3D Photography using Layered Depth Imaging (LDI) and learning-based inpainting model

Hieu Trung Le
Stanford University
450 Serra Mall, Stanford, CA 94305
leric@stanford.edu

Abstract

For the class project, we will be implementing the technique to convert a single RGB-D input image into a 3D photo that user can interact with by scrolling around and see the object in 3D. By using the technique called Layered Depth Image with explicit pixel connectivity as underlying representation and a learning-based inpainting model as introduced in [14], we will be able to synthesize new local color-and-depth content in the occluded region of the object in image in a spatial context-aware manner. With the newly synthesized color-and-depth content, we will be able to produce 3D photos that can be efficiently rendered with motion parallax using standard graphics engines. We will evaluate our method against other state of the art method using metrics such as SSIM (or structural similarity), PSNR (or peak-signal-to-noise ratio) and LPIPS (Learned Perceptual Image Patch Similarity).

1. Introduction

3D photography is a technique that allows the capturing views of the world with a camera and using image-based rendering techniques for novel view synthesis. The concept of 3D photography is not new and has been around for years. There have been a lot of interests in 3D photography over the years because it makes photos look real, as if they're just right in front of you. Comparing to 2D photography, 3D photography provides an immersive experience, almost lifelike in Virtual Reality (VR)

Traditional method of constructing 3D image requires elaborate capture setups involving many images with large baselines and special hardware such as those used in Facebook Manifold camera. Recent researches have tried to make capturing and reconstructing 3D photography more effortless using cell phone cameras that produce photos with smaller baselines. Facebook 3D Photos, for example, require capturing a single snapshot with a dual lens camera phone and can extrapolate the three-dimensional shape of objects in your image, and uses that to generate a convincing 3D effect. Facebook 3D Photos leverages the usage of layered depth image (LDI) representation. The color and depth in occluded regions are synthesized using optimized heuristics and isotropic diffusion algorithm for inpainting colors.

Other state-of-the-art techniques constructed the view synthesis algorithms by using the input images to estimate a 3D scene representation, which can then be reprojected to render novel views. This works well in certain scenario where the object is mostly visible in the input images. In other cases where the object is not entirely visible, the quality of novel views degrades rapidly as the target viewpoint moves further away from the input views and reveals the occluded scene content that the algorithms cannot generate.

For this project, we are presenting a new learning-based method that generates a 3D photo from an RGB-D input. By taking the depth from dual camera cell phone stereo or via estimation from a single RGB image, we then combine with the LDI representation introduced by Facebok 3D photos. By storing the connectivity between pixels across different pixels in the representation, we will be able to break the problem into many local sub-problems, which then can be solved iteratively. We then apply the standard CNN and fuse the inpainted regions back into the LDI and perform recursively until all depth edges are treated

2. Background/Related Work

2.1. Image-based rendering

Image-based modeling and rendering (or IBMR) methods rely on a set of two-dimensional images of a scene to generate a three-dimensional model and then render some novel views of this scene. Unlike traditional 3D computer graphics in which 3D geometry of the scene is known, image-based rendering techniques render novel views directly from input images. Image-based rendering techniques can be classified into three categories according to

how much geometric information is used: rendering without geometry, rendering with implicit geometry, and rendering with explicit geometry [15]. These methods work best when the images have sufficiently large baselines or are captured with depth sensors.

2.2. Novel view synthesis

Novel view synthesis is a technique aims at generating novel views from a single or multiple input images of an object. Traditional solutions are based on multi-view reconstruction using a geometric formulation such as [11] and [19]. These techniques take as input several views of an object. They will then build an explicit 3D representation (e.g., mesh [6], pointcloud [3], voxels map [5]) and render novel views of that object. This only works if there is input of multiple views of an object in order to construct the 3D representation. They also require some kind of 3D supervision as input

Other type of novel view synthesis also explore the usage of light fields [16] and [8], multi-plane images [20], and layered depth images [7]. Light fields enable photorealistic rendering of novel views but require multiple input image. The multiplane image representation stores multiple layers of RGB-α images at fixed depths andn has the advantage in capturing semi-reflective or semi-transparent surfaces. Layered depth images (or LDI [7]) is a view of the scene from a single input camera view and contains potentially multiple depth pixels at each discrete location in the image. When rendering from an LDI, the requested view can move away from the original LDI view and expose surfaces that were not visible in the first layer.

