Exploration of Capabilities and Limitations in View Change of the X-Fields Model

Jack Wang

January 15, 2022

Abstract

Generating images of the same scenes from different perspectives — whether that is from different points, from different angles, under varying illumination, or with other parameters — has a myriad of use cases, stretching from creating debug models to producing smooth videos. In the X-Fields model, hard-coded graphics tricks like lighting, 3D projection, and albedo are used to supplement neural networks in creating a differentiable map for the image parameters and the actual pixels using sample images and their corresponding coordinate values. Although X-Fields performs well on datasets of images concentrated on a 2D (x, y) plane relative to alternative interpolation methods, the original model cannot support broader, practical use cases like the interpolation of images in different 3D (x, y, z) positions. In this paper, we use 3D images and coordinates generated by the 3DB framework in our dimensionally expanded X-Fields model. We find that the new model can generate promising interpolation results with relatively sparse datasets and with large view angle changes; parameters such as learning rate, the bandwidth parameter in soft blending, and others have impact over the interpolation quality and construct trade-offs between training cost and interpolation quality; and that adding certain backgrounds (like the ocean) reference images can pose challenges for interpolation.

Introduction

Motivation

Generating images of the same scenes from different perspectives—whether that is from different points (video), from different angles (light fields), under varying illumination (reflectance fields), or with other parameters—has a myriad of use cases, stretching from creating debug models to creating smooth videos.

[Bemana 20] proposes an approach to interpolation across view, light, and time for any set of images, a process built on parameters defined as part of an "X-Field" (where X may be any combination of view, light, time, or other dimensions, such as the color spectrum). Their research shows how the right neural network (NN) can be used to create a universal, compact, and interpolatable X-Fields representation.

Although X-Fields performs well on the given example datasets relative to alternative interpolation methods, the current X-Fields model struggles in adapting to broader use cases. Presently, compatible X-Fields training datasets contain very specific training examples that are not easy to capture in real-world situations and involve an unnecessarily complicated file-naming process. I seek to make X-Fields compatible with more data and simplify the user experience. Also, while X-Fields are shown to be effective for interpolating viewpoints on an XY plane, I add another dimension (Z) to allow interpolation in 3D space.

Background

In this section, I introduce the X-Fields model proposed by [Bemana 20] and the debugging framework for computer vision models (3DB) [Leclerc 21]. I extend the X-Field model by adding another coordinate feature and test the performance by samples generated from the 3DB framework.

In the X-Fields model, [Bemana 20] proposes an approach to represent an X-Field of the same scene from different views (video), from various angles (light fields), under varied illumination (reflectance fields), and under a variety of other conditions allowing users to explore view, light, and time changes by learning a neural network (NN) to map coordinates to 2D images. By knowing the "basic tricks" of graphics (e.g., lighting, 3D projection, and occlusion) in a hard-coded and differentiable form, the X-Fields NN encodes the input as an implicit map such that for any view, light, or coordinate, it can quantify how it will move if view, time or light coordinates change for any pixel.

The 3DB model is aimed to identify and evaluate the failure modes in computer vision models. It leverages a 3D simulator to render realistic scenes that can be fed into any computer vision system. Users can customize a set of transformations to apply to the scene, such as view points, background changes, etc. The 3DB framework is used to generate training and testing datasets by specifying a set of pose changes on a predefined object.

Related work

This section summaries the previous techniques used to interpolate discrete images in terms of view. The view interpolation also refers to the concept of light fields (LFs), which is the set of all images of a scene for all views. [Levoy 96] and [Gortler 96] first formalize the concept of light fields and develop the hardware to capture the views by generalizing from observers' positions and orientations. Results show that simple linear blending for view interpolations may lead to ghosting effects.

