRAPH)* (2019).
- Huamin Wang and Ruigang Yang. 2005. Towards Space: Time Light Field Rendering. In *Proc. 3D*.
- Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018. Video-to-Video Synthesis. In *NeurIPS*.
- Ting-Chun Wang, Jun-Yan Zhu, Nima Khademi Kalantari, Alexei A. Efros, and Ravi Ramamoorthi. 2017. Light Field Video Capture Using a Learning-Based Hybrid Imaging System. *ACM Trans. Graph. (Proc. SIGGRAPH)* 36, 4 (2017).
- Tom White. 2016. Sampling Generative Networks. *Arxiv 1609.04468* (2016).
- Bennett Wilburn, Neel Joshi, Vaibhav Vaish, Eino-Ville Talvala, Emilio Antunez, Adam Barth, Andrew Adams, Mark Horowitz, and Marc Levoy. 2005. High performance imaging using large camera arrays. In *ACM SIGGRAPH*. 765–76.
- Zexiang Xu, Sai Bi, Kalyan Sunkavalli, Sunil Hadap, Hao Su, and Ravi Ramamoorthi. 2019. Deep View Synthesis from Sparse Photometric Images. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 76.
- Zexiang Xu, Kalyan Sunkavalli, Sunil Hadap, and Ravi Ramamoorthi. 2018. Deep Image-based Relighting from Optimal Sparse Samples. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 126.
- Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*. 586–95.
- Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. 2017. Unsupervised Learning of Depth and Ego-Motion from Video. In *CVPR*.
- Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. 2018. Stereo Magnification: Learning View Synthesis Using Multipane Images. *ACM Trans. Graph. (Proc. SIGGRAPH)* 37, 4 (2018).
- Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. 2016. View Synthesis by Appearance Flow. In *ECCV*.
- C Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. 2004. High-quality Video View Interpolation Using a Layered Representation. In *ACM Trans. Graph.*, Vol. 23.
- Yuliang Zou, Zelun Luo, and Jia-Bin Huang. 2018. DF-net: Unsupervised Joint Learning of Depth and Flow Using Cross-task Consistency. In *ECCV*. 36–53.