2.3. Learning-based View Synthesis from a Single Image

A more challenging problem of novel view synthesis is to generate novel views using just a single view of the object as input. In this case multi-view geometry cannot be leveraged, making this an intrinsically difficult problem. Deep learning, however, has made novel view synthesis possible by relying on inductive biases, similarly to what humans do. When provided with the image of an object, human have the ability to picture how it would look like from a different viewpoint by unconsciously learning an implicit model for the 3D structure of the object. They will then be able to fit the current observation to their mental model and use it to hallucinate a novel view of the car from a different angle. Several researches such as [18] and [21] have tried to explore this area by using sing deep generative model to render novel views of an object given a single input view. However, training deep generative model require expensive supervision at training time. It will also require images with camera poses or multiple views of the same object that are difficult to replicate outside of academic setting

2.4. Single-image Depth Estimation

Recent advances in deep neural networks and with the introduction of annotated depth image datasets enabled improvements in monocular depth estimation. Depth Estimation is a crucial step towards inferring scene geometry from 2D images. The goal of monocular depth estimation is to predict the depth value of each pixel, given only a single RGB image as input. More recently, it has been shown that convolutional neural network (CNN) architectures can be used to infer geometrical information solely from presented monocular RGB depth or intensity images as shown in [2]

However, depth estimation from a single image remains an open research problem. The quality of the predicted depth maps varies depending on the image type and the depth maps from existing methods are in many scenarios not suitable for generating high-quality novel view synthesis

2.5. Image inpainting

Image Inpainting is a task of reconstructing missing regions in an image. It allows us to fill in missing regions in images with plausible content. It has important application in image restoration, manipulation, re-targeting, compositing, and image-based rendering.

Existing works for image inpainting can be mainly divided into two groups. The first group represents traditional diffusion-based or patch-based methods with low-level features. The second group attempts to solve the inpainting problem by a learning-based approach.

Traditional diffusion or patch-based approaches such as [9] and [1] typically use variational algorithms or patch similarity to propagate information from the background regions to the holes. These methods work well for stationary textures but are limited for non-stationary data such as natural images.

For learning-based approach, Pathak et al 2016 [4] have used convolutional neural network trained to generate the contents of an arbitrary image region conditioned on its surroundings. The Context Encoder in [4] consists of an encoder capturing the context of an image into a compact latent feature representation and a decoder which uses that representation to produce the missing image content. Context encoders are trained in a completely unsupervised manner and it is required to both understand the content of an image, as well as produce a plausible hypothesis for the missing parts

3. Technical Approach

3.1. Method Overview

Given an input RGB-D image, our method shall be as follow:

1. initialize a trivial Layered depth image (LDI), which has single layer and is fully 4-connected

2. pre-process the image in order to detect major depth discontinuities. Once we have these measures, we then group them into simple connected depth edges

3. apply the algorithm by iteratively selecting a depth edge for inpainting and disconnect the LDI pixels across the edge and only consider the background pixels of the edge for inpainting

4. extract a local context region of the "known" side and generate a synthesis region on the "unknown" side item merge the synthesized pixels back into the LDI

5. repeat iteratively proceeds until all depth edges have been treated

3.2. Layered depth image initialization

Layered depth image (LDI) is a view of the scene from a single input camera view, but with multiple pixels along each line of sight. It is similar to a regular image, except at every position in the pixel lattice it can hold any number of pixels. Each LDI pixel will store a color and a depth value. In our we method, we explicitly represent the local connectivity of pixels by storing pointers to either zero or at most one neighbor for each pixel. LDI is useful because they can handle an arbitrary number of layers and sparse which implies that they are efficient at storing and can be converted into a light-weight textured mesh

The first step will be to initialize a trivial LDI which uses a single layer and is fully 4-connected

3.3. Image pre-processing

In this step, we will normalize the depth channel by mapping the min and max disparity values to 0 and 1. All parameters related to spatial dimensions are tuned and adjusted proportionally. First, we create a single layer and connecting every LDI pixel to its four cardinal neighbors. We then proceed to sharpen the depth maps using a bilateral median filter (Figure 1 a-d) in order to find depth discontinuities in the depth maps. This will later assist us in inpainting the occluded parts of the scene.