The distance of a capture can also influence interpolation results. Sparse captures mean less number of images with greater distinction among neighboring images. Examples of sparse capture include 34 views on a sphere [Lombardi 19] and 40 ones on a hemisphere [Malzbender 01]. On the other hand, dense captures refer to a larger number of similar images. [Kalantari 16] applies dense images (very close positions) for a Lytro camera.

In the early stage, Unstructured Lumigraph Rendering (ULR) is used to create proxy geometry to warp multiple images into a target view and blend them with corresponding weights [Buehler 02]. Recent work includes per-view geometry [Hedman 16] and learned ULR blending weights [Hedman 18] to allow for sparse input and shade effects.

Later on, researchers focused on learning synthesized novel views for LF data. [Kalantari 16] learned depth maps in an unsupervised way and interpolated views via a Lytro camera. [Flynn 16] proposed to decompose LFs into multiple depth planes of output views and build a plan sweep volume (PSV) mechanism with dependent views. [Zou 18] constructed multi-plane image (MPI) representations via learning how neighboring views can impact the output one.

Another attractive idea has been to use a volumetric occupancy representation. [Penner 17] inferred a good volumetric / MPI representation that can be facilitated with learned gradient descent, where the gradient components directly encode visibility and effectively inform the NN of the occlusion relations in the scene. The advantage of MPI techniques is in avoiding explicit depth reconstruction, which allows for softer and better results. The X-Fields model involves a learning route as well, but explains the entire X-Field and uses 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.

[Zou 18]'s work on Appearance Flow combines the idea of warping pixels with learning how to warp. They typically take a single input view in account and [Sun 18] utilizes multiple views to improve warped view quality. Both methods use an implicit representation of the warp field

(i.e. a NN that for every pixel in one view predicts from where to replicate the value of each pixel in one image in the new view). While those approaches worked best for training with fixed camera positions, [Chen 19] introduced an implicit NN of per-pixel depth that allows for variable view interpolations. All these methods require extensive training for certain types of scenarios such as cars, chairs, or urban city views. The X-Fields model expands on this line of work further by creating an implicit NN representation that generalizes entire geometry, motion, and illumination changes. Its task on one hand is simpler since they do not generalize across different scenes, yet it is at the same time more difficult as I generalize over more dimensions and strive for state-of-the-art visual quality.

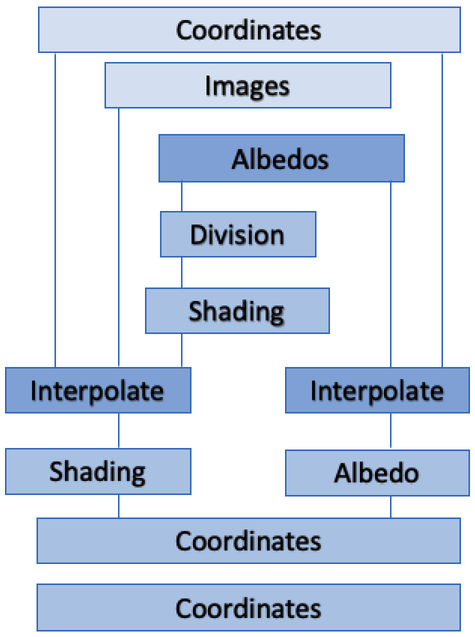


Figure 1: Data flow for an example with three dimensions.

Proposed Method

Objective

I adopt the original X-Field [Bemana 20], as a nonlinear function,

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

where trainable parameters θ map from the n_d- dimensional X-Field coordinate $x \in \mathcal{X}$ to 2D RGB images with n_p pixels. The capture modality determines the X-Field dimension. For example, the dimension of X-Fields for our 3D implementation would be three spatial coordinates.
As illustrated in X-Fields models, The subset of observed X-Fields coordinates is denoted as $\mathcal{Y} \subseteq \mathcal{X}$. The input image $L_{out}(y)$ is captured by the known coordinate $y \in \mathcal{Y}$. To minimize the following objective function, we can find parameters θ by

$$\theta \;=\; \arg\min_{\theta'} \; E_{y\sim\mathcal{Y}} || \, L_{out}^{(\theta)}(y) \;-\; L_{in}(y) \; ||_1$$

where $E_{y\sim\mathcal{Y}}$ is the expectation among coordinates $\mathcal{Y}$. The architecture L_{out} is trained to match images L_{in} using known coordinates y in order to capture unobserved coordinates. The proposed method is based on interpolation and it requires $\mathcal{X}$ is convex combination of $\mathcal{Y}$.

