

Learning Depth Completion of Transparent Objects using Augmented Unpaired Data

Floris Erich, Bruno Leme, Noriaki Ando, Ryo Hanai, Yukiyasu Domae

Abstract—We propose a technique for depth completion of transparent objects using augmented data captured directly from real environments with complicated geometry. Using cyclic adversarial learning we train translators to convert between painted versions of the objects and their real transparent counterpart. The translators are trained on unpaired data, hence datasets can be created rapidly and without any manual labeling. Our technique does not make any assumptions about the geometry of the environment, unlike SOTA systems that assume easily observable occlusion and contact edges, such as ClearGrasp. We show how our technique outperforms ClearGrasp in a dishwasher environment, in which occlusion and contact edges are difficult to observe. We also show how the technique can be used to create an object manipulation application with a humanoid robot. Supplementary URI: https://florise.github.io/faking_depth_web/.

I. INTRODUCTION

The manipulation of transparent objects in complex environments has been a topic of interest for the past decades, due to the limited ability of commodity RGBD sensors to estimate the depth of objects that are non-Lambertian. Saxena et al. showed translucent object manipulation in a dishwasher environment using a neural network in 2008 [1]. Lysenkov et al. presented a system for manipulating transparent objects by matching noise in depth maps to pre-measured CAD models of the objects in 2012 [2, 3]. Various works have tried to use synthetic data captured from simulation and rendering, one of the most prominent being ClearGrasp, which was introduced by Sajjan et al. [4]. Other works have used the ClearGrasp dataset while improving the depth estimation method [5]. These methods have as limitation that they require manual modeling of the transparent objects that are to be manipulated and/or the environment in which the manipulation will take place, or require models of highly similar objects and environments.

There are also various works that focus on *grasp point generation* for transparent objects instead of depth completion [6, 7, 8], but *depth completion* of transparent objects has as benefit that it allows the application of existing point-cloud based techniques for downstream tasks. Some works use specific sensors that are not typically found in robot systems, such as Light Field Cameras [8], polarization cameras [9] or background projections [10], but in this work we use a low cost RGBD sensor (Intel RealSense D405) that is commonly found in commercial robots. Other works use

markers and object models to create datasets of matching transparent RGB and Depth [11, 12], however in this case the markers remain part of the data or additional effort is required to remove the markers from the dataset, whereas in this work we have separate datasets for transparent objects and for augmented objects. Yet other works focus mostly on transparent surfaces [13] instead of objects, or transparent object segmentation instead of depth estimation [14]. In recent years Neural Radiance Fields (NeRF) [15] have enabled transparent object depth estimation from RGB data and manipulation without any prior training [16, 17, 18, 19], but at inference time techniques that use NeRF require multiple views of a scene and known camera extrinsics per view, whereas in this work we only require a single view at inference time.

Our main contribution is a technique for transparent object depth completion based on adversarial learning with cycle constraints (CycleGAN) which is trained using a set of views of transparent objects and a set of views of opaque clones. The opaque clones are the transparent objects augmented by painting them, so that they still have a similar shape as the objects that are to be manipulated, but do not exhibit the aspects that negatively affect depth detection. Our method learns a function that can convert sensor measurements (Depth or RGBD) of the augmented objects to and from sensor measurements of the original transparent objects. CycleGAN-based techniques have been applied in robotics before, in works such as RL-CycleGAN [20] and RetinaGAN [21]. In both cases the technique was used for sim2real transfer, whereas to our knowledge we are the first to apply the CycleGAN technique for “real2real” transfer in robotics. Adversarial learning has been applied to transparent object depth completion before, but only in a supervised setting [22]. Figure 1 shows an overview of our method. All training data is created by recording data from the real environment using the RGBD camera that is also used in the manipulation application. Unlike previous approaches the training data does not have to be paired with labeled ground truth data. Training data is split into a set of samples of transparent objects and a set of samples of the opaque clones. Transparent object samples contain color information of the objects that we want to manipulate, while the opaque clone samples contain the depth data that we want to associate with the transparent object samples. Depth data for transparent object samples is noisy and practically unusable as sole input for robot manipulation, while the color data for opaque clone samples is only useful at training time. Both datasets should be comparable in terms of environment and the

Floris Erich, Noriaki Ando, Ryo Hanai and Yukiyasu Domae are with the Industrial CPS Research Center, National Institute of Advanced Industrial Science and Technology, Japan. Bruno Leme is with the Horticultural Sciences Department, University of Florida, USA. Floris Erich is the corresponding author and can be reached at [\[first name\].\[last name\]@aist.go.jp](mailto:[first name].[last name]@aist.go.jp).