After sharpening the depth maps, we then proceed to find discontinuities (Figure 1e) by thresholding the disparity difference between neighboring pixels. We then clean up any isolated speckles or short segments by using binary map and labeling depth discontinuities as 1 and other as 0. By using the connected component analysis, we will be able to merge adjacent discontinuities into a collection of "linked depth edges". The final linked depth edges will be used as a component in the iterative inpainting procedure.

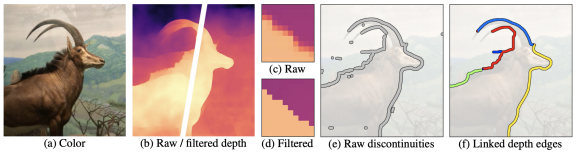


Figure 1 [14]

3.4. Extract local context region and generate synthesis region

In this step, we will apply the inpainting algorithm on the linked depth edges generated in previous step. The initial LDI is fully connected. A depth edge (discontinuity) is marked in gray in Figure 2. First we will start by disconnecting the LDI pixels across the depth or discontinuity forming a foreground silhouette (green) and a background silhouette (red). For the background silhouette we spawn a context region (blue) and a synthesis region (red) of new LDI pixels. Here we are generating a synthesis region which is extending its surrounding content into the occluded region. We then iteratively expand the context and synthesis regions alternately by stepping left/right/up/down. We then define a context region such that inpainting networks only considers the content in the context region. We then run this algorithm for 100 iterations to expand the context regions to improve the performance of synthesis.

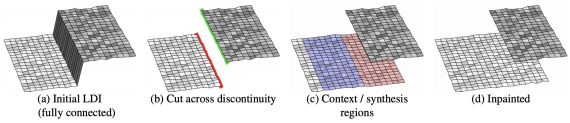


Figure 2 [14]

3.5. Apply context-aware color and depth inpainting to synthesize color and depth values

In this step, we will be synthesizing color and depth values given the context and synthesis regions from the previous step. First we break down the inpainting tasks into three sub-networks: edge inpainting network, color inpainting network and depth inpainting network. The edge inpainting network will help us predict the depth edges in the synthesis regions and infer the structure that can be used for constraining the color and depth values. The similar approach is applied to the depth inpating.

Given the color, depth, the extracted and linked depth edges as inputs, first we randomly select one of the edges from the extracted and linked depth edges as a subproblem. Using an edge inpainting network, we will start with inpainting the depth edge in the synthesis region (red). To produce the inpainted color, we then concatenate the inpainted depth edges with the context color together and apply a color inpainting network. To produce the inpainted depth, we concatenate the inpainted depth edges with the context depth and apply a depth inpainting network.

We will keep applying our inpainting model until no further inpainted depth edges are generated.

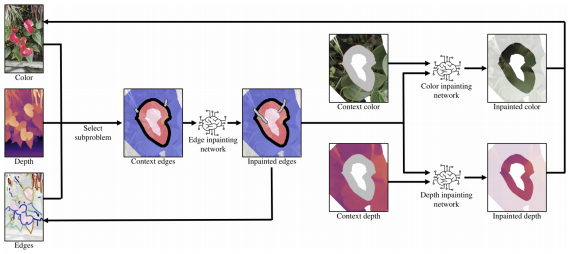


Figure 3 [14]

We then train our model on the MSCOCO dataset because of its wide diversity in object types and scenes. By obtaining the pseudo ground truth depth map via pre-trained MegaDepth, we will be able to extract context/synthesis regions and randomly place these context-synthesis regions on the images in the COCO dataset. This will allow us to obtain the ground truth content.

