Context Aware Layered Depth Inpainting for 3D photography

Jimut Bahan Pal[1] , Anal Bera[1].
M.Sc. 1st year, Computer Vision Course.
Instructor : Tamal Maharaj[1].

[1]Department of Computer Science.
Ramakrishna Mission Vivekananda Educational and Research Institute.

⚠️ DISCLAIMER: The original paper is

3D Photography using Context-aware Layered Depth Inpainting

We contributed very little to the existing literature. The authors are:

Meng-Li Shih[1,2] Shih-Yang Su[1] Johannes Kopf[3] Jia-Bin Huang[1]

[1]Virginia Tech [2]National Tsing Hua University [3]Facebook

Spoiler...

Table of Contents

Introduction

Overview

Conversion of a single RGB image to its 3D depth variant

The RGB-D variant gives better results, but it generally needs high resolution **dual** camera to capture the depth map.

Use of parallax to create fake 3D environment from single 2D image

⏩ We use parallax to create fake hallucinated effect. We also use inpainting to paint areas of occlusion, for better view.

⏩ Learning based inpainting model, which synthesizes new local color and depth content into occluded region in a spatial context aware manner.

⏩ Photos can be rendered using 3D graphics engines which is portable, and gives better results than other existing methods with low computation cost.

Motivation

Why do we need 3D photography?

- 3D photography is fascinating way to record and reproduce visual perception.
- More impressive experience than 2D photography - just like color from b/w photos
- Extension of 3D photography can create virtual reality

Normal flat surfaces when displayed with a parallax can cause 3D effects

Classic techniques to capture 3D images requires too much cameras from different angles and precision

We can use Graphics Interchange Format (GIFs) to provide standard experience with very less computation

Recent Novel Methods

(a) Depth-warping (holes) (b) Depth-warping (stretching) (c) Facebook 3D photo (d) Our result

Figure 1. **3D photography from a single RGB-D image.** Naïve methods either produce holes (a) or stretch content (b) at disocclusions. Color and depth inpainting using diffusion is better, but provides a too smooth appearance (c). Our approach is capable of synthesizing new color/depth texture and structures, leading to more photorealistic novel views (d).

Recent Novel Methods

Facebook's 3D Photo uses a Layered Depth Image (LDI) representation

- This can also be converted to light weight mesh representation for rendering
- Color and depth from occlusion are synthesized by heuristics
- Unable to extrapolate structures and produce overly smooth results
- Several learning based methods use similar multi-layer representations.

Other methods use fixed numbers of predefined layers for each pixel

- This can be problematic, abrupt change of depth discontinuities in a picture.
- Can cause holes, uneven stretching and unpleasant results
- This method surpases all the recent methods!

Overview of the Algorithm

- **Breaking of the picture to subproblems to iteratively solve inpainting.**
- **Use of CNN, hence breaking makes this procedure much easier.**
- **Spatially adaptive context region to condition for the inpainting model.**
- **After synthesis we fuse the images back to LDI.**
- **Good results for synthesized textures in occluded regions.**
- **The algorithm determines the number of layers by adapting to the input image.**
- **All these factors lead to the creation of automated layer extraction algorithm for better results from the current competitors**

Related Works

Few Methods to create 3D photography

Representation of Novel View Synthesis

Light fields - enable photorealistic views, but requires many input photos to get good results

Multi-plane - multiple layers of RGB-α images at fixed depth, which captures semi reflective and transparent surfaces

Layered Depth images -

→ Earlier methods used fixed number of layers.

→ This work uses auto-capture of any number of layers preserving depth discontinuities.

→ This can be converted to textured mesh for efficient rendering.

Few Methods to create 3D photography

Image based rendering

Light fields - enable photorealistic views, but requires many input photos at multiple angles to get good results, just like 3D reconstruction

Requires large baseline for multi-view stereo algorithms to work well

Recent advances of these types of work includes -

→ learning based blending, soft 3D reconstruction.

→ Handling reflections, relighting.

→ Reconstructing mirror and glass surfaces.

Few Methods to create 3D photography

Learning based view synthesis

CNN based methods have been applied to synthesize novel views from sparse light field data

Several new methods have been proposed but they often focus on single specific image, synthetic 3D scene/ objects, specific view hallucination assuming piecewise planar scenes

Requires running of a forward pass of the pre trained network to synthesize the image of a given viewpoint, which is computationally expensive.

This (our) method can be embedded to arm processor or mobile devices too!

Few Methods to create 3D photography

Image inpainting

Refers to the filling of the missing regions of images with plausible content.

General inpainting refers to the addition of patches of known regions to unknown regions by belief propagation algorithm (Markov Models etc.)

Procedure for the inpainting is as follows:

⏩ We apply the inpainting locally around each depth discontinuity with adaptive hole and context regions.

⏩ We inpaint the depth values and discontinuities in the missing regions

Few Methods to create 3D photography

Depth inpainting

Refers to filling missing depth values in general

- ⏩ When commodity grade dual cameras fails
- ⏩ Image editing tasks such as objects removal on stereo images.

Our algorithm inpaint the depth of visible surfaces

It mainly focus on recovering the depth of occluded or hidden surfaces.

Few Methods to create 3D photography

CNN-based single depth estimation

⏩ Recently showed promising results in creating depth from single image.

⏩ Due to the scarcity of available datasets, these focussed on specific visual domains such as indoor scenes or street views

⏩ The accuracy of these algorithms is not yet competitive to the muti-view stereo algorithms

⏩ The field is quite promising because many datasets will be available in future i.e., 3D movies, synthetic data, depth annotations and stereo dataset

⏩ We obtain a depth estimation using a pre-trained model

⏩ Removing the dependency such as multiple images in case of stereo this method is applicable to a wide variety of images.

Materials and Methods

Layered Depth Image

⏩ The method takes a single RGB-D image, i.e., aligned depth and color image pair and generates a Layered Depth Image (LDI), with in-painted color and depth of occluded parts.

⏩ LDI is similar to regular 4 connected image except:

→ It can hold any number of pixels from zero to many in every position of the pixel lattice

→Pixels does not have neighbours across depth discontinuities

⏩ Each LDI pixels stores colour and a depth value

⏩ They are used in 3D photography because they can naturally handle any number of layers

⏩ They are sparse and memory efficient and can be converted to lightweight textured mesh for fast 3D rendering

Methods

These are the list of processes involved for creation of the final results!

- Image Preprocessing
- Context and Synthesis Regions
- Context aware color and Depth Inpainting
  - Model
  - Multi layer inpainting
  - Training data generation
- Converting to 3D textured mesh

Image Preprocessing

⏩ Single RGB-D images ← Input

⏩ We normalize the depth channel by mapping the min and max disparity values

⏩ We lift the image to LDI, by creation of a single layer everywhere and connecting each pixels to its four cardinal neighbours

⏩ We need to find the depth discontinuities since we need to inpaint the existing content - but they are usually blurred by existing stereo methods (dual camera etc.)

⏩ We sharpen the image, using bilateral median (non-linear, edge preserving and noise reducing) filter 7x7 window size $\sigma_{spatial} = 4.0$ and $\sigma_{intensity} = 0.5$

⏩ After sharpening the depth map, we find discontinuities by thresholding the disparity difference between neighbouring pixels