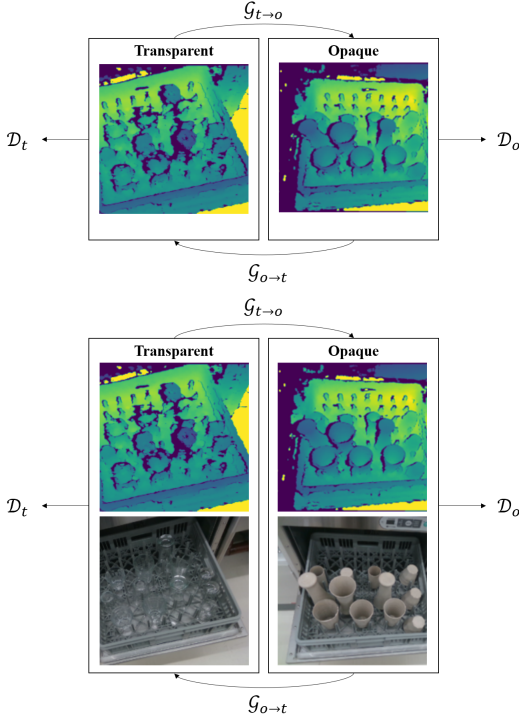


Fig. 1: Two cyclic adversarial training conditions were tested, direct depth to depth and RGBD to RGBD. For each both a U-Net inspired model and a ResNet inspired model were used for the generator architecture. Because we have used cyclic adversarial training, it is not required for samples to be paired.

types of objects used, but our technique does not require matching sensor transformation or object transformations between samples.

We have publicly released the datasets and neural network source code. Both can be found in the supplementary materials accessible via the URI in the abstract. We hope that other researchers can use the dataset and/or neural network source code to create new methods for unpaired training of transparent object depth completion methods.

II. METHOD

A. Adversarial loss

A Generative Adversarial Network (GAN) is a combination of a generator $\mathcal{G} : X \rightarrow Y$ and discriminator $\mathcal{D} : Y \rightarrow \{Real, Fake\}$, in which the generator tries to learn how to generate realistic samples, while the discriminator tries to learn how to distinguish between real samples $y \sim Y$ and generated samples $\mathcal{G}(x)$ [23]. In an ordinary GAN, x is a vector sampled from a random distribution X . The discriminator is trained to minimize its error on correctly identifying real samples as real while maximizing its error on incorrectly identifying generated samples as real. The objective function can be described as a minimax game:

$$g^* = \arg \min_{\mathcal{G}} \max_{\mathcal{D}} \mathbb{E}_{y \sim Y} \log \mathcal{D}(y) + \mathbb{E}_{x \sim X} \log(1 - \mathcal{D}(\mathcal{G}(x)))$$

In practice, we use the LSGAN loss functions [24] with $a = 0$, $b = 1$, $c = 1$, which separately defines the generator loss for a single sample $x \sim X$ as $\mathcal{L}_{\mathcal{G}}(\mathcal{D}, \mathcal{G}, x) = (\mathcal{D}(\mathcal{G}(x)) - 1)^2$ and the discriminator loss for a single pair of samples $x \sim X, y \sim Y$ as $\mathcal{L}_{\mathcal{D}}(\mathcal{D}, \mathcal{G}, x, y) = (\mathcal{D}(y) - 1)^2 + \mathcal{D}(\mathcal{G}(x))^2$. Whereas in an ordinary GAN the generator is given a random vector as input, we want the generator to act as a translator, hence X is instead a source domain containing samples that we want to translate to a target domain Y .

B. Cyclic loss

$\mathcal{G}(x)$ should not only look like it was sampled from Y , but it should also maintain its structure from X . To accomplish this we follow the CycleGAN approach [25] and introduce an inverse mapping $\mathcal{F} : Y \rightarrow X$ and add constraints that force a network to maintain details for performing the inverse mapping, i.e. $\mathcal{G}(\mathcal{F}(y)) \approx y$ and $\mathcal{F}(\mathcal{G}(x)) \approx x$. To enable our network to learn this, we adopt cyclic losses: $\mathcal{L}^{Cyc}(\mathcal{G}, \mathcal{F}, x) = \|\mathcal{F}(\mathcal{G}(x)) - x\|_1$. This loss is applied bidirectionally, i.e. $\mathcal{L}^{Cyc}(\mathcal{G}, \mathcal{F}, x, y) = \mathcal{L}^{Cyc}(\mathcal{G}, \mathcal{F}, x) + \mathcal{L}^{Cyc}(\mathcal{F}, \mathcal{G}, y)$. To train a network that approximates \mathcal{F} , we also need to introduce a discriminator $\mathcal{D}_{\mathcal{F}}$ that tries to learn whether samples are from X or generated using the translator \mathcal{F} .

C. Identity loss

To encourage \mathcal{G} and \mathcal{F} to not change background data for their inputs (which should remain the same for both domains), we need to assure that $\mathcal{G}(y) \approx y$ and $\mathcal{F}(x) \approx x$. In other words, if an input is already in the right output domain for the translator, then the translator should have no effect. We accomplish this using identity losses [26]: $\mathcal{L}_{\mathcal{G}}^{Identity}(\mathcal{G}, y) = \|\mathcal{G}(y) - y\|_1$ and $\mathcal{L}_{\mathcal{F}}^{Identity}(\mathcal{F}, x) = \|\mathcal{F}(x) - x\|_1$.