Architecture

The X-Field L_{out} is modeled from three aspects. First, the output appearance is an interpolation of appearance from known images. Second, appearance is assumed to be a product of shading and albedo. Third, it is assumed that the unobserved shading and albedo at x is a warp of obsessed shading and albedo at y. The whole pipeline is illustrated in Figure 1 and each step of the implementation including de-light, interpolation, warping and consistency is described in the following sections.

De-light

The functioning of de-light is to decompose appearance into a combination of shading. Then the observed image is splitted into $L_{in}(y) = E(y) \odot A(y)$, where $\odot$ is the pixel-wise product between a shading image $E(y)$ and an albedo image $A(y)$. And the shading and albedo images are interpolated independently by

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

where operator int (interpolation) will be introduced in the following.

Interpolation

All observed images are warped by interpolation and the individual results are merged. The operator int is defined as

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

where input I refers to shading E or albedo A, as they share the same operation in operator int.

Warping

Based on the observed and unobserved X-Field coordinates, warping deforms an observed image into an unobserved image by

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

Flow

The Jacobian of a specific pixel p is defined as

$$flow_{\partial}(x)[p] \ = \ \frac{\partial p(x)}{\partial x} \quad \in \ \mathcal{X} \ \rightarrow \ \mathbb{R}^{2 \times n_d}$$

where the X-Field coordinate x is the only input to the flow computation, and the above Jacobian equation is the only output. This function is implemented using CNNs in particular. And details can be found in [Bemana 20].

Consistency

Each image pixel is weighted by its flow consistency to merge all observed images warped to the unknown X-Field coordinates. The consistency of one pixel p, warped to coordinate x from y, is the partition of unity of a weight function:

$$cons(y \to x)[p] = w(y \to x)[p]\Big(\sum_{y' \in \mathcal{Y}} w(y' \to x)[p]\Big)^{-1}$$

where the weight function w is decreasing with the delta of the pixel position p and the backward flow at the position q, defined as

$$w(y \to x)[p] = \exp(-\ \sigma\,|\,p\ -\ flow_{\Delta}(y \to x[q]|_1)$$

Implementation

In my model, the original X-Fields model is improved by generalizing the input information of image datasets and adding the 3D view coordinates.

The original X-Fields model derives parameter (view, light, time, etc) values from image file names, and the information from file names implicitly contains the coordinates of view and light and steps in time by a fixed sequence of numbers. It may be not convenient for users to label the image file names manually. My program extracts information from JSON files with a specified format. In addition to being more intuitive, this change is compatible for further generalization with more parameters.

Also, I add another coordinate Z to construct a 3D view interpolation instead of the 2D one from the original model. I made sure that my extension preserves the functionality of the original X-Fields program by manually inputting the coordinates of the example "island" dataset and successfully replicating the resulting interpolation video after setting Z as a dummy variable. After exploring the tugboat dataset containing viewpoints from the surface of an entire sphere surrounding the object, I find that the interpolation results are relatively poor. This prompted me to perform more controlled experiments with more defined and less sparse viewpoints. I generated new image datasets to lessen the view change and to separate the background and foreground (in some datasets I replace the default ocean background with pure black). I also vary the resolutions of such datasets to find the resolution's impact over interpolation and video rendering.

In addition, I vary different parameters in training—including learning rate, learning stop threshold, sigma and scaling down factor—to study these parameters' impacts on results.