Image Preprocessing

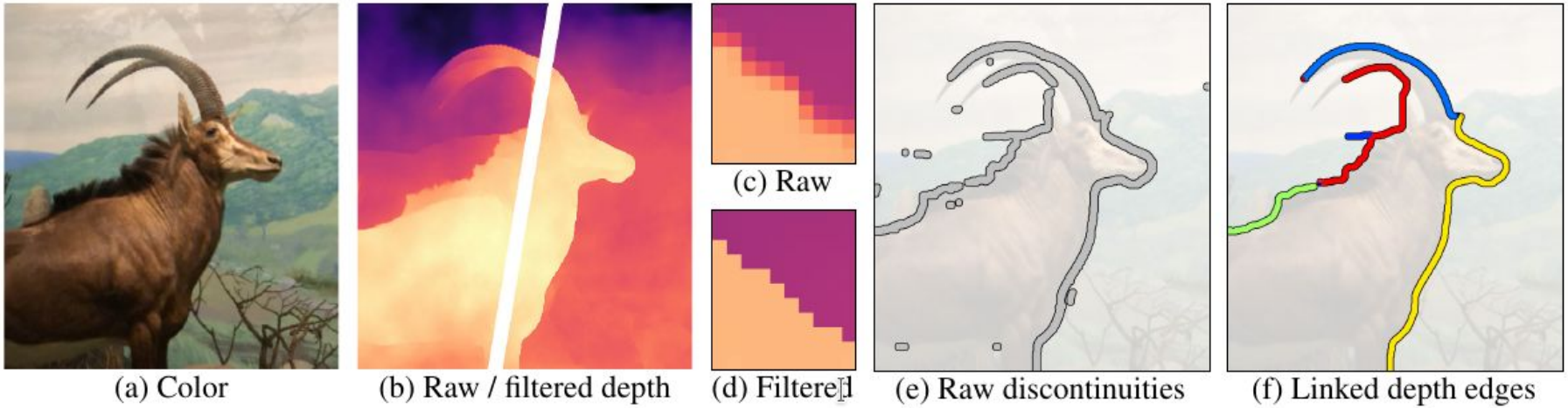


Figure 2. **Preprocessing.** Preprocessing of the color and depth input (a-b). We use a bilateral median filter to sharpen the input depth maps (c-d), detect raw discontinuities using disparity thresholds (e), and clean up spurious threshold responses and link discontinuities into connected depth edges (f). These linked depth edges form the basic unit for our inpainting process.

⏩ This results in isolated speckles as shown in Figure 2(e).

⏩ This can be cleaned up using connected component analysis to merge adjacent discontinuities into a collection of “linked depth edges”.

Image Preprocessing

⏩ To avoid merging edges at the junctions, we separate them based on local connectivity of the LDI.

⏩ Finally we remove short segments (<10 pixels) including both isolated and dangling ones

⏩ The final edges are shown in Figure 2(f) by the application of the iterative algorithm

Context and Synthesis Regions

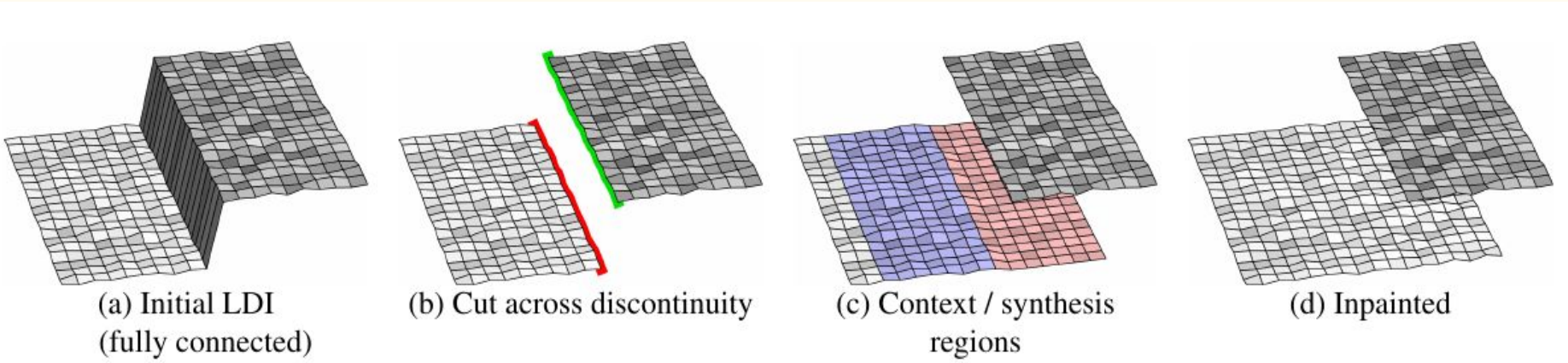


Figure 3. **Conceptual illustration of the LDI inpainting algorithm.** (a) The initial LDI is fully connected. A depth edge (discontinuity) is marked in gray. (b) We first cut the LDI pixel connections across the depth, forming a foreground silhouette (green) and a background silhouette (red). (c) For the background silhouette we spawn a context region (blue) and a synthesis region (red) of new LDI pixels. (d) The synthesized pixels have been merged into the LDI.

⏩ Inpainting algorithm operates at one of the previously computed depth edges at a time

⏩ Given one of these edges as in Figure 3(a), we need to synthesize new color and depth content in the adjacent occluded region.

Context and Synthesis Regions

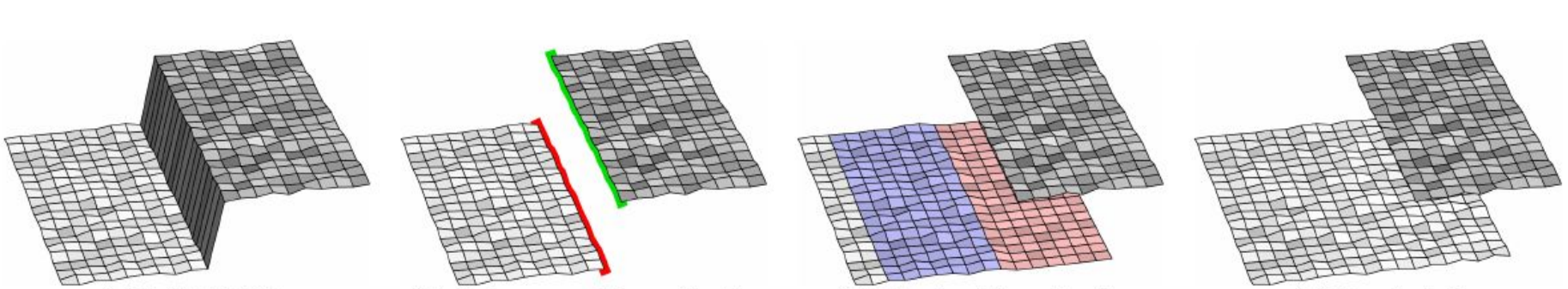


Figure 3. **Conceptual illustration of the LDI inpainting algorithm.** (a) The initial LDI is fully connected. A depth edge (discontinuity) is marked in gray. (b) We first cut the LDI pixel connections across the depth, forming a foreground silhouette (green) and a background silhouette (red). (c) For the background silhouette we spawn a context region (blue) and a synthesis region (red) of new LDI pixels. (d) The synthesized pixels have been merged into the LDI.

⏩ We start by disconnecting the LDI pixels across discontinuity Fig. 3(b)

⏩ We call the pixels silhouette pixels which have missing neighbors

⏩ The background silhouette needs inpainting, by extending the surrounding content to occluded region

Context and Synthesis Regions

⏩ We start by generating a synthesis region, a contiguous region of new pixels (Figure 3c, red pixels).

⏩ These are essentially just 2D pixel coordinates at this point.

⏩ We initialize the color and depth values in the synthesis region using a simple iterative flood-fill like algorithm

⏩ It starts by stepping from all silhouette pixels one step in the direction where they are disconnected. These pixels form the initial synthesis region.

⏩ We then iteratively expand (for 40 iterations) all pixels of the region by stepping left/right/up/down and adding any pixels that have not been visited before.

Context and Synthesis Regions

⏩ For each iteration, we expand the context and synthesis regions alternately and thus a pixel only belong to either one of the two regions.

⏩ Additionally, we do not step back across the silhouette, so the synthesis region remains strictly in the occluded part of the image. Fig (4) shows a few examples.

Figure 4. **Context/synthesis regions.** Context regions (blue) and synthesis regions (red) for three example connected depth edges (black) from Figure 2(f).

Context and Synthesis Regions

⏩ Many technologies uses this inpainting to fill holes in images based on known surrounding content, but in our method we may not find any known content

⏩ In our case inpainting is performed on a connected layer of an LDI pixels, and it should only be constrained by surrounding pixels that are directly connected to it.

⏩ Any foreground or background layer is entirely irrelevant for this synthesis unit and should not constraint or influence it in any way.

⏩ We achieve this by explicitly defining a context region (Figure 3c) for synthesis. Inpainting method only considers this region and doesn't see any other part of LDI.

⏩ The context region is generated using a similar flood-fill like algorithm.

Context and Synthesis Regions

⏩ One difference, however, is that this algorithm selects actual LDI pixels and follows their connection links, so the context region expansion halts at silhouettes.