D. Complete loss functions

For brevity we only show the loss functions for the $X \rightarrow Y$ mapping. Combining the adversarial loss, cyclic loss and identity loss, we now have the following loss functions for the translators:

$$\mathcal{L}_{\mathcal{G}_{X \rightarrow Y}}(\mathcal{G}, \mathcal{F}, \mathcal{D}, x, y) = \Lambda_1 \mathcal{L}_{\mathcal{G}}(\mathcal{D}, \mathcal{G}, x) \quad (1)$$

$$+ \Lambda_2 \mathcal{L}^{Cyc}(\mathcal{G}, \mathcal{F}, x, y) \quad (2)$$

$$+ \Lambda_3 \mathcal{L}_{\mathcal{G}}^{Identity}(\mathcal{G}, y) \quad (3)$$


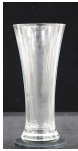



GAN loss weight Λ_1 and identity loss weight Λ_3 are 1.0 and 0.5, respectively. Initial cyclic loss weight Λ_2 is 10.0 and linearly decays to 1.0 at epoch 90. The weights were adopted from the original CycleGAN paper and manually tuned. The weight decay strategy was suggested by Wang and Lin [27].

The loss for the discriminator is defined as follows:

$$\mathcal{L}_{\mathcal{D}_Y}(\mathcal{D}, \mathcal{G}) = (\mathcal{D}(y) - 1)^2 + \mathcal{D}(\mathcal{G}(x))^2$$

Swapping x and y , X and Y , \mathcal{G} and \mathcal{F} gives the generator loss and discriminator loss for the $Y \rightarrow X$ mapping.

TABLE I: Drinking glasses with dimensions in mm

| Object | Wine | Tall | Mid | Low | Slim |
|----------------------|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|
| Picture |  |  |  |  |  |
| Height | 200 | 184 | 122 | 82 | 105 |
| Top \varnothing | 62 | 82 | 80 | 80 | 50 |
| Bottom \varnothing | 71 | 60 | 54 | 50 | 50 |

E. Network architectures

We perform our experiments with both U-Net and ResNet for the generator network architecture and PatchGAN for the discriminator network architecture. Zhu et al. [25] use a custom architecture with residual blocks, but we have experienced faster training times while maintaining similar validation loss using a U-Net architecture [28] for the generator, as originally proposed by Isola et al. [29]. Our network architectures and further training details can be found in the appendix.

III. EXPERIMENTS

A. Dataset creation method

Our test environment is a single level dishwasher, as can be found in restaurants and convenience stores. We created 60 configurations of transparent drinking glasses, and 60 configurations of opaque clones, using the same five object types (see Table I), with at most three of each object type in a scene. In each configuration we vary the position and amount of objects. To create the opaque clones, we spray painted the objects with a brown stone pattern. For each configuration we captured approximately 500 RGBD samples with shape $1280 \times 720 \times 4$, by moving the RGBD camera (Intel RealSense D415) around the configuration in a random pattern. In total we thus have approximately 30000 RGBD samples of dishwasher configurations with transparent objects, and approximately 30000 RGBD samples of dishwasher configurations with opaque objects.

The total time spend on collecting the training data was less than ten hours (excluding preparation time, i.e. painting the objects), with no labeling required, which demonstrates how easy it is to create large datasets with our approach. We created a paired dataset using the same objects in the same environment with different object positions for each sample to quantitatively evaluate our technique. Creating this dataset took less than eight hours, but only contains 41 samples. The paired dataset is split into a test set of 15 samples and a validation set of 26 samples. The test set is used to save the best model during training, while the validation set is used in this paper for the evaluation of the models. The validation set has novel views and object configurations, but the environment and objects are not novel. The paired dataset is useful for evaluating the performance of a trained model, but was not required for training the model itself.

B. Choice of network and modality

In Table II we evaluate results using different network designs as well as other transparent object depth estimation techniques. We evaluated two mapping methods, each trained on the unpaired dataset and evaluated on the paired validation dataset, one method uses direct depth to depth mapping, the other uses RGBD to RGBD mapping. For each method we have tested both the U-Net based generator and the ResNet based generator. We evaluate each network based on the Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Relative error (Rel), percentage of pixels with error less than 5%, less than 10% and less than 25%. All these metrics are only calculated for the area covered by the transparent objects in each sample. These metrics are commonly used for evaluating depth completion effectiveness [30, 4]. We report the number of trainable parameters as an indicator of network size. Our results show that for this task the U-Net architecture produces the best result overall, while the difference between using RGBD and only Depth is minor. We also evaluated the following baseline methods on our validation data:

- Joint Bilateral Filter (JBF) [31]: A statistical method in which we use the RGB input to modify the depth map. We use a Gaussian kernel in the experiment. We performed a grid search to determine the best parameters, testing out kernel size 3×3 , 7×7 , 11×11 and 15×15 , σ_d values 0.5, 1.0, 2.0, 4.0 and 16.0, σ_r values 0.1, 0.5, 1.0, 2.0, 4.0. The reported results were obtained with kernel size 15×15 , $\sigma_d = 16.0$ and $\sigma_r = 2.0$. We chose to compare with JBF as it is a purely statistical method (no training required) and is commonly applied to process depth data in industry.
- ClearGrasp [4]: A recent state-of-the-art method for depth completion of transparent objects by using a surface normal detection network, object boundary detection network and transparent object segmentation network. We chose to compare with ClearGrasp as other recent works have also used it as baseline.
- Ours: RGBD-U-Net, trained on unpaired data. We picked the RGBD-U-Net variant for the comparison, as it operates on similar inputs as JBF and ClearGrasp.

