ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 3201
WSCG 2022 Proceedings

# Depth Completion for Close-Range Specular Objects

S. Pourmand
XLIM, UMR CNRS 7252
Université de Limoges
F-87000 Limoges, France
shahrzad.pourmand@unilim.fr

N. Merillou
XLIM, UMR CNRS 7252
Université de Limoges
F-87000 Limoges, France
nicolas.merillou@unilim.fr

S. Merillou
XLIM, UMR CNRS 7252
Université de Limoges
F-87000 Limoges, France
stephane.merillou@unilim.fr

## ABSTRACT

Many objects in the real world exhibit specular reflections. Due to the limitations of the basic RGB-D cameras, it is particularly challenging to accurately capture their 3D shapes. In this work, we present an approach to correct the depth of close-range specular objects using convolutional neural networks. We first generate a synthetic dataset containing such close-range objects. We then train a deep convolutional network to estimate normal and boundary maps from a single image.With these results, we propose an algorithm to detect the incorrect area of the raw depth map. After removing the erroneous zone, we complete the depth channel.

## Keywords

Depth completion, RGB-D images, synthetic dataset, specular reflections.

## 1. INTRODUCTION

With the availability of affordable RGB-D cameras, there have been a lot of advances in 3D computer vision to improve their capabilities for consumer use. This includes a variety of applications on 3D reconstruction, robot manipulation, virtual and augmented reality. However, these RGB-D cameras have some limitations; their depth estimation frequently suffers from missing or incorrect values, especially on shiny and transparent objects.

The Intel Realsense D435 [IR21] is a good and affordable RGB-D camera that produces a color image along with its corresponding depth map. It uses stereo vision technology with an infrared projector/sensor to accurately measure depth. However, we can observe some limitations by testing the camera, illustrated on figure 1:

- Missing data in the raw depth-map;

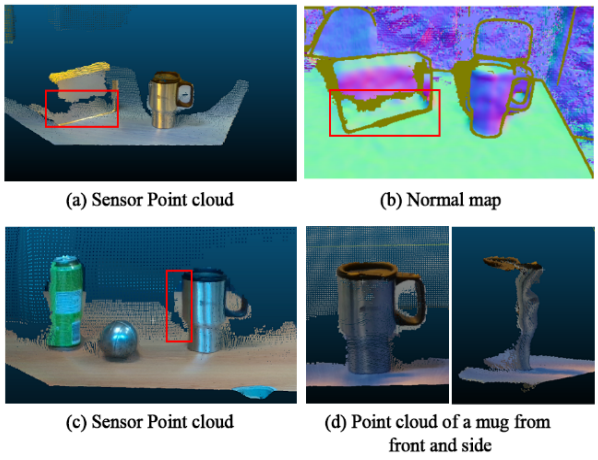


**Figure 1. Examples of Intel Realsense D435 camera limitations; (a) shows invalid values on a shiny box, (b) is the corresponding normal map. (c) shows the confusion of the depth near edges and (d) shows the**



- Pixels with invalid values on the shiny or transparent surfaces;
- Confusion of the depth near edges of the foreground object and background or two close foreground objects;
- Noisy estimation of shiny object's curvature.