Input context/synthesis w/o dilation w/ dilation

Figure 5. **Handling imperfect depth edges.** As the detected depth edges may not align well around occlusion boundaries, we dilate the synthesis region by 5 pixels. This strategy helps reduce artifacts in the inpainted regions.

Context and Synthesis Regions

⏩ We run this algorithm for 100 iterations, as we found that synthesis performs better with slightly larger context regions.

⏩ In practice, the silhouette pixels may not align well with the actual occluding boundaries due to imperfect depth estimation.

⏩ To tackle this issue, we dilate the synthesis region near the depth edge by 5 pixels (the context region erodes correspondingly). Figure 5 shows the effect of this heuristic.

Context aware color and depth-inpainting

Model

Given the context and regions our next goal is to synthesize the color and depth values

Even though we perform the synthesis on an LDI, the extracted context and synthesis regions are locally like images, so we can use standard network architectures designed for images.

One straightforward approach is to inpaint the color image and depth map independently.

The in-painted depth map, however, may not be well-aligned with respect to the inpainted color.

Context aware color and depth-inpainting

Model

To address the issue, we break down the architecture of the model into 3 parts (shown in Figure 6, next page):

Edge in-painting network

Color in-painting network

Depth in-painting network

Given the context as input we use the edge inpainting network to predict the depth edges in synthesis regions, producing the inpainting edges

Performing this step first helps infer the structure (in terms of depth edges) that can be used for constraining the content prediction (the color and depth values).

Context aware color and depth-inpainting

Model

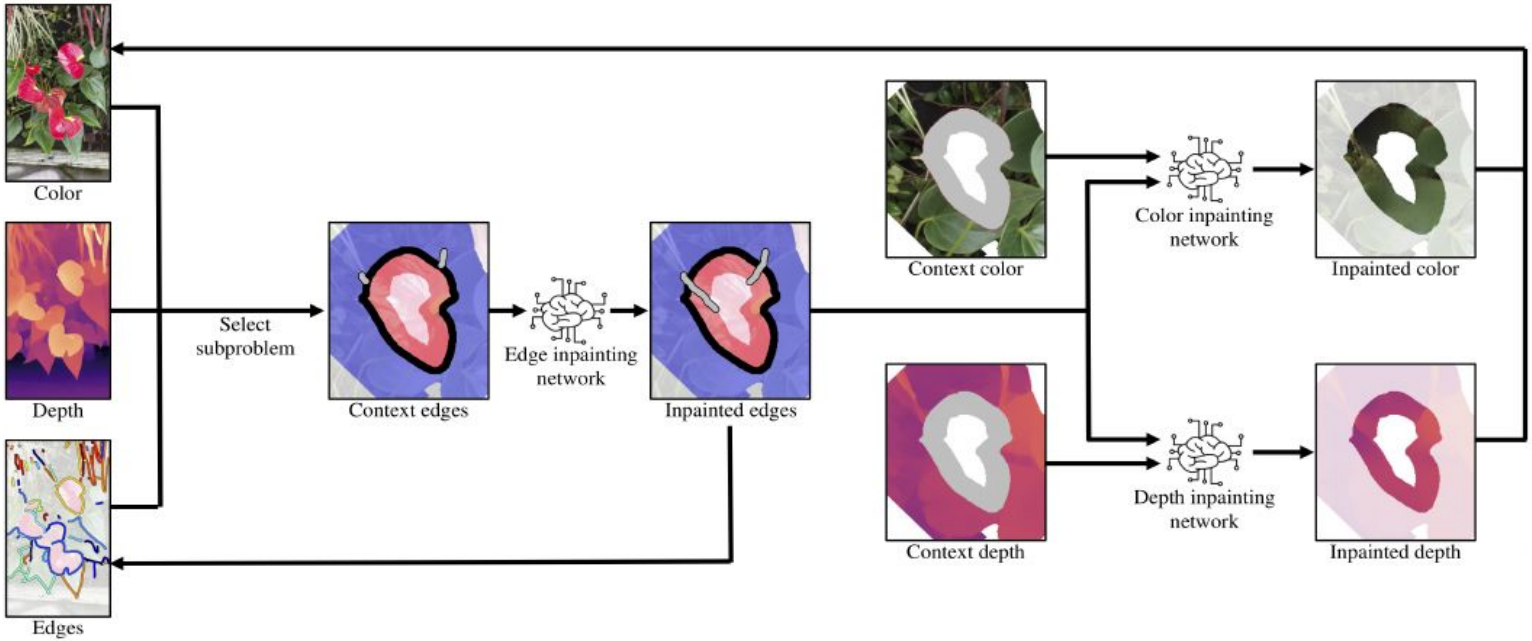


Figure 6. **Context-aware color and depth inpainting.** Given the color, depth, the extracted and linked depth edges as inputs, we randomly select one of the edges as a subproblem. We start with inpainting the depth edge in the synthesis region (red) using an *edge inpainting network*. We then concatenate the inpainted depth edges with the context color together and apply a *color inpainting network* to produce the inpainted color. Similarly, we concatenate the inpainted depth edges with the context depth and apply a *depth inpainting network* to produce the inpainted depth.

Context aware color and depth-inpainting

Model

We take the concatenated in-painted edges and context color as input and use the color inpainting network to produce inpainted color.

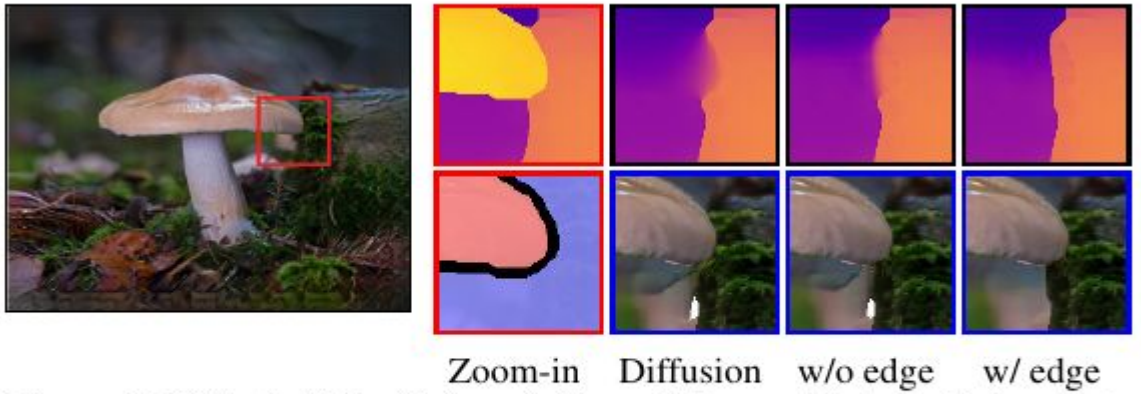


Figure 7. **Effect of depth inpainting.** Edge-guided depth inpainting produces more accurate structure inpainting, particularly for depth-complex regions (e.g., T-junctions). Blue box: synthesized novel view.

Context aware color and depth-inpainting

Model

Figure 7 shows an example of how the edge-guided inpainting is able to extend the depth structures accurately and alleviate the color/depth misalignment issue.

Multi-layer inpainting

In depth-complex scenarios, applying our inpainting model once is not sufficient as we can still see the hole through the discontinuity created by the inpainted depth edges.

Hence we apply our model until no further inpainting depth images are generated

Context aware color and depth-inpainting

Multi-layer inpainting

Here applying our inpainting model once fills the missing layers.

Several holes are still visible when viewed from a certain viewpoint

Applying inpainting model one or two times fixes the artifacts iteratively.

Figure 8 shows the process of inpainting

Figure 8. **Multi-layer inpainting.**

Context aware color and depth-inpainting

Training data generation

The model can train on any given dataset with proper annotated data

MSCOCO dataset was used for its wide diversity in scenes (330 K images, 1.5 million object instances, 80 classes, 5 captions per image etc.)

We create a depth map of synthetic dataset

We apply pre trained MegaDepth (refer to supp Fig. 1 next page) on COCO to obtain pseudo ground truth depth maps

We randomly randomly sample and place these context-synthesis regions on different images in the COCO dataset.

We thus can obtain the ground truth content (RGB-D) from the simulated occluded region.