Table III shows the qualitative results of applying methods on some items of our validation set. We show input color, input depth, our result (using RGBD-U-Net), ClearGrasp result (cropped), JBF result (cropped) and ground truth depth. The ground truth depth was acquired by carefully replacing opaque clones of the objects in the place of the transparent objects. The depth range for visualization is clipped to minimum 450 mm, maximum 1000 mm.

As can be seen from the figures, our method often finds a good solution to filling in missing depth pixels based on surrounding depth pixels, especially when a distinctive shape can be observed in the input data. ClearGrasp can dramatically fail to complete the depth of objects, which we assume is caused by the complicated geometry (further discussed below). JBF can produce results that sometimes

TABLE II: Quantitative evaluation of different network architectures and other methods (mean values on validation set)

| Network | Parameters (S) | Parameters (D) | RMSE (m) ↓ | MAE (m) ↓ | Rel ↓ | 1.05 ↑ | 1.10 ↑ | 1.25 ↑ |
|--------------|----------------|----------------|--------------|--------------|--------------|--------------|--------------|--------------|
| RGBD-U-Net | 218,007,172 | 11,165,441 | 0.061 | 0.040 | 0.072 | 0.528 | 0.767 | 0.940 |
| RGBD-ResNet | 35,282,828 | 11,165,441 | 0.092 | 0.074 | 0.135 | 0.250 | 0.482 | 0.853 |
| Depth-U-Net | 217,997,953 | 11,162,369 | 0.058 | 0.035 | 0.061 | 0.589 | 0.861 | 0.954 |
| Depth-ResNet | 35,264,003 | 11,162,369 | 0.145 | 0.128 | 0.229 | 0.113 | 0.232 | 0.581 |
| JBF | - | - | 0.067 | 0.048 | 0.083 | 0.477 | 0.688 | 0.950 |
| ClearGrasp | - | - | 0.090 | 0.057 | 0.120 | 0.404 | 0.555 | 0.840 |

TABLE III: Qualitative results of our method, ClearGrasp and JBF.



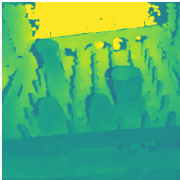




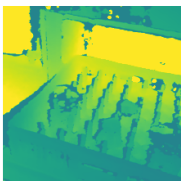
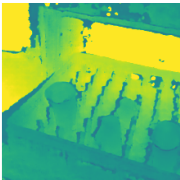
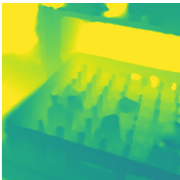

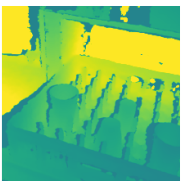

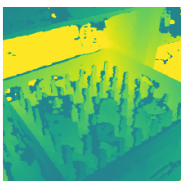
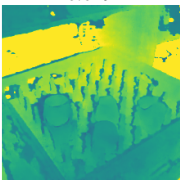

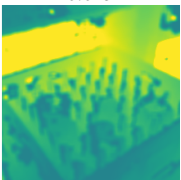
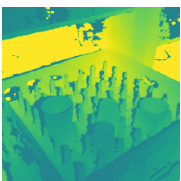



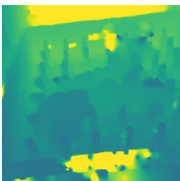




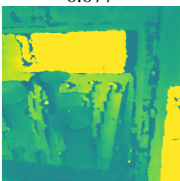
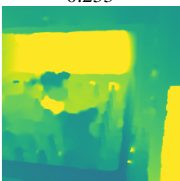




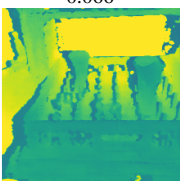
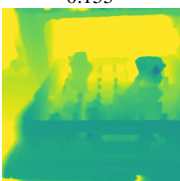
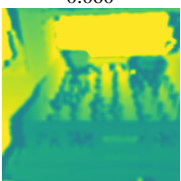

| Captured Color | Captured Depth | Our result | ClearGrasp result | JBF result | Ground truth depth |
|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| Three samples with the lowest MAE using our method | | | | | |
|  |  |  0.024 |  0.070 |  0.039 |  |
|  |  |  0.029 |  0.049 |  0.043 |  |
|  |  |  0.030 |  0.044 |  0.038 |  |
| Three samples with the highest MAE using our method | | | | | |
|  |  |  0.077 |  0.233 |  0.051 |  |
|  |  |  0.066 |  0.155 |  0.060 |  |
|  |  |  0.055 |  0.080 |  0.079 |  |



Fig. 2: A grasp of a transparent object performed by our robot using our network for point cloud depth completion.

even outperforms our method, but it introduces Gaussian noise which might complicate downstream tasks.