Results

I generated datasets containing images of a tugboat from different viewpoints using the 3DB framework. Altering factors like dataset size (dimensions per axis), viewpoint angle range (defined by data points distributed on the surface area of 1/8th versus a whole sphere), image resolution, and backgrounds (ocean and black in the case of my experiments), I study the parameters' effect on interpolation quality and computational complexity. In addition to experimenting with 3DB input rendering settings, I investigated how changing X-Fields parameters like output resolution, learning rate, learning stop threshold, sigma, and factor influenced model results and training resources consumed. Training the model on datasets of 5, 10, and then 15 dimensions per axis consistently increased quality at the tradeoff of greater necessary computational power and memory. For example, when learning rate reduces, training time increases and quality of interpolation improves.

Also, my attempt to train my revised X-Fields on a 20x20x20 dataset failed because it required more than the 10GB of memory available to me.

The biggest 3DB datasets in my experiment include 15 images with equal distance. In my parameter experiment, I first tried different learning rate values. As shown in Figure 2, the results with learning rate 0.00001 (the lowest setting) are best.

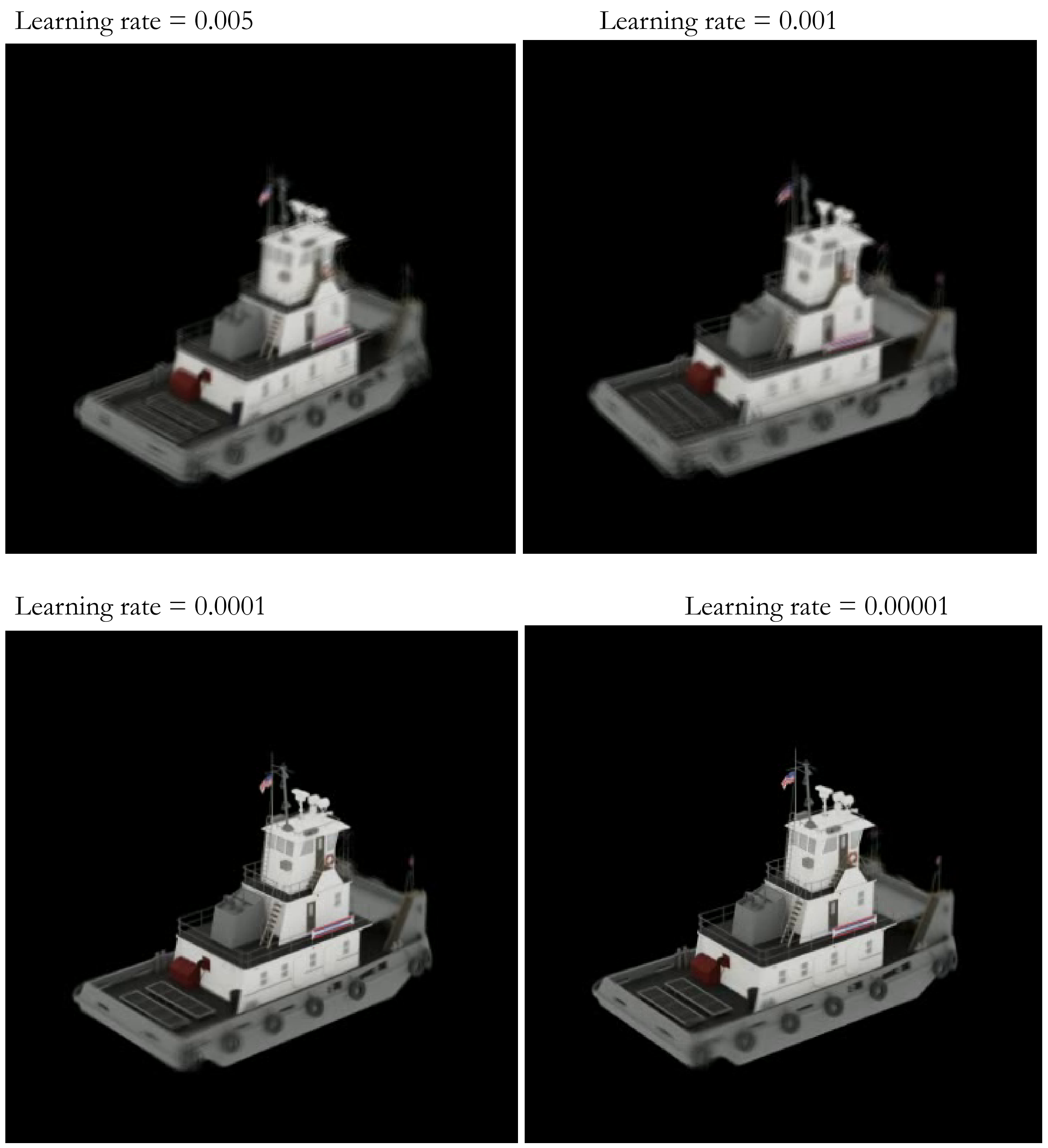


Figure 2: Sample Interpolation With Different Learning Rates

I also tried changing sigma, the bandwidth parameter in soft blending, from 0.1 to 0.5, which improved the interpolation quality. The sample interpolations are provided in Figure 3.

Sigma = 0.1 Sigma = 0.5

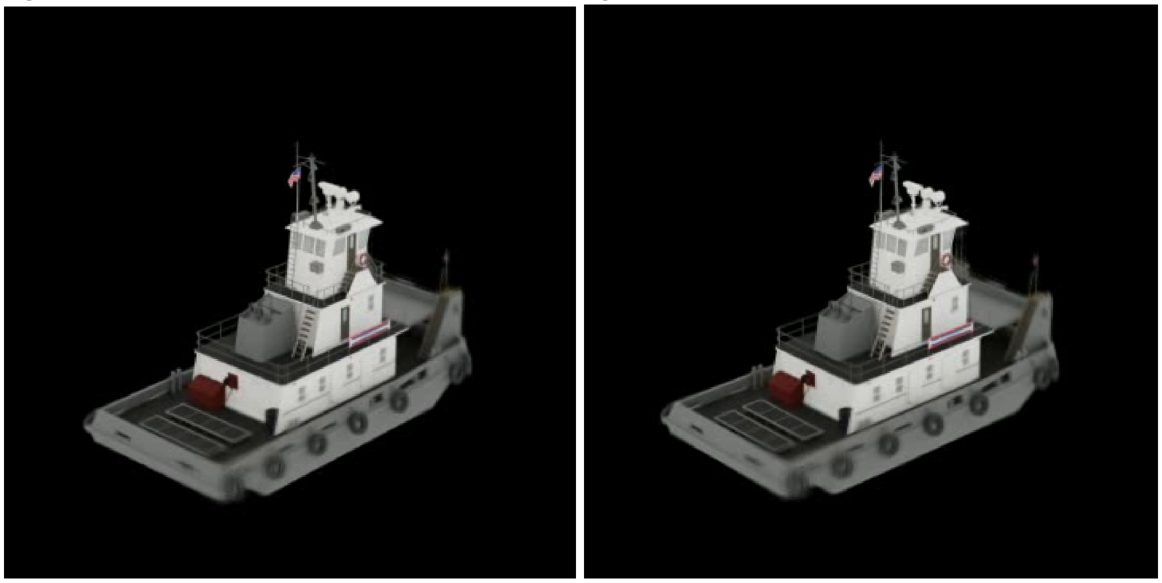


Figure 3: Sample Interpolation With Different Values of Sigma

Adding water to the reference frames increased the ghosting effect. One likely reason for this is the various unpredictable reflections on the surface of the water. Note that one reason for the images' lower quality overall (including the one of the tugboat with the black background) is that these results come from 10x10x10 datasets because using 3DB to generate renders with the water background is very computationally expensive. Sample interpolations are shown in Figure 4.