Context aware color and depth-inpainting

Training data generation

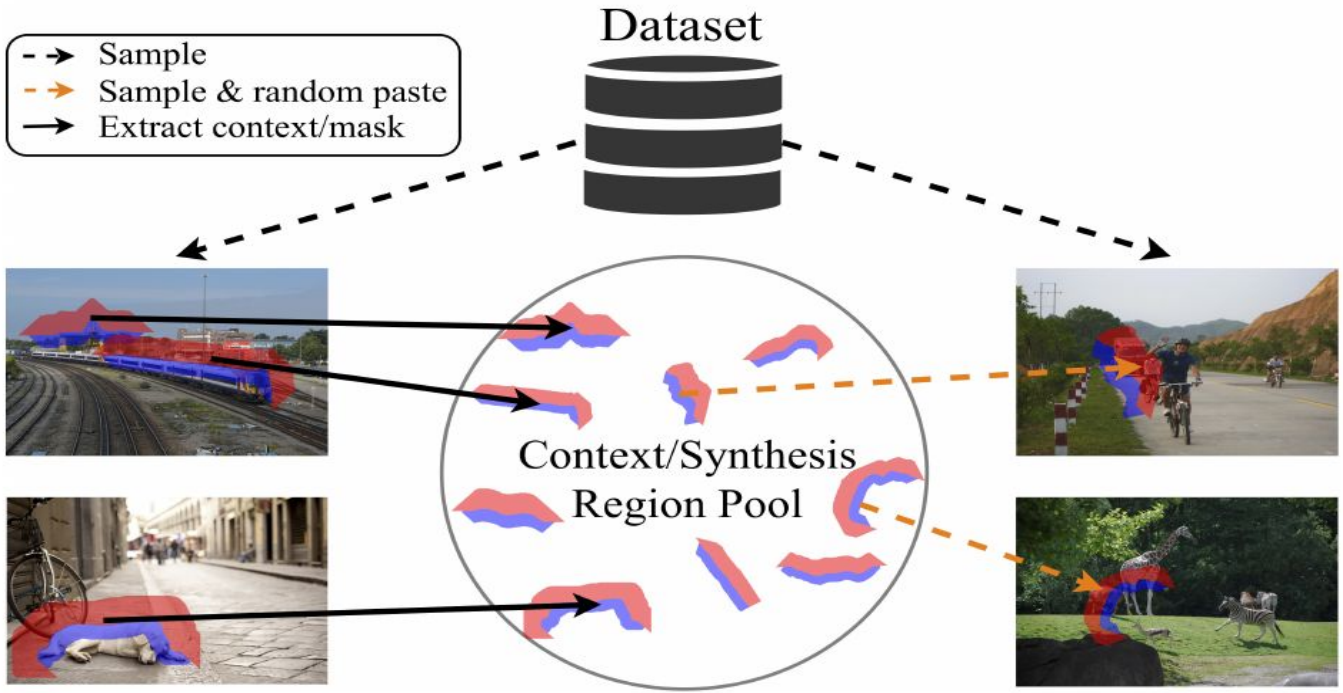


Figure 1. **Dataset generation process.** We first form a collection of context/synthesis regions by extracting them from the linked depth edges in images on the COCO dataset. We then randomly sample and paste these regions onto *different* images, forming our training dataset for context-aware color and depth inpainting.

Context aware color and depth-inpainting

Converting to 3D textured mesh

We form the 3D textured mesh by integrating all the inpainted depth and color values back into the original LDI.

Using mesh representations for rendering allows us to quickly render novel views, without the need to perform per-view inference step.

Consequently, the 3D representation produced by our algorithm can easily be rendered using standard graphics engines on edge devices.

Experimental Results

In this section...

We will look at these stuffs!

- Implementation Details
  - Training the model!
  - Inpainting model Arch.
  - Training data
- Visual Comparisons
  - Comparison with methods with MPI representations
  - Comparison with Facebook 3D photos
- Handling different depth maps

Implementation Details

Training the Inpainting model

For the edge generator, we follow the adam optimizer with $\beta=0.9$ and an initial learning rate of 0.0001

We train both the edge and depth generator model using context-synthesis regions on MS COCO for 5 epochs

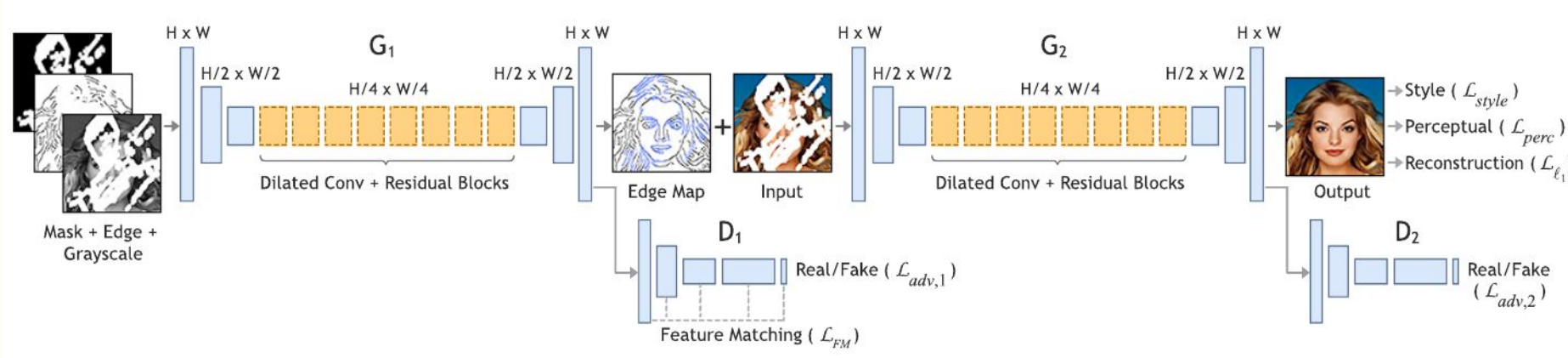
Inpainting model architecture

For edge inpainting network we follow the architecture of “Edgeconnect: Generative image inpainting with adversarial edge learning”

For depth and color inpainting network we use the standard U-Net architecture with partial convolution

Implementation Details

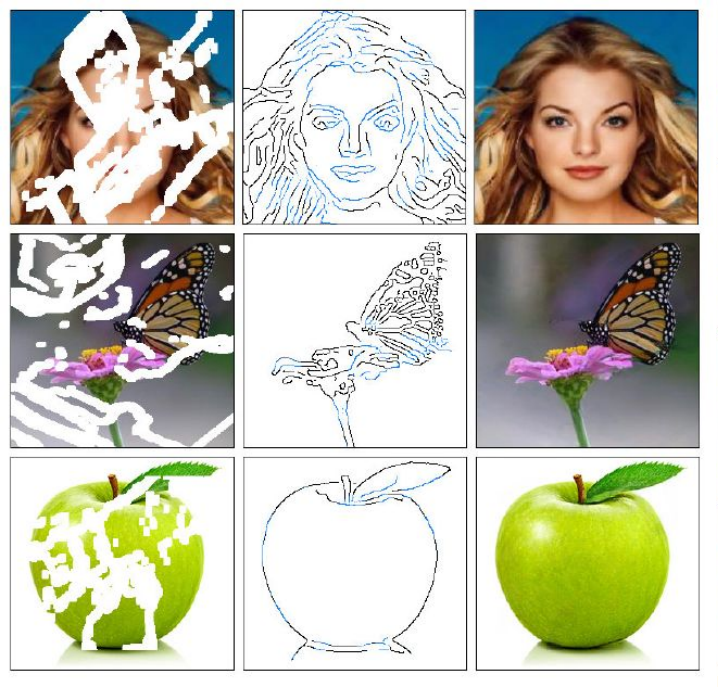
Edgeconnect is a GAN, where we give an incomplete image (with holes) and a corresponding edge map, and it recreates the original images



It can also be used to do other creative tasks such as object removal, object insertion, and object mixing

Implementation Details

Edgeconnect used (i) as it is, with edge values (ii) object removal and (iii) merging two images