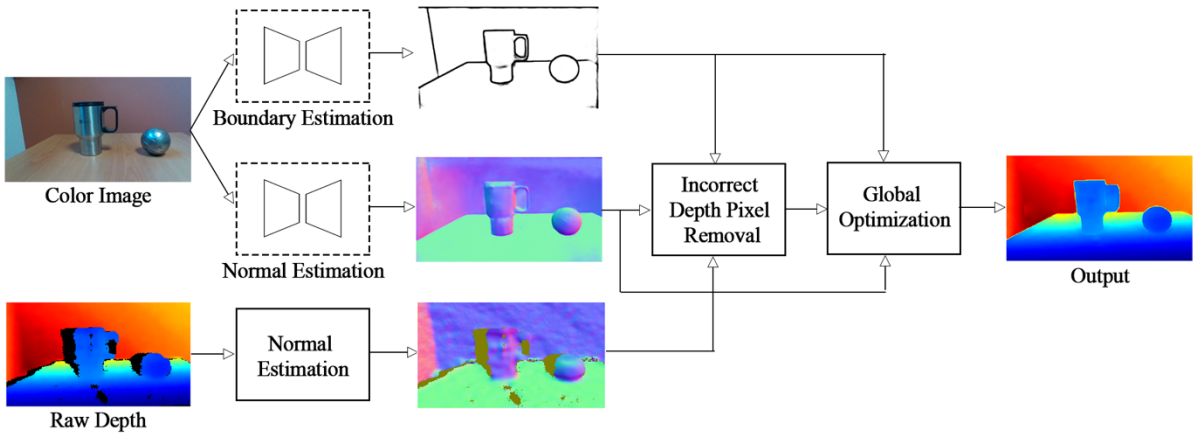


**Figure 2: Proposed pipeline: surface normals and boundaries are estimated from the color image. They are then used to remove the unreliable regions in the depth map. The depth is finally completed using a global optimization step.**

In this work, we propose a pipeline to refine captured depth values of glossy objects for close-range applications. Our approach is illustrated in figure 2. Given a single color image, we first use convolutional neural networks (CNN) to predict both the surface normals and objects boundaries. The CNN is trained with our own synthetic datasets (section 3.1). At this point, we can remove incorrect depth values in three steps. First, we remove the boundary of the objects using an estimated boundary mask. Second, we compute the normals directly from the sensor raw depth map and compare them to the normals estimated from the color image in order to remove areas where they differ significantly. Third, we use morphological transformations to remove the persisting noise. Eventually, we use a global optimization approach to fill the holes in the depth map.

## 2. RELATED WORK

### 2.1. Depth completion algorithm

Since the appearance of low cost RGB-D cameras, many methods have been proposed to overcome their limitations, which appear as holes and incorrect areas in the depth map. Most of these methods utilize a color image as guidance to correct the depth map.

Huang et al [HHC14] use the edges of the color image to detect the unreliable zone in depth. Then they removed the existing depth in the unreliable zone to prevent erroneous depth propagation. Next, they used the fast marching method (FMM) [T04] which was first introduced for image inpainting and then revised for depth inpainting [LGL12] to complete the depth.

In recent years, with the latest advances in deep learning, a lot of data-driven approaches have emerged. Zhang and Funkhouser [ZF18] completed the depth channel of an RGB-D image using global optimization. They first predict the normals from the color image and then solve for completed depth. Their work is mainly focused on large indoor environments with large holes. However, the lack of depth denoising prevents their method to obtain correct results in depth completion.

Shreeyak et al. [SMPN20] proposed a modification to the previous method to estimate the depth of close-range transparent objects for manipulation. Their method consists of detecting transparent objects in the color image, removing their depth values, and using global optimization to complete the depth of all missing areas. However, their method is not suitable in our case because the depth captured by depth cameras is not as inaccurate for specular objects as it is for transparent objects. As a result, removing all the depth values of the objects would be excessive.

Xian et al [XQLL20] have developped a method to eliminate incorrect depth based on a segmentation network trained with labelled RGB-D images of large indoor scenes.

### 2.2. 3D understanding from the single color image

It has been shown that a single color image can be used to directly infer depth [EPF14, LKY17], or other geometric features [SR12, EF15, WFG15]. One of the geometric features that is tightly coupled with depth is normal-map. Normal-map is a pixel-map that contains the orientation of the surface at each point. Wang et al. [WFG15] proposed a method to predict surface normals at local and global scales, then they used a fusion network to combine both predictions into a final result. Eigen and Fergus [EF15] designed a single multiscale convolutional network to predict depth, surface normals, and semantic labels.

Recently, Yinda Zhang et al. [ZS17] utilized a U-net architecture with VGG-16 as backbone for normal estimation and achieved state-of-the-art results.

In this work, we use U-Net architecture with inceptionv4 [SI16] as our model.