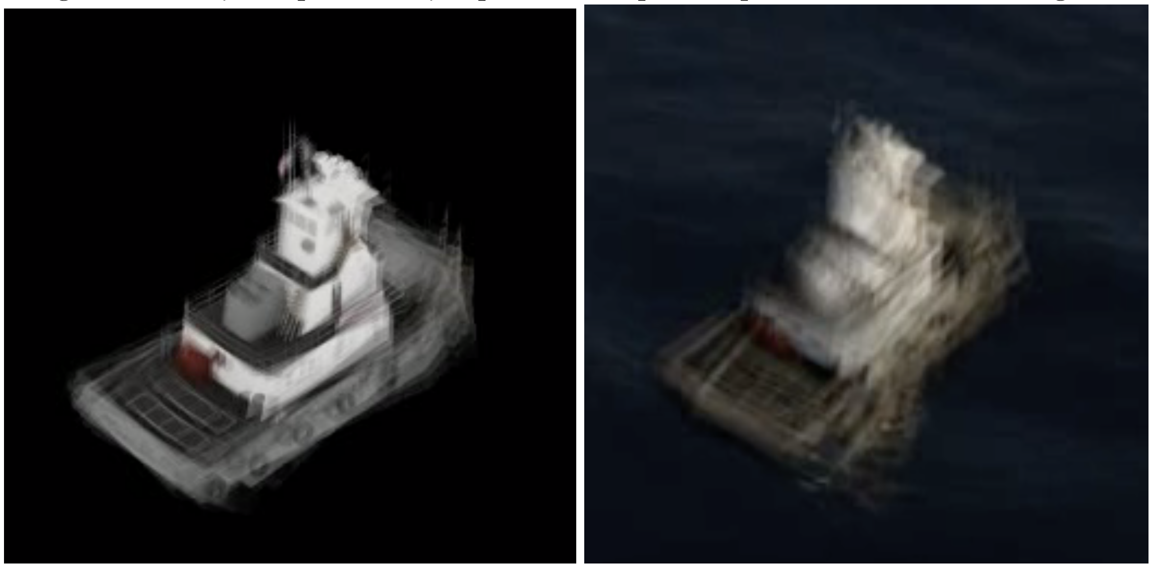


Figure 4: Sample Interpolation With Different Backgrounds

dims/axis	lr	threshold	scope	background	resolution	factor	Video resolution	sigma	nfg	num_n
5	0.0005	0.01	full	black	224	2	128	0.1	16	6
5	0.00001	0.01	positive	black	512	1	512	0.1	16	6
10	0.0005	0.01	full	black	224	2	128	0.1	16	6
10	0.0005	0.01	positive	black	224	2	128	0.1	16	6
10	0.0005	0.01	positive	water	224	2	128	0.1	16	6
15	0.005	0.001	positive	black	2160	6	360	0.1	16	6
15	0.001	0.001	positive	black	2160	6	360	0.1	16	6
15	0.0001	0.001	positive	black	2160	6	360	0.1	16	6
15	0.00005	0.001	positive	black	2160	6	360	0.1	16	6
15	0.00001	0.001	positive	black	2160	6	360	0.5	16	6
15	0.00001	0.001	positive	water	2160	6	360	0.1	16	6
15	0.00001	0.001	positive	black	2160	4	560	0.1	16	6