Implementation Details

U-Net: Convolutional Networks for Biomedical Image Segmentation

The architecture is shown in the figure

We apply deconvolution to get feature maps for faster segmentation

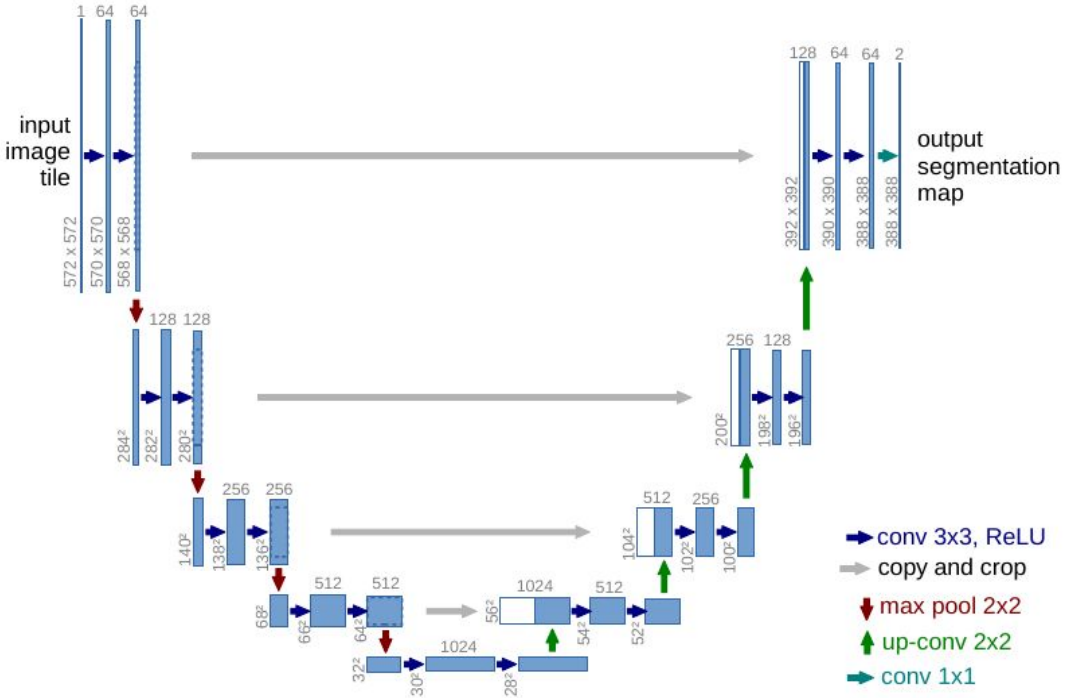


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Implementation Details

Architecture of the Edgeconnect network for edge inpainting network

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

Implementation Details

Architecture of the U-Net for depth and color inpainting network

Table 2. **Model architecture of our color and depth inpainting models.** W denote partial convolution layer as PConv, and denote BatchNorm as BN. We add the context and synthesis region together as the partial masks for the PConv layers.

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	-	-

Implementation Details

Training data

We use the 118k images from COCO 2017 set for training.

We select at most 3 pairs of regions from each image to form the context-synthesis pool.

During training, we sample one pair of regions for each image, and resize it by a factor between [1.0, 1.3].

Visual Comparisons

Comparison with method with MPI representations

*MPI (Message Passing Interface) for distributed computing, specially used for Deep Learning

We compare our proposed model against MPI-based approaches on RealEstate10K dataset.

RealEstate10K is a large dataset of camera poses corresponding to **10 million** frames derived from about **80,000** video clips, gathered from about **10,000** YouTube videos

We use DPSNet to obtain the input depth maps for our method.

We render the novel views of MPI-based methods using the pretrained weights provided by their authors.

Visual Comparisons

DPSNet: End-to-end Deep Plane Sweep Stereo

Outputs a depth map by taking a sequence of input images

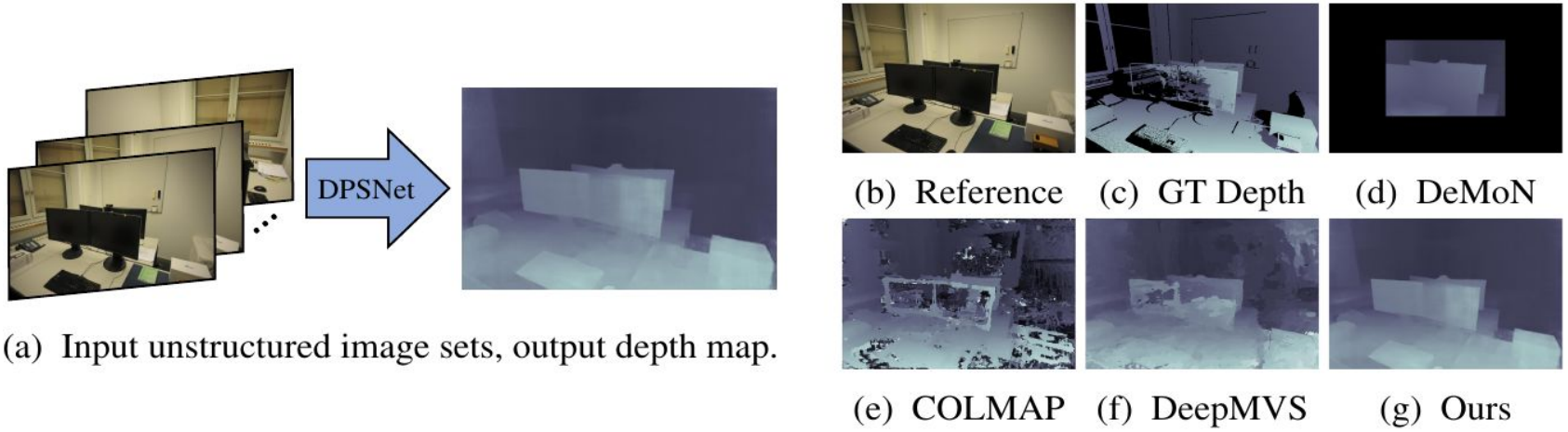
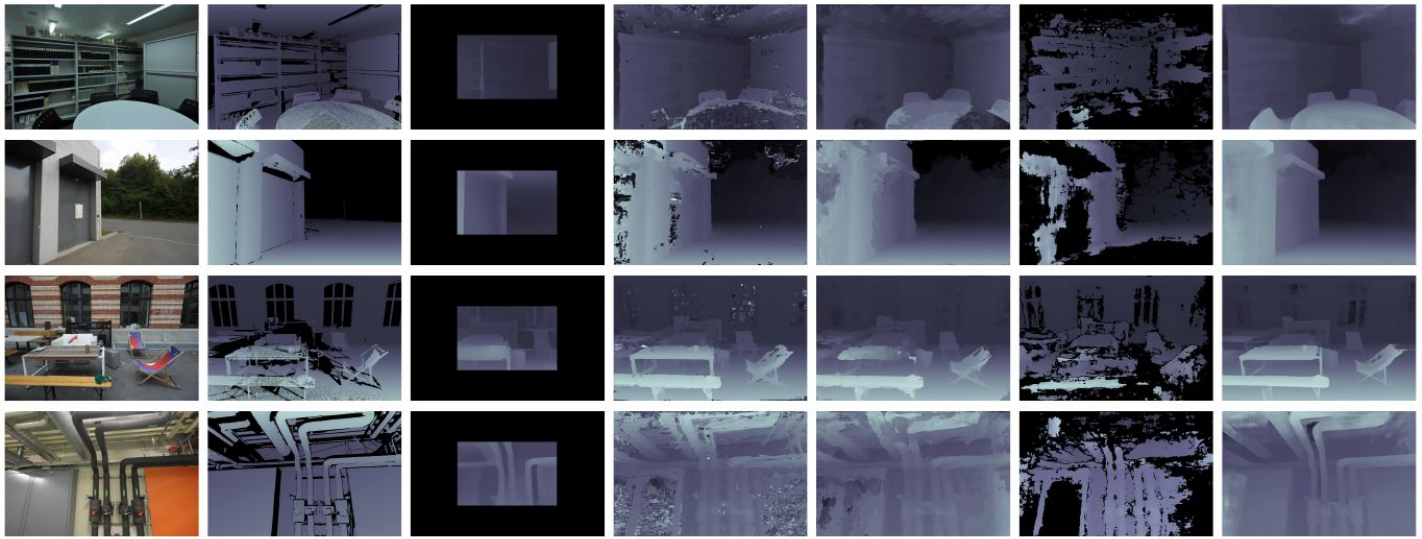


Figure 1: Results of DPSNet with comparisons to state-of-the-art methods.