- training inpainting model: we will train the edge-generator model using the ADAM optimizer with $\beta = 0.9$ with different learning rate varying from 0.0001 to 0.001. We will also train both the edge and depth generator model using the same context-synthesis regions dataset

- training edge inpainting network: we will follow architecture in EdgeConnect: Generative Image Inpainting with Adversarial Edge Learning where there are two stages: 1) edge generator, and 2) image completion network and each stage consists of a generator/discriminator pair. The edge generator network architecture can be found in Figure 4

Edge Generator

Module	Filter Size	#Channels	Dilation	Stride	Norm	Nonlinearity
Conv1	7×7	64	1	1	SN→IN	ReLU
Conv2	4×4	128	1	2	SN→IN	ReLU
Conv3	4×4	256	1	2	SN→IN	ReLU
ResnetBlock4	3×3	256	2	1	SN→IN	ReLU
ResnetBlock5	3×3	256	2	1	SN→IN	ReLU
ResnetBlock6	3×3	256	2	1	SN→IN	ReLU
ResnetBlock7	3×3	256	2	1	SN→IN	ReLU
ResnetBlock8	3×3	256	2	1	SN→IN	ReLU
ResnetBlock9	3×3	256	2	1	SN→IN	ReLU
ResnetBlock10	3×3	256	2	1	SN→IN	ReLU
ResnetBlock11	3×3	256	2	1	SN→IN	ReLU
ConvTranspose12	4×4	128	1	2	SN→IN	ReLU
ConvTranspose13	4×4	64	1	2	SN→IN	ReLU
Conv14	7×7	1	1	1	SN→IN	Sigmoid

Discriminator

Module	Filter Size	#Channels	Dilation	Stride	Norm	Nonlinearity
Conv1	4×4	64	1	2	SN	LeakyReLU(0.2)
Conv2	4×4	128	1	2	SN	LeakyReLU(0.2)
Conv3	4×4	256	1	2	SN	LeakyReLU(0.2)
Conv4	4×4	512	1	1	SN	LeakyReLU(0.2)
Conv5	4×4	1	1	1	SN	Sigmoid

Figure 4 [14]

- training depth and color inpainting networks: we will use a standard U-Net architecture with partial convolution. The architecture of this U-Net Architecture is shown in Figure 5

Module	Filter Size	#Channels	Dilation	Stride	Norm	Nonlinearity
PConv1	7×7	64	1	2	-	ReLU
PConv2	5×5	128	1	2	BN	ReLU
PConv3	5×5	256	1	2	BN	ReLU
PConv4	3×3	512	1	2	BN	ReLU
PConv5	3×3	512	1	2	BN	ReLU
PConv6	3×3	512	1	2	BN	ReLU
PConv7	3×3	512	1	2	BN	ReLU
PConv8	3×3	512	1	2	BN	ReLU
NearestUpsample	-	512	-	2	-	-
Concatenate (w/ PConv7)	-	512+512	-	-	-	-
PConv9	3×3	512	1	1	BN	LeakyReLU(0.2)
NearestUpsample	- 512	-	2	-	-	
Concatenate (w/ PConv6)	-	512+512	-	-	-	-
PConv10	3×3	512	1	1	BN	LeakyReLU(0.2)
NearestUpsample	-	512	-	2	-	-
Concatenate (w/ PConv5)	-	512+512	1	-	-	-
PConv11	3×3	512	1	1	BN	LeakyReLU(0.2)
NearestUpsample	-	512	-	2	-	-
Concatenate (w/ PConv4)	-	512+512	-	-	-	-
PConv12	3×3	512	1	1	BN	LeakyReLU(0.2)
NearestUpsample	-	512	-	2	-	-
Concatenate (w/ PConv3)	-	512+256	-	-	-	-
PConv13	3×3	256	1	1	BN	LeakyReLU(0.2)
NearestUpsample	-	256	-	2	-	-
Concatenate (w/ PConv2)	-	256+128	-	-	-	-
PConv14	3×3	128	1	1	BN	LeakyReLU(0.2)
NearestUpsample	-	128	-	2	-	-
Concatenate (w/ PConv1)	-	128+64	-	-	-	-
PConv15	3×3	64	1	1	BN	LeakyReLU(0.2)
NearestUpsample	-	64	-	2	-	-
Concatenate (w/ Input)	-	64 + 4 or 64 + 6 (Depth / Color Inpainting)	-	-	-	-
PConv16	3×3	1 or 3 (Depth / Color Inpainting)	1	1	-	-