### 2.3. Dataset for training neural network

There are several real-world datasets available for training neural networks including NYU depth v2 [SHKF12], ScanNet [DCSH17], and Matterport3D [CDFH17] which are all highly used datasets.

However, real-world datasets generally suffer from noise and missing data in shiny, transparent, or distant areas as they are limited by the quality of the capturing device.

To overcome these limitations, researchers have been increasingly using synthetic data for training networks for vision and robotic tasks [KMHV17, SMPN20]. The advantage of using synthetic data for training is that it is easy to obtain pixel-perfect ground truth data. Furthermore, it is affordable to create large-scale datasets. In this work, we will generate our own dataset containing a mix of close-range shiny (with specular reflections) and rough objects (with diffuse reflections) to meet our needs.

## 3. PROPOSED METHOD

### 3.1. Generating data

There are several RGB-D datasets available to train neural networks consisting of synthetic or real-world data. Yet, none of these are especially focused on close-range glossy objects. Therefore, we choose to generate our own customized dataset. We have created a scene generator in Blender. The generated scene would consist of a plane with different objects on it. The ground (plane) is randomly textured from 82 textures [URL1]. The camera is randomly placed in the scene, always focused on the ground. The environment lighting is randomly chosen from 120 indoor HDRI environments [URL2]. Next, some objects are chosen from primitive shapes presented in figure 3. These shapes represent a large variety of objects: smooth sphere, mesh with smoothed or hard edges, objects with or without holes. 3D objects are added to the scene as in Shreeyak et al. [SMPN20], by using Blender physics simulation to drop them on the ground: objects will collide with each other and the plane insuring a random final positioning. The textures of the objects are chosen from 47 textures [URL1] (including Blender's checker pattern) or a solid color. Their roughness and metallic values are also chosen randomly.

Each of the created scenes is rendered with Blender Cycles at the resolution of 256x256 pixels. Corresponding normal map and boundary map are also generated for each image. Before generating the normal map, a bounding-sphere is created in the scene. This sphere contains the plane and all the objects to ensure that if the HDRI background is visible in the rendered scene, the normals of those areas do not stay empty. This does not create an issue when training our network because the background information is irrelevant in our context. The generated normals are then oriented from world space to camera space and normalized in the range of -1 and 1. The resulting normal map is saved in OpenEXR format. This method permits us to create thousands of images for the next step (illustrated in figure 4).

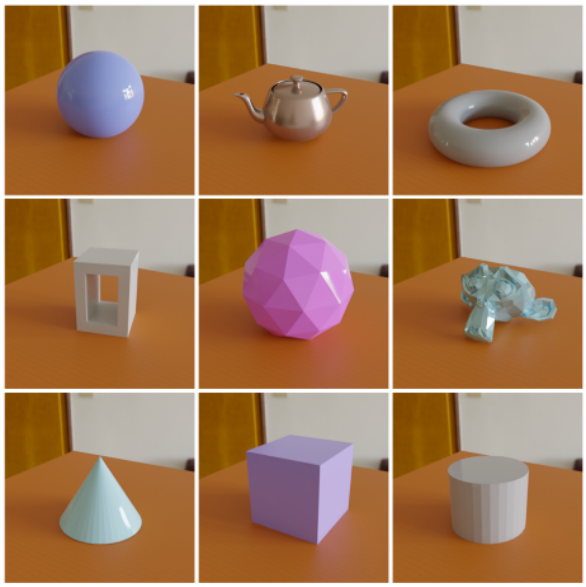

**Figure 3: Blender primitive shapes used for dataset generation**

### 3.2. Network Architecture and Training

We experimented with U-Net [RFB15] and DeepLabV3 [CP17] as our encoder-decoder model architecture. We trained DeeplabV3 with resnet101 [HZ16] and Unet with resnet101, inceptionv4 [SI16] and vgg16 [SZ14] as backbones [PY20]. All models have been pre-trained on ImageNet [RDSK15].