C. Object manipulation using humanoid robot

We evaluated grasping of transparent objects using a humanoid robot (RT Corporation Scirus 17). ROS [32] and MoveIt [33] were used to control the robot, with motion planning and sensor processing executed on the built-in Intel Corporation NUC7i7BNH, and transparent depth estimation performed on an external notebook (CPU: Intel Core i9-10980HK). The frame rate of transparent depth completion was around 2 FPS, using only CPU. We attempted 42 grasps using 2 objects from our training set: Tall and Wine. We only attempted grasps in the region in which the robot could grasp the opaque version of the objects. We use the RGBD-U-Net network architecture. To select the object to be grasped, we first capture the background depth map without the object present. After placing the object, we calculate the difference between the point cloud with the object present and the point cloud without the object present, giving us an estimate of the object position. The X, Y coordinates in the camera image are translated to a 3D point using the mean depth reported for the object. We count grasps as successful if the gripper can firmly grip the object. For the tall glass, 18 out of 21 attempts were successful. For the wine glass, 21 out of 21 attempts were successful. Figure 2 gives an example of a successfully executed grasp of a transparent object with the robot.

IV. DISCUSSION AND LIMITATIONS

Our method outperforms ClearGrasp on our dataset. Figure 3 shows an example pair of surface normals, occlusion edges and masks generated by ClearGrasp on a sample from our validation set. As can be seen in the images, it seems that ClearGrasp underperforms in complex geometry settings due to an abundance of occlusion edges being present. Our method is trained end-to-end on RGBD data, so it

does not suffer from this limitation. More details about this experiment can be found in the online supplemental material.

We have also tried to apply transfer learning on ClearGrasp using data generated using a custom simulation constructed using SuperCaustics [34, 35], a transparent object dataset generator that used real time ray tracing and generates data that is compatible with ClearGrasp. We have spend roughly the same amount of time generating the assets (drinking glass models and dishwasher model) as we have spend recording training data for our method. Figure 4 shows some example images of this new dataset. After training for 100 additional epochs using 3802 samples of additional domain specific training data added to the ClearGrasp training dataset, we were unable to get better results than with the base ClearGrasp model. More details about this experiment can be found in the online supplemental material.

A limitation of our system is that our system does not generalize well to unknown objects, which are either left unchanged in the output point cloud or are replaced by one of the objects in our training set. This shows that our system specifically learns how to complete the depth of objects in the training set, and does not learn general properties of transparency. Introducing a larger variety of shapes in the training set could increase the capability of the network to generalize to new objects. We performed a small experiment with four novel objects in the same environment used in this paper, for which the results are reported in the online supplemental material.

We trained a network to directly infer depth from transparent depth or RGBD. This has acceptable results in our case, but there are more sophisticated methods of completing depth. ClearGrasp for example combines surface normal vectors, contact/occlusion edges and masks with the original noisy depth map to complete depth. It is possible to engineer features such as this using our method, by for example using an HSV filter to create masks of opaque objects and generating surface normal vectors from the opaque depth map, but we leave the exploration of this for future work.

It is likely that there is a combination of hyper-parameters, loss functions, network designs and gradient normalization techniques that could improve the results. There are many choices for adversarial loss functions for GAN's, but we did not perform a thorough analysis to compare different loss functions. Some works suggest that the choice of loss function is less important than the choice of hyper-parameters [36], but different loss functions could contribute to improving the training time and quality of the results. Besides the choice of loss function, there are ways to normalize the gradients of both the generator and the discriminator. We only did limited experimentation with this, but we did not manage to further improve the results.

V. CONCLUSION

We have presented a technique that allows easy creation of datasets for training the vision system of a robot for performing transparent object depth completion in real environments that have complicated geometry. There are many tasks that

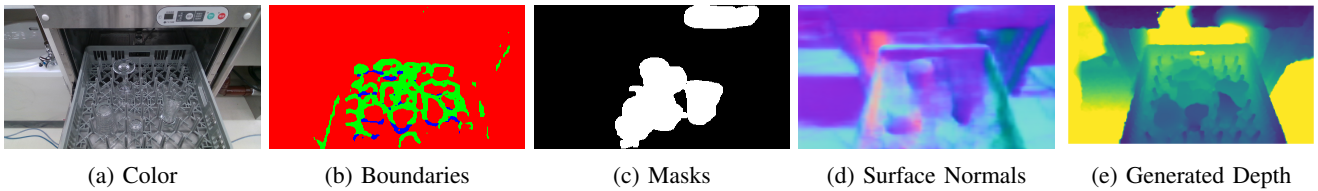


Fig. 3: Typical output of running ClearGrasp on our validation set.