Figure 5 [14]

To train our color inpainting model, we adopt objective functions as follow. First, we define the reconstruction loss for context and synthesis regions

$$L_{\text{synthesis}} = \frac{1}{N}||S \odot (I - I_{gt})||, \qquad L_{\text{context}} = \frac{1}{N}||C \odot (I - I_{gt})||.$$

In these equations, S and C are the binary mask indicating synthesis and context regions, N is the total number of pixels, I is the inpainted result, and I_{gt} is the ground truth image. $\odot$ denotes the Hadamard product.

Next we define the perceptual loss:

$$L_{perceptual} = \sum_{p}^{P-1} \frac{||\psi_p(I) - \psi_p(I_{gt})||}{N_{\psi_p}}$$

In this equation $\Psi_p(.)$ is the output of the pth layer and N_{Ψ_p} is the total number of elements in $\Psi_p(.)$.

The style loss is defined as:

$$L_{style} = \sum_{p}^{P-1} \frac{1}{C_p C_p} || \frac{1}{C_p H_p W_p} \left[(\psi_p^I)^\top \psi_p^I - (\psi_p^{I_{gt}})^\top \psi_p^{I_{gt}} \right] ||$$

where C_p, H_p, W_p is the number of channels, height, and width of the output $\Psi_p(.)$

The total variation loss is defined as:

$$L_{tv} = \sum_{(i,j) \in S, (i,j+1) \in S} \frac{||I(i,j+1) - I(i,j)||}{N} + \sum_{(i,j) \in S, (i+1,j) \in S} \frac{||I(i+1,j) - I(i,j)||}{N}.$$

By combining all ob these, we obtain the objective loss function for the color impanting network

$$L = L_{context} + 6L_{synthesis} + 0.05L_{perceptual} + 120L_{style} + 0.01L_{tv}$$

For the objective loss function for the depth inpainting model, we use the equation $L_{context} + L_{synthesis}$

3.6. Merge the synthesized pixels back into the LDI and converting to 3D textured mesh

The final step will be integrating all the inpainted depth and color values back into the original LDI to form the 3D textured mesh. Using this 3D generated textured mesh, we will be able to render rendering allows us render novel views without the need to perform inference. This 3D textured mesh can be rendered using standard graphics engines.

4. Experiment

The problem can be described as follow: Given an input RGB-D image, produce a 3D photo with synthesized texture and structures in occluded regions. In another word, we will be synthesizing novel view in the image. Such photo will allow user to scroll around and interact with in order to see the object in 3D. We then sample one pair of regions for each image, and resize it during training.

4.1. Dataset

The dataset we will be using is the 118k images from COCO 2017 set for training. We will select at most 3 pairs of regions from each image to form the context-synthesis pool. We then sample one pair of regions for each image, and resize it during training.

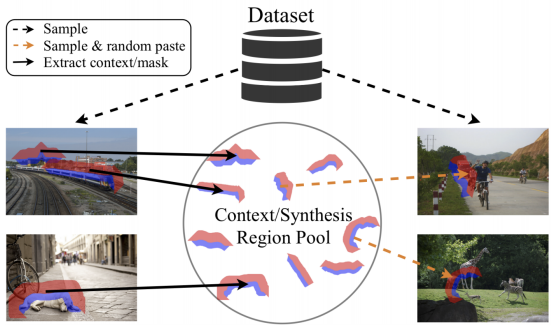


Figure 4 [14]

4.2. Evaluation

We quantify the performance of our model vs other models using SSIM (or structural similarity, which is used for measuring the similarity between two images) and PSNR (or peak-signal-to-noise ratio for comparing which method provides a better quality image) metrics between the synthesized target views and the ground truth. We will also use LPIPS (Learned Perceptual Image Patch Similarity) to quantify how well does the generated view align with human perception. We will also perform ablation studies to see how each of the components contribute positively or negatively to the final performance.