Our experiments led us to choose U-Net with inception4 as our model architecture for both normal estimation and boundary detection, as it generates better results with our dataset. U-net is (a U-shape) encoder-decoder architecture that uses skip-connections between corresponding layers of encoder and decoder. We train both networks on the data that we generated using Blender.

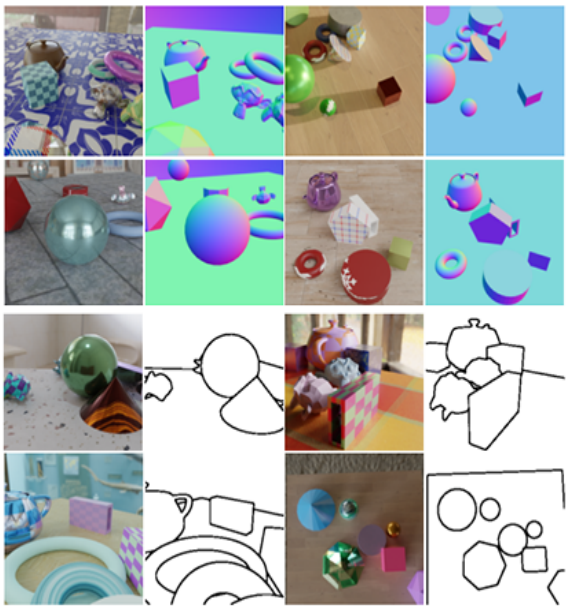

**Figure 4: Some examples of our synthetic training datasets. First and second rows illustrate sample data used for normal estimation training, while the two bottom rows illustrate data used for boundary detection training**

**Normal estimation:** We modify the last layer of the network to generate a three-channel output. Therefore, we will have the three components of the normal vector per pixel: x, y, and z. We then normalize the output to L2-norm. To calculate the loss, we use an element-wise dot product between the network's output and the generated synthetic ground-truth.

We train the network in two steps; first, we train the network on 10,000 synthetic images, which are then augmented using Gaussian blur. In the second step, we reduce the learning rate and re-train the network on 5000 synthetic images of very close-range objects. We have again augmented the data using Gaussian blur. We empirically found that re-training the network on very close-range objects with a reduced learning rate gives better results than training the network on all data in a single step.

**Boundary estimation:** For this task, we modify the last layer of the network to generate a single channel output. The activation function of the last layer is then set to sigmoid to output the value of each pixel between 0 and 1. Note that the boundary pixels of the ground-truth data are zero, while the non-boundary pixels are one. We dilate the boundary pixels before training to be able to use the result of the network directly for boundary removal, as described in section 3.3.

We use the binary cross-entropy loss as the loss function. Since the number of non-boundary labels is significantly higher than the number of boundary pixels, we use a loss weight that is ten times greater for boundary pixels than for non-boundary ones, as suggested by Yang et al [YPCLY16].

We train the network on 5000 synthetic images and we augment them using Gaussian blur and random changes of brightness, contrast, saturation, and hue.

### 3.3. Incorrect Depth Pixel Removal

RGB-D cameras generally do not measure edge depth correctly, causing confusion between the depths of the foreground object and the background or the depths of two close foreground objects. Thus, we begin by removing the depth of the object's boundaries. First we create a binary mask from the estimated boundary map, and then we apply this mask to the depth map to remove the border of objects.

Next, we compute the normal map of the depth map by considering adjacent 3D points [ZPK18]. We compare each of these normals to its corresponding one in the normal map estimated from the color image using a simple dot product. We remove the areas where the difference between the angles of the normals is greater than 30 degrees: we experimentally found that the normal estimation network gives nearly perfect results from this value, thus the normal map generated from color image would be reliable, see section 4. This step is performed to remove erroneous zones that appear on our objects as a result of specular reflections.

At last, we remove any small noise persisting in the two previous steps. This is done by generating a mask from the depth map by setting every zero value to zero (which are the holes in the depth map) and every non-zero value to one. We then use a morphological opening on the mask with a 5x5 circular structuring element as our kernel. We apply the mask to the depth map to remove the noise.