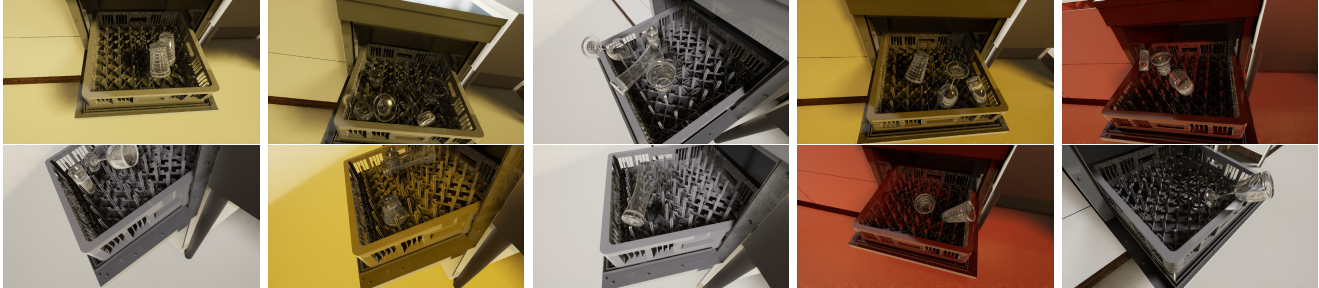


Fig. 4: Examples of simulated training data. Corresponding boundaries, masks and surface normals were also generated.

we want robots to perform in the future in environments such as convenience stores and restaurants. We imagine that robots can have different programs to deal with each task, switching program depending on the task at hand, with models trained using the techniques discussed in this paper. In the future we want to explore using our technique for tasks such as teaching a robot where to grasp objects, how to segment an object into parts and how to find particular objects in a scene.

APPENDIX

We use the Adam optimizer [37] with initial learning rate of $2e^{-4}$ for the generator and $1e^{-4}$ for the discriminator. After epoch 100 we start to linearly decrease the learning rate to 0 for both the generator and the discriminator over 100 epochs. Each network is trained for 200 epochs in total. Each epoch consists of 1000 training steps. We sequentially train both generators and both discriminators with one sample at each step. The activation function for the final layer of both types of generators is a hyperbolic tangent. Training of each network can take up to 48 hours using a single NVIDIA Tesla V100 GPU on a PC equipped with Intel Xeon E5-2698v4. We trained multiple networks in parallel on one machine, so training time might be negatively affected. All implementations use TensorFlow [38]. When loading the data, we perform random horizontal crops and random horizontal flips, producing an RGBD patch of 512x512 pixels. Depth is clipped between 450 mm (the minimum depth of the RGBD sensor) and 2000 mm.

The U-Net generator is adopted from Isola et al. [29, 39]. U-Net is an encoder-decoder architecture with skip connections. Schematically, the generator encoder architecture can be represented as $C64-C128-C256-C512-C1024-C1024-C1024-C1024-C1024$, the generator decoder layer as $CD1024-CD1024-CD1024-C1024-C512-C256-C128-C64-C(1/3/4)$. Ck represents a Convolution-InstanceNormalization-ReLU layer with k filters. CDk

represents a Convolution-InstanceNormalization-Dropout-ReLU layer with k filters and a dropout rate of 50%. All convolution layers are 4×4 spatial filters with stride 2, downsampling in the encoder and upsampling in the decoder. Ordinary ReLU activation is used in the encoder, while LeakyReLU with slope 0.2 is used in the decoder. In the final layer of the decoder, $C1$ is used for Depth only, $C3$ for RGB, $C4$ for RGBD. Skip connections concatenate activations from layer i to layer $n-i$. Instance normalization is used in every layer, except for the first encoder layer.

The ResNet generator is adopted from Zhu et al. [25, 40]. It can be represented as $c7s1-64, d128, d256, R256, R256, R256, R256, R256, R256, R256, R256, u128, u64, c7s1-(1/3/4)$. We use 9 ResNet blocks. $c7s1-k$ denotes a 7×7 Convolution-InstanceNorm-ReLU layer with k filters and stride 2. $R256$ denotes a residual block with two 3×3 Convolution-InstanceNorm-ReLU layers with 256 filters. uk denotes a 3×3 Deconvolution-InstanceNorm-ReLU layer with k filters and stride 2. Reflect padding is used in the ResNet blocks to reduce artifacts.

We use a PatchGAN discriminator [41, 42, 29]. The discriminator architecture can be represented as $C64-C128-C256-C512-C1024$. Instance normalization is used in every layer, except for the first layer. A Leaky ReLU activation with slope 0.2 is used in every layer. A convolution with stride 1, filter size 1 and kernel size 4 is used to produce a discriminated output map.

ACKNOWLEDGMENT

This work was supported by JST [Moonshot R&D][Grant Number JPMJMS2031]. This research is subsidized by New Energy and Industrial Technology Development Organization (NEDO) under a project JPNP20016. This paper is one of the achievements of joint research with and is jointly owned copyrighted material of ROBOT Industrial Basic Technology Collaborative Innovation Partnership.