Table 1: Record of Different Experiments

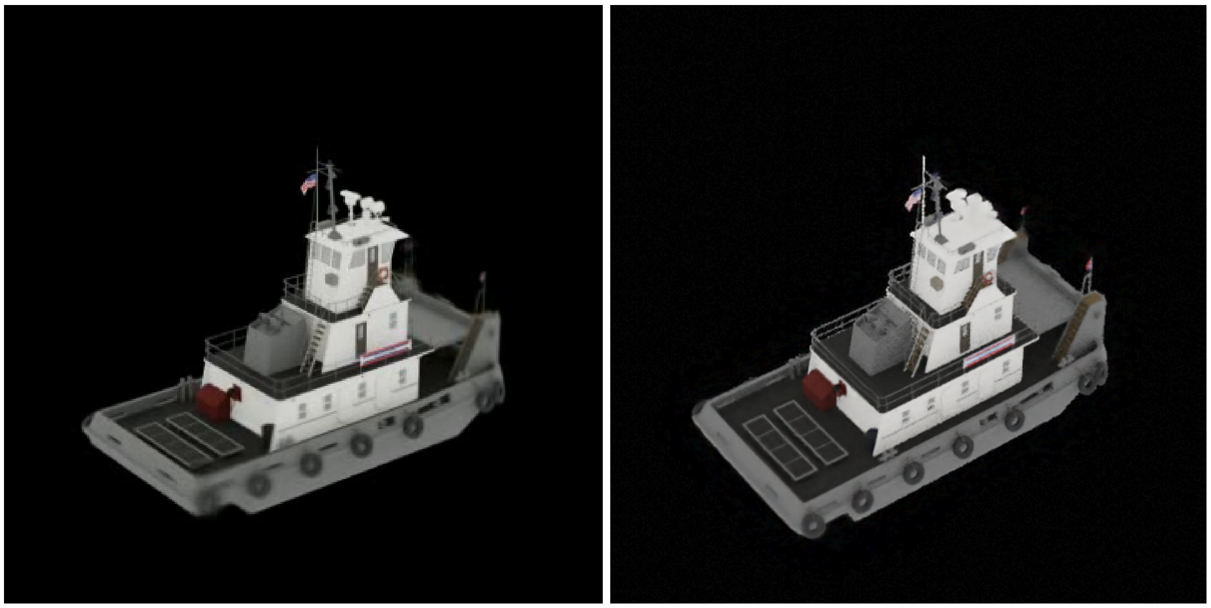
Conclusions & Future Work

In conclusion, by controlling dataset generation and adjusting separate parameters, I have found that the X-Fields model has the following capabilities and limitations:

1. It can generate promising interpolation results with relatively sparse datasets and with large view angle changes.
2. Parameters such as learning rate and the bandwidth parameter in soft blending have impacts over the interpolation quality and construct trade-offs between training cost and interpolation quality.
3. Certain backgrounds added to reference images can pose a challenge for interpolation.

Computer vision is a dynamic area of research in which interpolation is an important application. New types of neural networks, such as Transformer Network [Choi 20][Zhihao 21][Khan 21][Ye 21], can be combined with CNN to reduce training cost and improve interpolation accuracy. The combination of Transformer Networks and X-Fields hard-coded graphics knowledge is worth further investigation and experimenting. Zhizhao et al. [Zhihao 21] presents a Transformer-based video interpolation framework that addresses the problem of content-aware aggregation weights and long-range dependencies with self-attention operations. The proposed method avoids the high computational cost by use of local attention into video interpolation and the spatial-temporal domain. Choi et al. [Choi 20] proposes a novel end-to-end deep NN for video frame interpolation that synthesizes an intermediate frame effectively without the explicit estimation of motion. The empirical results show that the motion estimation can be simply replaced by a combination of PixelShuffle, parameter-free feature transformations.

I have generated some sample frames from Choi's [Choi 20] pretrained CAIN model with the default interpolation ratio of 3:1 as an initial exploration. Comparison of interpolation results with X-Fields with the same reference frames is shown below:



X-Fields CAIN

Figure 5: Comparison of X-Fields and CAIN Interpolation Samples

References

[Choi 20] Choi, M., Kim, H., Han, B., Xu, N., & Lee, K. M. (2020). Channel Attention Is All You Need for Video Frame Interpolation. Proceedings of the AAAI Conference on Artificial Intelligence, 34(07), 10663-10671. https://doi.org/10.1609/aaai.v34i07.6693 https://github.com/myungsub/CAIN

[Khan 21] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah, Transformers in Vision: A Survey, arXiv:2101.01169v2 [cs.CV] 22 Feb 2021.