4.3. Results

We will obtain Quantitative comparison on randomly sample 1500 video sequences from RealEstate10K dataset. We expect our method to performs competitively on SSIM and PSNR as well as LPIPS comparing to other novel MPI-based approaches such as PB-MPI, LLFF, Stereo Magnification and Xview on RealEstate10K dataset

After finishing those set up and training, we were able to perform quantitative comparisons by comparing the performance of our model against other proposed methods on the RealEstate10K dataset. We use 3 well known metrics as mentioned earlier: SSIM (structural similarity), PSNR (peak-signal-to-noise ratio). As these metrics do not capture the perceptual quality of the synthesized view, we include LPIPS (Learned Perceptual Image Patch Similarity) to quantify how well does the generated view align with human perception. The methods we are comparing to are: Stereo Mag [20] (or Stereo magnification: Learning view synthesis using multiplane images), PB-MPI [12](or Pushing Boundary Multiplane Image) and LLFF [10] (or Local light field fusion), Multiview+depth (MVD) encoding and Layered Depth Images (LDI) combined [13] and DIBR (Depth-image-based rendering) [17]

Methods	SSIM	PSNR	LPIPS
PB-MPI[12]	0.8367	24.78	0.0782
Stereo Mag[20]	0.8567	25.91	0.0678
LLFF [10]	0.8102	23.21	0.0716
MVD + LDI [13]	0.8209	24.12	0.0786
DIBR[17]	0.8409	22.32	0.0884
Ours method	0.8978	27.73	0.0638

Table 1

4.4. Analysis

By reading from Table 1, we can see that our method performs competitively comparing to other method in term of SSIM and PSNR. This means that our method produce result with high similarity with the ground truth as well as higher quality image overall.

Our method doesn't perform quite as well with other method in term of LPIPS metrics (or Learned Perceptual Image Patch Similarity). This means that the generated

view of our model does not align with human perception as compared with other models.

We think there are several reason that our LPIPS metrics are performing poorly:

- estimation of depth/disparity map from a single image remain a challenging problem
- our model sometimes fails to produce satisfactory results for scenes with complex
- our model is unable to handle reflective/transparent surfaces well structures

In the next step, we hope to improve the performance of our model by using depth edge as guidance and dilation heuristic as well as color inpainting model to improve performance in disocculded regions.

Here we conduct an ablation study to better understand how each proposed components contribute to final performance. We will sample 200 triplets from testing sequences and evaluate the effect of Using depth edge as guidance as well as using inpainted color on both the entire image and disoccluded regions. The result is shown below

Methods	SSIM	PSNR	LPIPS
Inpaint w/o edge	0.7295	25.52	0.082
Inpaint w edge	0.8232	26.34	0.085

Table 2

Methods	SSIM	PSNR	LPIPS
Inpaint w/o color	0.8426	24.24	0.078
Inpaint w color	0.8829	25.39	0.089

Table 3

From the data in Table 2, we have learned that using the edge-guided inpainting leads to minor improvement in quantitative metrics. In Table 3, we have learned that our model benefits from the efficacy of color inpainting model and result in better performance

4.5. Qualitative comparison

In this section, we provide qualitative comparison between the output of our method comparing to other method

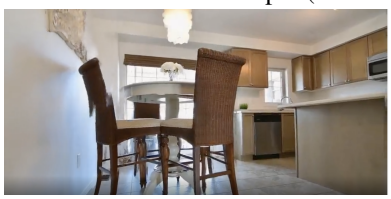
Input image:

Our method's output (screen capture from video)

Zhou et al 2018 's output (screen capture from video)

Srinivasan et al 2019 's output (screen capture from video)

Mildenhall et al 2019 's output (screen capture from video)



Choi et al 2019's output (screen capture from video)

Wiles et al 2020's output (screen capture from video)

As noticed from the screen capture above, our result was able to show smooth image transition where other method had problem showing smooth transition and had some visual artifacts during transition between different frame

5. Conclusion

In this project, we have learned about the algorithm for creating 3D photography from just a single RGB-D image. By creating a completed layered depth image (LDI) repre-

sentation through context-aware color and depth inpainting, we were able to synthesize new pixel and produce a real 3D photo that user can scroll and interact with. We also vaidate against a variety of everyday scene and photo and obtains amazing result. The experiment conducted against other state of the art method also show that the algorithm performs competitively while having fewer visual artifacts.