REFERENCES

- [1] Ashutosh Saxena, Justin Driemeyer, and Andrew Y. Ng. “Robotic Grasping of Novel Objects using Vision”. In: *The International Journal of Robotics Research* 27.2 (2008), pp. 157–173. DOI: [10.1177/0278364907087172](https://doi.org/10.1177/0278364907087172).
- [2] Ilya Lysenkov, Victor Eruhimov, and Gary Bradski. “Recognition and pose estimation of rigid transparent objects with a kinect sensor”. In: *in Proc. of Robotics: Science and Systems*. 2012. DOI: [10.15607/rss.2012.viii.035](https://doi.org/10.15607/rss.2012.viii.035).
- [3] Ilya Lysenkov and Vincent Rabaud. “Pose estimation of rigid transparent objects in transparent clutter”. In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 162–169. DOI: [10.1109/ICRA.2013.6630571](https://doi.org/10.1109/ICRA.2013.6630571).
- [4] Shreeyak Sajjan et al. “Clear Grasp: 3D Shape Estimation of Transparent Objects for Manipulation”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 3634–3642. DOI: [10.1109/ICRA40945.2020.9197518](https://doi.org/10.1109/ICRA40945.2020.9197518).
- [5] Luyang Zhu et al. “RGB-D Local Implicit Function for Depth Completion of Transparent Objects”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 4647–4656. DOI: [10.1109/CVPR46437.2021.00462](https://doi.org/10.1109/CVPR46437.2021.00462).
- [6] Thomas Weng et al. “Multi-modal Transfer Learning for Grasping Transparent and Specular Objects”. In: *IEEE Robotics and Automation Letters* 5.3 (July 2020). arXiv: 2006.00028, pp. 3791–3798. ISSN: 2377-3766, 2377-3774. DOI: [10.1109/LRA.2020.2974686](https://doi.org/10.1109/LRA.2020.2974686).
- [7] Xingyu Liu et al. “KeyPose: Multi-View 3D Labeling and Keypoint Estimation for Transparent Objects”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 11599–11607. DOI: [10.1109/CVPR42600.2020.01162](https://doi.org/10.1109/CVPR42600.2020.01162).
- [8] Zheming Zhou, Xiaotong Chen, and Odest Chadwicke Jenkins. “LIT: Light-Field Inference of Transparency for Refractive Object Localization”. In: *IEEE Robotics and Automation Letters* 5.3 (2020), pp. 4548–4555. DOI: [10.1109/LRA.2020.3001499](https://doi.org/10.1109/LRA.2020.3001499).
- [9] Agastya Kalra et al. “Deep Polarization Cues for Transparent Object Segmentation”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 8599–8608. DOI: [10.1109/CVPR42600.2020.00863](https://doi.org/10.1109/CVPR42600.2020.00863).
- [10] Yiming Qian, Minglun Gong, and Yee-Hong Yang. “3D Reconstruction of Transparent Objects with Position-Normal Consistency”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, June 2016, pp. 4369–4377. ISBN: 978-1-4673-8851-1. DOI: [10.1109/CVPR.2016.473](https://doi.org/10.1109/CVPR.2016.473). URL: <http://ieeexplore.ieee.org/document/7780842/>.
- [11] Haoping Xu et al. “Seeing Glass: Joint Point Cloud and Depth Completion for Transparent Objects”. In: *Conference on Robot Learning (CoRL)*. 2021. URL: <https://arxiv.org/abs/2110.00087>.
- [12] Hongjie Fang et al. “TransCG: A Large-Scale Real-World Dataset for Transparent Object Depth Completion and Grasping”. In: *arXiv preprint arXiv:2202.08471* (2022).
- [13] Jiaying Lin, Zebang He, and Rynson W.H. Lau. “Rich Context Aggregation with Reflection Prior for Glass Surface Detection”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 13410–13419. DOI: [10.1109/CVPR46437.2021.01321](https://doi.org/10.1109/CVPR46437.2021.01321).
- [14] Enze Xie et al. “Segmenting transparent object in the wild with transformer”. In: *arXiv preprint arXiv:2101.08461* (2021).
- [15] Ben Mildenhall et al. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. Aug. 2020. DOI: [10.48550/arXiv.2003.08934](https://doi.org/10.48550/arXiv.2003.08934). arXiv: 2003.08934 [cs].
- [16] Jeffrey Ichnowski et al. “Dex-NeRF: Using a Neural Radiance Field to Grasp Transparent Objects”. In: *arXiv:2110.14217 [cs]* (Oct. 2021). arXiv: [2110.14217](https://arxiv.org/abs/2110.14217) [cs].
- [17] Floris Erich et al. “Neural Scanning: Rendering and determining geometry of household objects using Neural Radiance Fields”. In: *2023 IEEE/SICE International Symposium on System Integration*. 2023. DOI: [10.1109/SII55687.2023.10039147](https://doi.org/10.1109/SII55687.2023.10039147).
- [18] Qiyu Dai et al. “GraspNeRF: Multiview-based 6-DoF Grasp Detection for Transparent and Specular Objects Using Generalizable NeRF”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023. URL: <http://arxiv.org/abs/2210.06575>.
- [19] Justin Kerr et al. “Evo-NeRF: Evolving NeRF for Sequential Robot Grasping of Transparent Objects”. In: *2022 Conference on Robot Learning*. 2022. URL: <https://openreview.net/forum?id=Bxr45keYrf>.
- [20] Kanishka Rao et al. “RL-CycleGAN: Reinforcement Learning Aware Simulation-to-Real”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 11154–11163. DOI: [10.1109/CVPR42600.2020.01117](https://doi.org/10.1109/CVPR42600.2020.01117).
- [21] Daniel Ho et al. “RetinaGAN: An Object-aware Approach to Sim-to-Real Transfer”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 10920–10926. DOI: [10.1109/ICRA48506.2021.9561157](https://doi.org/10.1109/ICRA48506.2021.9561157).
- [22] Yingjie Tang et al. “DepthGrasp: Depth Completion of Transparent Objects Using Self-Attentive Adversarial Network with Spectral Residual for Grasping”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 5710–