### 3.4. Global optimization

After removing the unreliable depth from the depth map, we use the optimization algorithm proposed by Zhang and Funkhouser [ZF18] to complete the depth. This approach takes the estimated normal map and estimated boundary map from the color image as input and uses them as a guide to complete the depth map. The optimization algorithm has the objective function as follows:

$$E = \lambda_D E_D + \lambda_S E_S + \lambda_N E_N B\,,$$

$$E_D = \sum_{p \in T_{obs}} \| D(p) - D_0(p) \|^2,$$

$$E_N = \sum_{p,q \in N} \| < v(p,q), N(p) > \|^2,$$

$$E_S = \sum_{p,q \in N} \| D(p) - D(q) \|^2,$$

where $E_D$ is the distance between the estimated depth and the raw depth, $E_S$ ensures that the neighbor pixels have similar depth values and $E_N$ uses the dot product to ensure that the normal estimated from the image and the estimated depth are consistent. $B$ has a value between [0,1] and down-weights the $E_N$ based on the predicted probability that a pixel is on the boundary.

## 4. EXPERIMENTAL RESULTS

We evaluate the normal estimation network on a synthetic test set; the test set consists of 100 synthetic unseen images that we generated using our data generation method, but composed of new (unseen) objects. For all the images, we compute the mean and the median of the angular error measured between the estimated normals and ground truth ones. We also compute the percentage of the pixels with errors smaller than 30, 22.5, and 11.5 degrees (common metric also used for example in [SMPN20]). The results are presented in Table 1 and show a perfect matching above 30°.

| | Mean | Median | 11.25° | 22.5° | 30° |
|---|---|---|---|---|---|
| Our model | 24,2 | 18,3 | 38,7 | 54,3 | 99,8 |

**Table 1. Evaluation metrics on the synthetic test set for normal estimation network.**

We have tested our approach on real data captured by the Intel Realsense D435 camera. The testing environment consists of a table with a few objects on it. We use a variety of specular objects with different shapes and materials; Some of them have similar shapes to the synthetic objects while others have previously unseen shapes. The images and depth maps are taken under ambient light and at the resolution of 424x240 pixels.

We train and test our proposed pipeline on a desktop computer equipped with an Intel Core(TM) i9 2.80GHz CPU with 16 GB RAM and NVIDIA GeForce RTX 2080 GPU. For each test of RGB-D data with a resolution of 424x240 pixels, estimating normals and boundaries of the color image and removing unreliable region from the depth map takes about 0.3 seconds while optimization on depth map takes about 1.8 seconds. We compare our proposed method with the method proposed by Zhang and Funkhouser [ZF18]. We have used the code they provide on their github page without making any modification. For the optimization used in both Zhang and Funkhouser's and ours, we use the following values: $\lambda_D = 1000, \lambda_S = 0.001$ and $\lambda_N = 1$.

As illustrated in figure 5, we can see that our proposed pipeline greatly improves the depth completion results in close-range tasks. Note that the object boundaries got sharper and depth inside the objects is more consistent, especially for the box in rows three and four.

## 5. CONCLUSION

In this paper, we propose a depth correction and completion pipeline for close-range shiny objects. Our contribution is twofold; first, we prepare a synthetic dataset consisting of close-range specular and non-specular objects. This permits to train a CNN with pixel-perfect values of normals and boundaries. Second, we propose an algorithm to detect and remove unreliable regions in depth map, based on the normals and boundaries predicted from a single color image. Eventually, we complete the depth using the optimization method proposed by Zhang and Funkhouser [ZF18]. A theoretically possible drawback could arise if the depth of a specular object is incorrect, while its normal map is accurate. We will investigate this as future work. We will also estimate the influence of noise differential (synthetic images vs real captures) in our kind of pipeline. Despite theses possible drawbacks, in most cases, our method produces excellent results for close-range objects.