Visual Comparisons

DPSNet: Comparison with state of the art related methods



(a) Images (b) GT depth (c) DeMoN (d) COLMAP (e) DeepMVS (f) MVSNet (g) Ours

Figure 5: Depth map results on four-view image sequences from the ETH3D dataset.

Visual Comparisons

DPSNet: Pipeline

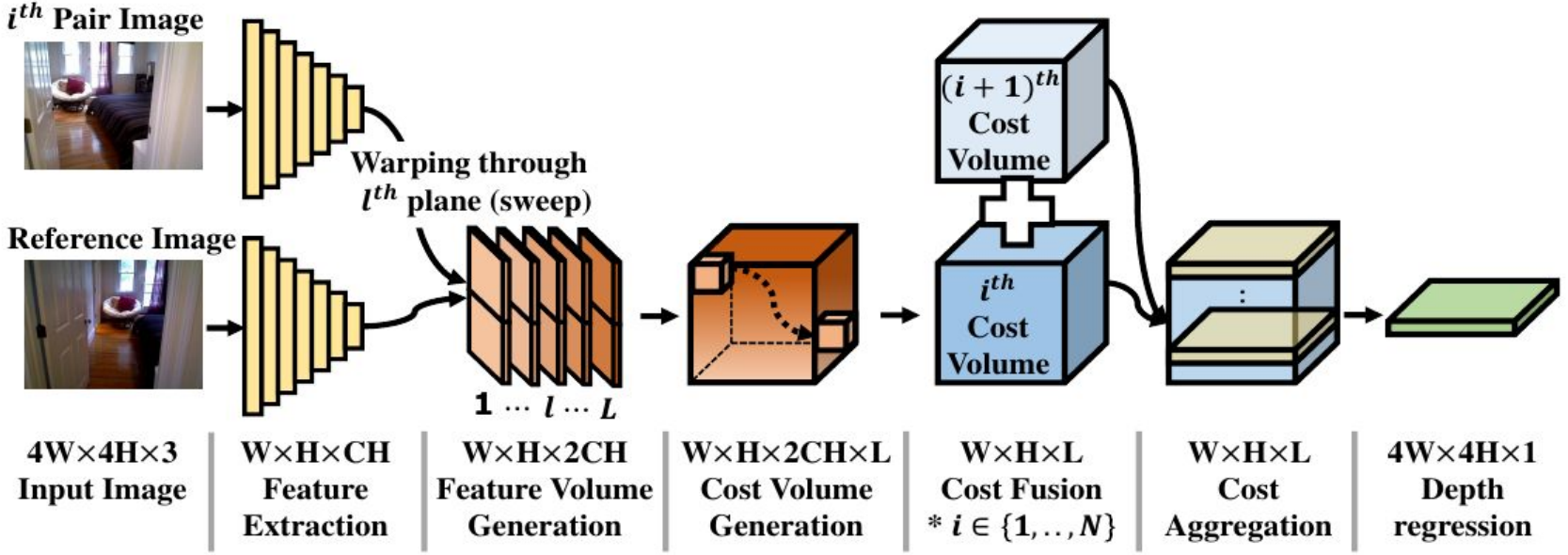


Figure 2: Overview of the DPSNet pipeline.

Visual Comparisons

Figure 9 shows two challenging examples with complex depth structures

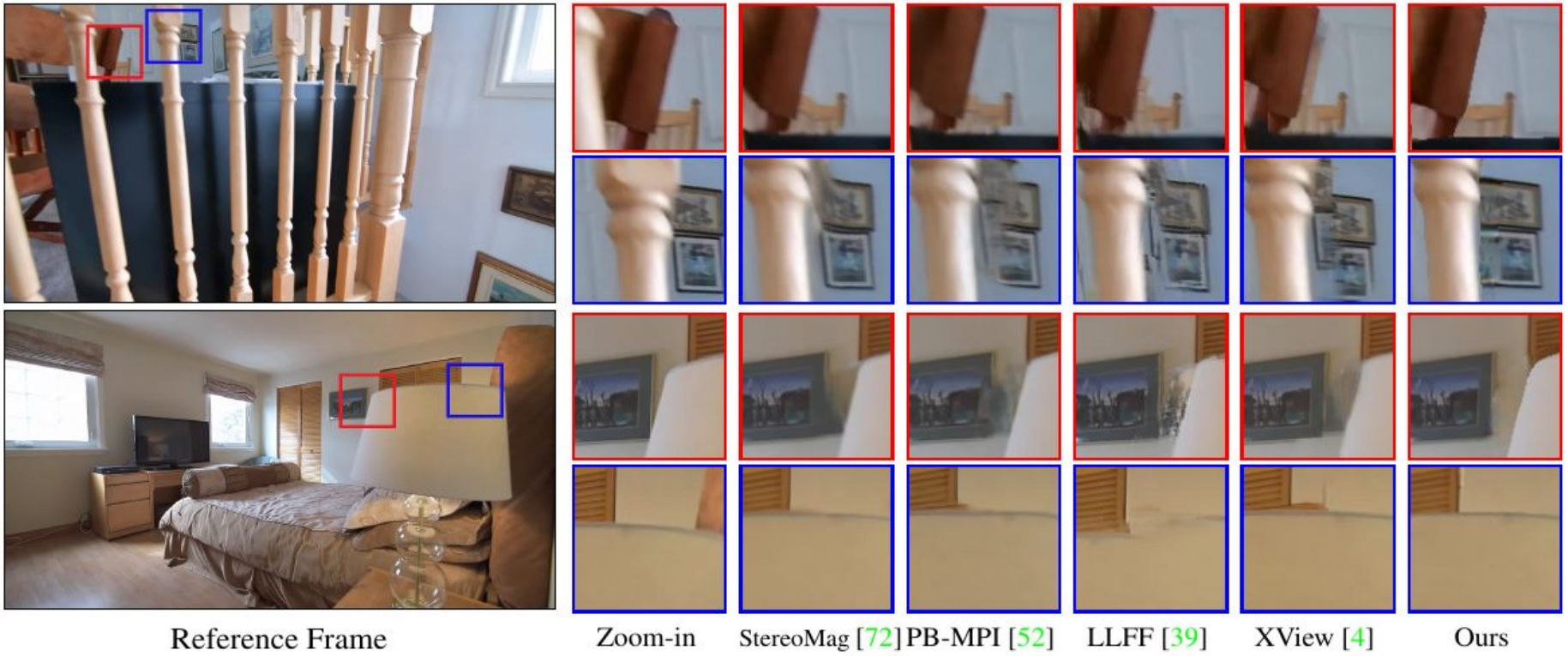


Figure 9. **Visual comparison with MPI-based methods.** Our method inpaints plausible structure and color in the occluded region.

Visual Comparisons

Comparison with method with MPI representations

Our method synthesizes plausible structures around depth boundaries; on the other hand, stereo magnification and PB-MPI produce artifacts around depth discontinuities.

LLFF suffers from ghosting effects when extrapolating new views.

Comparison with Facebook 3D photos

Here, we aim to evaluate the capability of our method on photos taken in the wild.

We extract the color images and the corresponding depth maps estimated from an iPhone X (with dual camera lens).

We use the same set of RGB-D inputs for both Facebook 3D photo and our algorithm. Figure 10 shows the view synthesis result in comparison with Facebook 3D photo.

Visual Comparisons

Comparison with Facebook 3D photos

The diffused color and depth values by the facebook 3D photo algorithm work well when small or thin occluded regions are revealed at novel views.

These artifacts, however, become clearly visible with larger occluded regions.

On the other hand, our results in general fills in the synthesis regions with visually plausible contents and structures.

Comparison with Facebook 3D photos

Not so good results in case of Facebook

Figure 10. **Visual comparison to Facebook 3D Photos.** Our approach fills plausible textures and structures at disocclusions.

Visual Comparisons

Handling different depth maps

We observe that our proposed model yields better perceptual quality (see Figure 11)

We test our method using depth maps generated using different approaches (Figure 12)

Figure 11. **Color inpainting leads to better visual quality.**

We select images from SUNRGBD dataset, and obtain the corresponding depth maps from three different sources:

1) depth estimated with MegaDepth

2) MiDas

Visual Comparisons

Handling different depth maps

and 3) Kinect depth sensor.