5716. DOI: [10 . 1109 / IROS51168 . 2021 . 9636382](https://doi.org/10.1109/IROS51168.2021.9636382).
- [23] Ian Goodfellow et al. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014.
- [24] Xudong Mao et al. “Least Squares Generative Adversarial Networks”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2813–2821. DOI: [10.1109/ICCV.2017.304](https://doi.org/10.1109/ICCV.2017.304).
- [25] Jun-Yan Zhu et al. “Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2242–2251. DOI: [10 . 1109/ICCV.2017.244](https://doi.org/10.1109/ICCV.2017.244).
- [26] Yaniv Taigman, Adam Polyak, and Lior Wolf. “Unsupervised Cross-Domain Image Generation”. In: *International Conference on Learning Representations (ICLR)*. 2017.
- [27] Tongzhou Wang and Yihan Lin. “CycleGAN with Better Cycles”. In: (). URL: https://www.tongzhouwang.info/better_cycles/report.pdf.
- [28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by Nassir Navab et al. Cham: Springer International Publishing, 2015, pp. 234–241. ISBN: 978-3-319-24574-4.
- [29] Phillip Isola et al. “Image-to-Image Translation with Conditional Adversarial Networks”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 5967–5976. DOI: [10.1109/CVPR.2017.632](https://doi.org/10.1109/CVPR.2017.632).
- [30] Yinda Zhang and Thomas Funkhouser. “Deep Depth Completion of a Single RGB-D Image”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 175–185. DOI: [10 . 1109/CVPR.2018.00026](https://doi.org/10.1109/CVPR.2018.00026).
- [31] Johannes Kopf et al. “Joint Bilateral Upsampling”. In: *ACM Trans. Graph.* 26.3 (July 2007), 96–es. ISSN: 0730-0301. DOI: [10 . 1145 / 1276377 . 1276497](https://doi.org/10.1145/1276377.1276497). URL: <https://doi.org/10.1145/1276377.1276497>.
- [32] Morgan Quigley et al. “ROS: an open-source Robot Operating System”. In: *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*. 2009.
- [33] David Coleman, Sachin Chitta, and Nikolaus Correll. *Reducing the barrier to entry of complex robotic software: a moveit! case study*. *Journal of Software Engineering for Robotics*. 2014.
- [34] Mehdi Mousavi, Aashis Khanal, and Rolando Estrada. “AI Playground: Unreal Engine-Based Data Ablation Tool for Deep Learning”. In: *Advances in Visual Computing*. Cham: Springer International Publishing, 2020, pp. 518–532. ISBN: 978-3-030-64559-5.
- [35] Mehdi Mousavi and Rolando Estrada. *SuperCausatics: Real-time, open-source simulation of transparent objects for deep learning applications*. 2021. arXiv: [2107.11008 \[cs.GR\]](https://arxiv.org/abs/2107.11008).
- [36] Mario Lucic et al. “Are GANs Created Equal? A Large-Scale Study”. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems. NIPS’18*. 2018, pp. 698–707.
- [37] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations (ICLR)*. 2015. URL: <https://arxiv.org/abs/1412.6980>.
- [38] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: <https://www.tensorflow.org/>.
- [39] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. In: *International Conference on Learning Representations (ICLR)*. 2016. URL: <https://arxiv.org/abs/1511.06434>.
- [40] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. “Perceptual Losses for Real-Time Style Transfer and Super-Resolution”. In: *European Conference on Computer Vision (ECCV)*. 2016. URL: <https://arxiv.org/abs/1603.08155>.
- [41] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. *Unsupervised Image-to-Image Translation Networks*. 2018. URL: <https://arxiv.org/abs/1703.00848>.
- [42] Chuan Li and Michael Wand. *Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks*. 2016. arXiv: [1604 . 04382 \[cs.CV\]](https://arxiv.org/abs/1604.04382).