The next step would be to train the model on larger data set with more objects to validate the efficacy of the model in different environment

6. Link to github repo

`https://github.com/leeric92/cs231a-finalproject`

7. Contributions & Acknowledgements

This report make use of this public code repo `https://github.com/vt-vl-lab/3d-photo-inpainting` which makes available the implementation of 3D photo generation process with learning-based inpainting model that iteratively synthesizes new local color-and-depth content [14]

References

[1] V. C. G. S. C. Ballester, M. Bertalmio and J. Verdera. Filling-in by joint interpolation of vector fields and gray levels. *IEEE*, 2001.

[2] C. Z. Y. T. F. Q. Chaoqiang Zhao, Qiyu Sun. Monocular depth estimation based on deep learning: An overview. *arXiv:2003.06620v2*, 2020.

[3] C. K. Chen-Hsuan Lin and S. Lucey. Learning efficient point cloud generation for dense 3d object reconstruction. *AAAI*, 2018.

[4] J. D. T. D. A. A. E. Deepak Pathak, Philipp Krahenbuhl. Context encoders: Feature learning by inpainting. *CVPR*, 2016.

[5] P. Henderson and V. Ferrari. Learning single-image 3d reconstruction by generative modelling of shape, pose and shading. *International Journal of Computer Vision*, 2019.

[6] S. S. S. L. A. E. Jhony K Pontes, Chen Kong and C. Fookes. Image2mesh: A learning framework for single image 3d reconstruction. *Asian Conference on Computer Vision*, 2018.

[7] L.-w. H. Jonathan Shade, Steven Gortler and R. Szeliski. Layered depth images. *In Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 1998.

[8] M. Levoy and P. Hanrahan. Light field rendering. *In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996.

[9] V. C. M. Bertalmio, G. Sapiro and C. Ballester. Image inpainting. *In Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000.

[10] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019.

[11] J. M. Paul E. Debevec, Camillo J. Taylor. Modeling and rendering architecture from photographs: A hybrid geometry and image-based approach. *SIGGRAPH*, 1996.

[12] J. T. B. R. R. R. N. N. S. Pratul P. Srinivasan, Richard Tucker. Pushing the boundaries of view extrapolation with multiplane images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[13] J. A. G. C. F. Rafael dos Anjos, João Madeiras Pereira. Multiview layered depth image. In *Journal of WSCG*, 2017.

[14] M.-L. Shih, S.-Y. Su, J. Kopf, and J.-B. Huang. 3d photography using context-aware layered depth inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[15] H.-Y. Shum and S. B. Kang. A review of image-based rendering techniques. *Microsoft research*, 2020.

[16] R. S. Steven J Gortler, Radek Grzeszczuk and M. F. Cohen. The lumigraph. *SIGGRAPH*, 1996.

[17] R. O. . M. S. Suryanarayana M. Muddala. Spatio-temporal consistent depth-image-based rendering using layered depth image and inpainting. In *EURASIP Journal on Image and Video Processing*, 2016.

[18] Y.-C. C. SXiaogang Xu and J. Jia. View independent generative adversarial network for novel view synthesis. *IEEE*, 2019.

[19] S. M. S. B. C. J. D. D. S. R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. *IEEE*, 2006.

[20] J. F. G. F. N. S. Tinghui Zhou, Richard Tucker. Stereo magnification: Learning view synthesis using multiplane images. *SIGGRAPH*, 2018.

[21] Y. L. C. Y. S. L. Xiaofeng Liu, Tong Che and J. You. Auto3d: Novel view synthesis through unsupervisely learned variational viewpoint and global 3d representation. *ECCV*, 2020.