We present the resulting 3D photos in Figure 12.

Figure 12. **Our method works with various sources of depth map.** We show the depth estimates on the top-left of novel views.

The results show that our method can handle depth maps from different sources reasonably well.

Our Results on Custom Inputs...

On a picture of Silhouette Lake (Agartala)

The original picture that is passed to the algorithm, just captured from single camera...

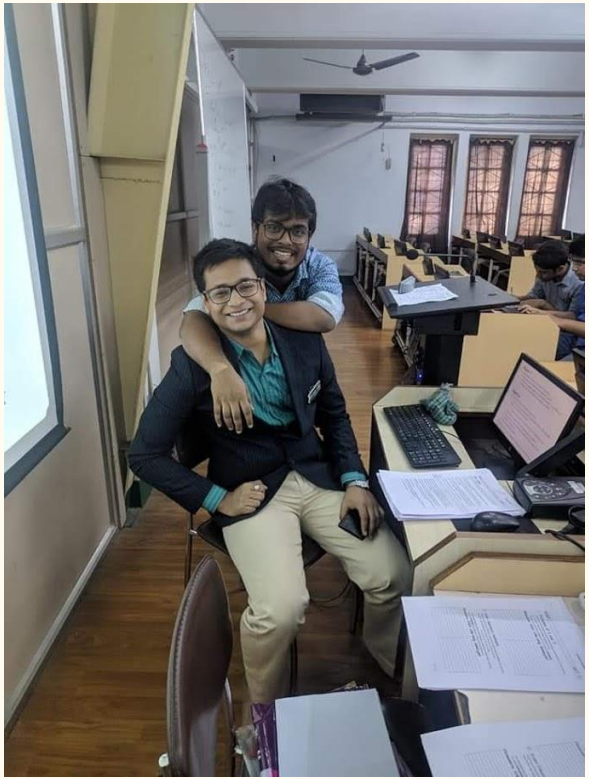
Zoom in

Circle

All in one = Swing

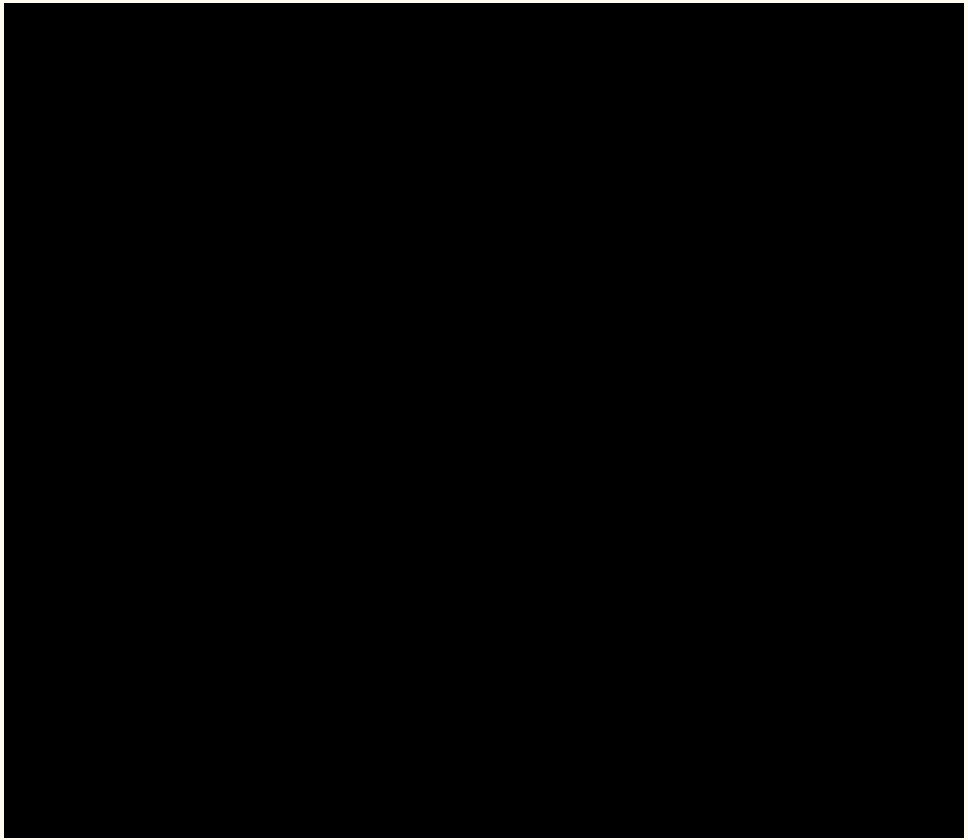
Old picture from St. Xavier's College, Kolkata

The original picture that is passed to the algorithm, just captured from single camera...

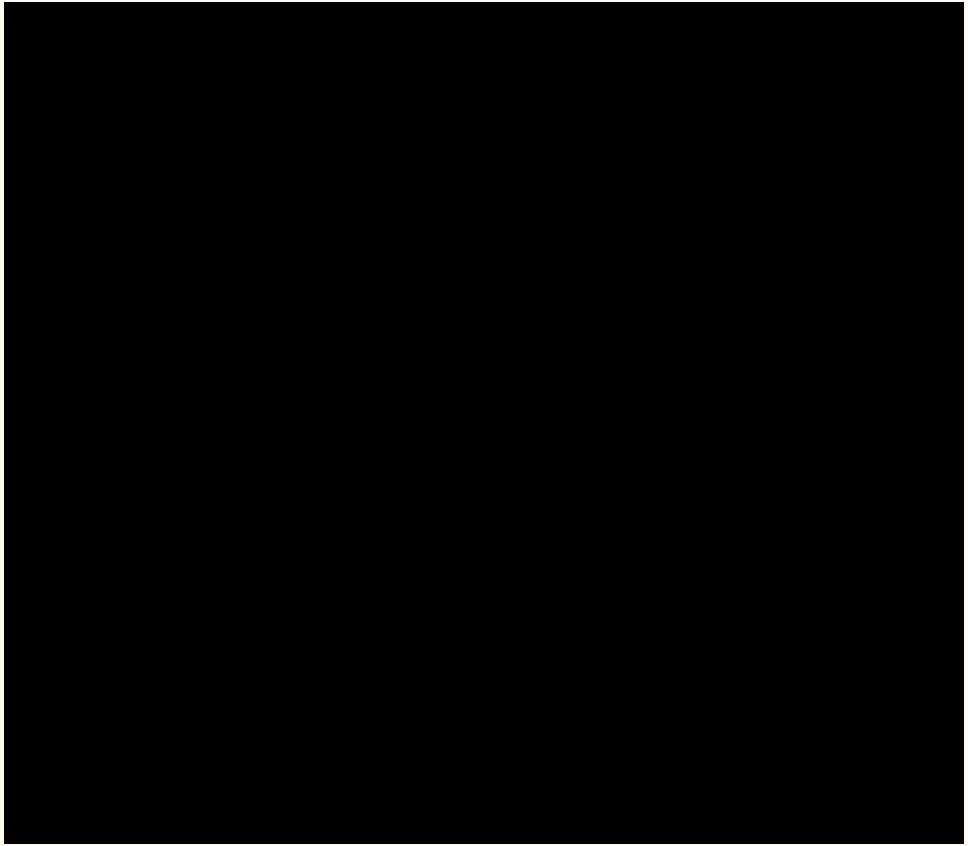


Zoom in

Zoom in dolly



Circle



Swing

Picture of a Village, Paikhala

The original picture that is passed to the algorithm, just captured from single camera...

Zoom in

Zoom in dolly

Circle

Swing

Picture of a picture

The original picture that is passed to the algorithm, just captured from single camera...