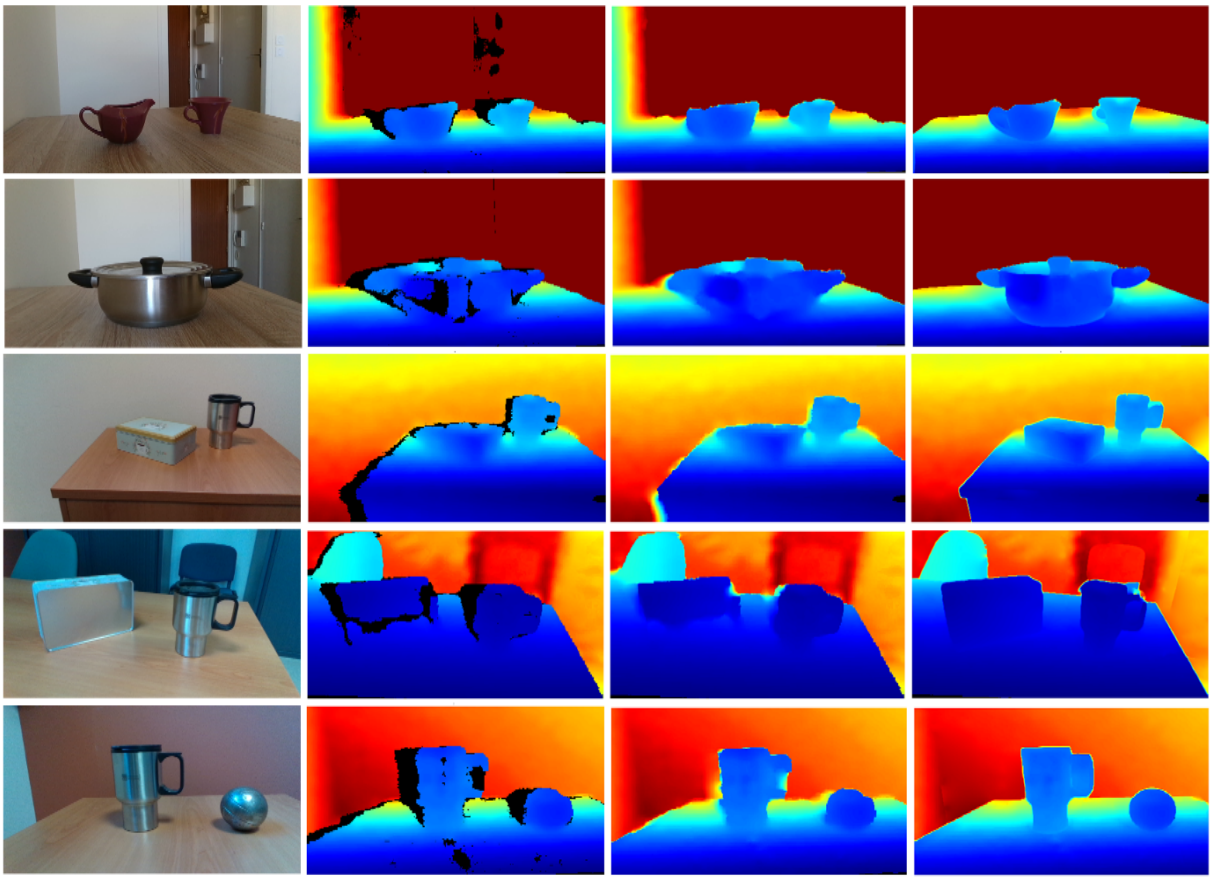


**Figure 5: Although we used the same depth completion approach as Zhang and Funkhouser [ZF18], this comparison demonstrates that our proposed pipeline considerably improves the depth completion results for close-range specular objects. (In the first two rows, the depth beyond one meter is clipped for better visualization)**

## REFERENCES

[CDFH17] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng and Yinda Zhang. Matterport3D: Learning from RGB-D Data in Indoor Environments. In International Conference on 3D Vision (3DV), 2017

[CP17] Liang-Chieh Chen, George Papandreou, Florian Schroff, Hartwig Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. 2017

[DCSH17] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser and Matthias Nießner. ScanNet: Richly-annotated 3d reconstructions of indoor scenes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017

[EF15] David Eigen, Rob Fergus. Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture. ICCV, 2015.