[Ye 21] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, Fei Sha, Few-Shot Learning via Embedding Adaptation with Set-to-Set Functions, arXiv:1812.03664v6, 13 Jun 2021.

[Levoy 96] Levoy, M., & Hanrahan, P. (1996, August). Light field rendering. In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (pp. 31-42).

[Gortler 96] Gortler, S. J., Grzeszczuk, R., Szeliski, R., & Cohen, M. F. (1996, August). The lumigraph. In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (pp. 43-54).

[Kalantari 16] Kalantari, N. K., Wang, T. C., & Ramamoorthi, R. (2016). Learning-based view synthesis for light field cameras. ACM Transactions on Graphics (TOG), 35(6), 1-10.

[Lombardi 19] Lombardi, S., Simon, T., Saragih, J., Schwartz, G., Lehrmann, A., & Sheikh, Y. (2019). Neural volumes: Learning dynamic renderable volumes from images. arXiv preprint arXiv:1906.07751.

[Malzbender 01] Malzbender, T., Gelb, D., & Wolters, H. (2001, August). Polynomial texture maps. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques(pp. 519-528).

[Buehler 02] Buehler, C., Bosse, M., McMillan, L., Gortler, S., & Cohen, M. (2001, August). Unstructured lumigraph rendering. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques (pp. 425-432).

[Hedman 16] Hedman, P., Ritschel, T., Drettakis, G., & Brostow, G. (2016). Scalable inside-out image-based rendering. ACM Transactions on Graphics (TOG), 35(6), 1-11.

[Hedman 18] Hedman, P., Philip, J., Price, T., Frahm, J. M., Drettakis, G., & Brostow, G. (2018). Deep blending for free-viewpoint image-based rendering. ACM Transactions on Graphics (TOG), 37(6), 1-15.

[Flynn 16] Flynn, J., Neulander, I., Philbin, J., & Snavely, N. (2016). Deepstereo: Learning to predict new views from the world's imagery. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5515-5524).

[Zou 18] Zou, Y., Luo, Z., & Huang, J. B. (2018). Df-net: Unsupervised joint learning of depth and flow using cross-task consistency. In Proceedings of the European conference on computer vision (ECCV) (pp. 36-53).

[Penner 17] Penner, E., & Zhang, L. (2017). Soft 3D reconstruction for view synthesis. ACM Transactions on Graphics (TOG), 36(6), 1-11.

[Sun 18] Sun, S. H., Huh, M., Liao, Y. H., Zhang, N., & Lim, J. J. (2018). Multi-view to novel view: Synthesizing novel views with self-learned confidence. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 155-171).

[Chen 19] Chen, X., Song, J., & Hilliges, O. (2019). Monocular neural image based rendering with continuous view control. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 4090-4100).

[Bemana 20] Bemana, M., Myszkowski, K., Seidel, H. P., & Ritschel, T. (2020). X-Fields: implicit neural view-, light-and time-image interpolation. ACM Transactions on Graphics (TOG), 39(6), 1-15.

[Leclerc 21] Leclerc, G., Salman, H., Ilyas, A., Vemprala, S., Engstrom, L., Vineet, V., ... & Madry, A. (2021). 3DB: A Framework for Debugging Computer Vision Models. arXiv preprint arXiv:2106.03805.

[Zhiha 21] Zhihao Shi, Xiangyu Xu, Xiaohong Liu, Jun Chen, Ming-Hsuan Yang. (2021). Video Frame Interpolation Transformer. arXiv preprint arXiv:2111.13817v1.