Zoom in

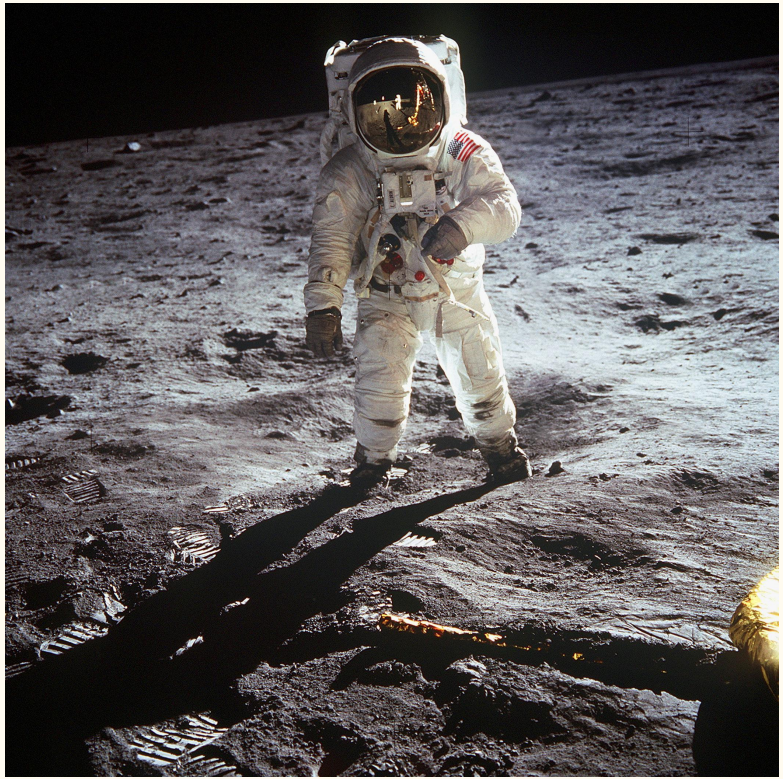
Zoom in dolly

Circle

Swing

Their results: Image of Edwin (Buzz) Aldrin

The original picture that is passed to the algorithm..

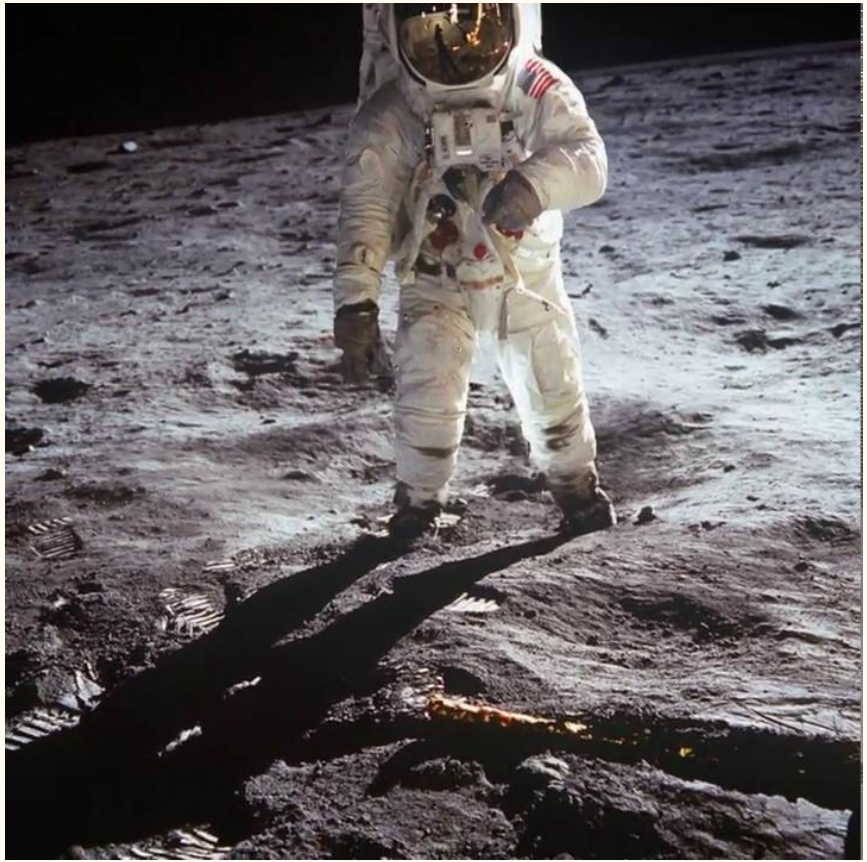


Disparity Map Created

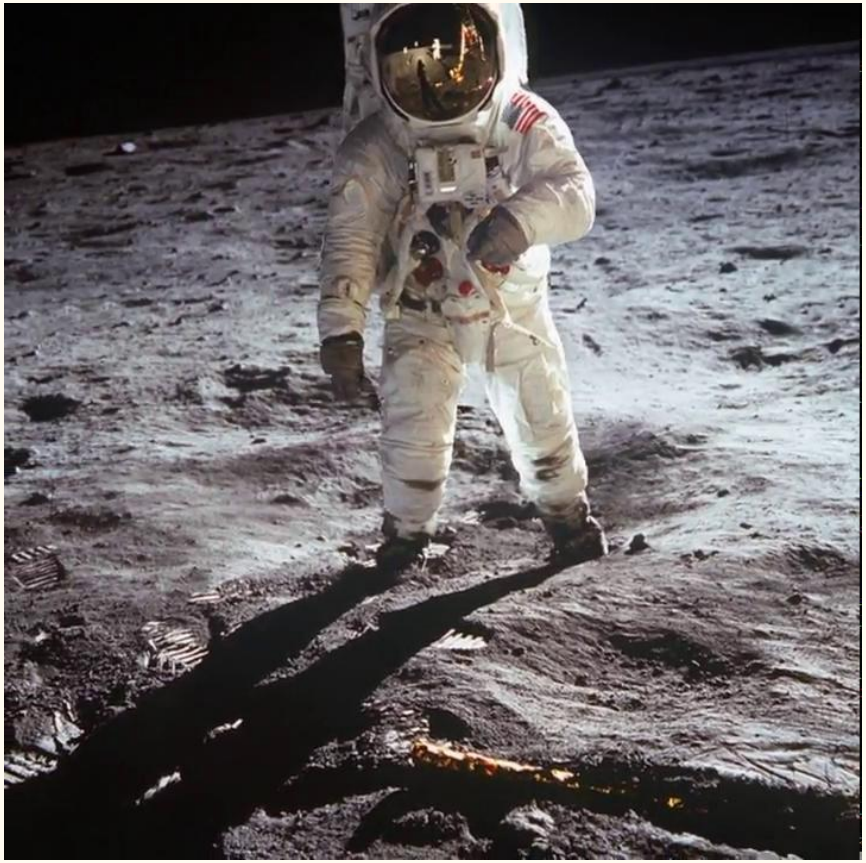
Zoom in

Zoom in dolly

Circle



Swing



Picture of a duck

The original picture that is passed to the algorithm..

Synsin

Ken-Burns

Ours

Picture of a boy

The original picture that is passed to the algorithm...

Result of FB's algorithm

Our Results

Scope for Improvements

Picture of a Stilt hut

The original picture that is passed to the algorithm, just captured from single camera...

Zoom in

Zoom in dolly

Circle

Swing

Picture of my Graduation in St. Xavier's College

The original picture that is passed to the algorithm, just captured from single camera...

Zoom in

Circle

Swing

St. Agnes school inside IIT KGP Campus, Circa 2004

The original picture that is passed to the algorithm, just captured from single camera...

Zoom in

Zoom in dolly

Circle

Swing

Conclusion

In this investigation, we present an algorithm for creating compelling 3D photography from a single RGB-D image.

Our core technical novelty lies in creating a completed layered depth image representation through context-aware color and depth inpainting.

We validate our method on a wide variety of everyday scenes.

Our experimental results show that our algorithm produces considerably fewer visual artifacts when compared with the state-of-the-art novel view synthesis techniques.

We believe that such technology can bring 3D photography to a broader community, allowing people to easily capture scenes for immersive viewing.

References

🔖 Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3d photography using context-aware layered depth inpainting, 2020.

🔖 Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, page 234–241, 2015.

🔖 Kamyar Nazeri, Eric Ng, Tony Joseph, Faisal Z. Qureshi, and Mehran Ebrahimi. Edgeconnect: Generative image inpainting with adversarial edge learning, 2019.

🔖 Sunghoon Im, Hae-Gon Jeon, Stephen Lin, and In So Kweon. Dpsnet: End-to-end deep plane sweep stereo, 2019.

🔖 J. B. Pal, and T. Maharaj. A Deeper Look into Hybrid Images, 2020.

Thank You

jimutbahanpal@{yahoo,google,outlook}.com