[EPF14] David Eigen, Christian Puhrsch and Rob Fergus. Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. CoRR, 2014.

[HHC14] Yung-Lin Huang, Tang-Wei Hsu, Shao-Yi Chien. Edge-aware depth completion for point-cloud 3D scene visualization on an RGB-D camera. In IEEE Visual Communications and Image Processing Conference, VCIP 2014

[HZ16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE

conference on computer vision and pattern recognition, 2016

[IR21] Intel RealSense D400 series Product Family Datasheet. URL: https://dev.intelrealsense.com /docs/intel-realsense-d400-series-product-family-datasheet

[KMHV17] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An Interactive 3D Environment for Visual Ai, 2017.

[LGL12] Junyi Liu, Xiaojin Gong and Jilin Liu. Guided inpainting and filtering for Kinect depth maps. In Proceedings of the 21st International Conference on Pattern Recognition (ICPR), 2012

[LKY17] Jun Li, Reinhard Klein and Angela Yao. A Two-Streamed Network for Estimating Fine-Scaled Depth Maps from Single RGB Images. In Proceedings of the IEEE International Conference on Computer vision, 2017.

[PY20] Pavel Yakubovskiy. Segmentation Models Pytorch, 2020, URL: https://github.com /qubvel/segmentation_models.pytorch

[RDSK15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Anderj Karpathy, Adity Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. IJCV, 2015.

[RFB15] Olaf Ronneberger, Philipp Fischer and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In International Conference on Medical image computing and computer-assisted intervention, pages 234-241. Springer, 2015.

[SHKF12] Nathan Silberman, Derek Hoiem, Pushmeet Kohli and Rob Fergus. Indoor segmentation and support inference from rgbd images. In European Conference on Computer Vision, pages 746-760. 2012

[SI16] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In Proc. AAAI Conference on Artificial Intelligence, 2016.

[SMPN20] Shreeyak S. Sajjan, Matthew Moore, Mike Pan, Ganesh Nagaraja, Johnny Lee, Andy Zeng, Shuran Song. ClearGrasp: 3D Shape Estimation of Transparent Objects for Manipulation - IEEE International Conference on Robotics and Automation (ICRA), 2020

[SR12] Alexander G. Schwing and Raquel Urtasun. Efficient Exact Inference for 3D Indoor Scene Understanding. In ECCV, 2012.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014

[T04] Alexandru Telea. An image inpainting technique based on the fast marching method. In Journal of Graphics Tools, vol.9, 2004

[URL1] https://resources.blogscopia.com/category /textures/, and https://3dtextures.me/

[URL2] https://polyhaven.com/hdris/

[WFG15] Xiaolong Wang, David F. Fouhey and Abhinav Gupta. Designing Deep Networks for Surface Normal Estimation. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[XQLL20] Chuhua Xian, Kun Qian, Guoliang Luo, Guiqing Li and Jianming Lv. Elimination of incorrect Depth Points for Depth Completion. In proceedings of CGI 2020, pp. 245-255

[YPCLY16] Jimei Yang, Brian Price, Scott Cohen, Honglak Lee and Ming-Hsuan Yang. Object Contour Detection with a Fully Convolutional Encoder-Decoder Network. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 193-202

[ZF18] Yinda Zhang and Thomas Funkhouser. Deep depth completion of a single RGB-D Image – The IEEE Conference on computer vision and pattern recognition, 2018

[ZPK18] Qian-Yi Zhou, Jaesik Park and Vladlen Koltun. Open3D: A Modern Library for 3D Data Processing, 2018.

[ZS17] Yinda Zhang, Shuran Song, Ersin Yumer, Manolis Savva, Joon-Young Lee, Hailin Jin, and Thomas Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.