Proceeding of the IEEE
International Conference on Robotics and Biomimetics
Dali, China, December 2019

Visual Completion Of 3D Object Shapes From A Single View For Robotic Tasks

Mohamed Tahoun[1,2], Carlos M. Mateo[2] Juan-Antonio Corrales-Ramón[2],
Omar Tahri[1], Youcef Mezouar[2] and Pablo Gil[3]

***Abstract*— The goal of this paper is to predict 3D object shape to improve the visual perception of robots in grasping and manipulation tasks. The planning of image-based robotic manipulation tasks depends on the recognition of the object's shape. Mostly, the manipulator robots usually use a camera with configuration eye-in-hand. This fact limits the calculation of the grip on the visible part of the object. In this paper, we present a 3D Deep Convolutional Neural Network to predict the hidden parts of objects from a single-view and to accomplish recovering the complete shape of them. We have tested our proposal with both previously seen objects and novel objects from a well-known dataset.**

I. INTRODUCTION

Knowing the complete 3D geometry of an object is indispensable for the physical interaction between the robots and the outside world such as object recognition, grasping, and object manipulation. In this work, we aim to tackle the problem of occlusion in grasping and manipulation tasks through predicting the complete 3D shape from a single 2.5D depth view. If the shape of the object was known, robots could get some ideas of what actions to consider like path planning and generating stable grasps. For this objective, we designed and trained a 3D convolutional neural network to do the shape reconstruction. This is a very challenging task because different 3D models can be obtained from the same single view. Therefore, our solution should have the ability of generalization.

We trained the model on a real depth map of handheld objects from YCB dataset [1]. These depth maps were segmented and voxelized to fit inside an occupancy grid to be the input of the CNN model. Also, the target 3D shape is voxelized to fit inside an occupancy grid of the same resolution to be compared with the predicted output. Figure 1 shows the runtime pipeline of the reconstruction operation.

The contributions of this work include: 1) A novel CNN architecture for 3D shape completion from a single arbitrary depth view; 2) An end-to-end trainable deep learning model with real depth views of multiple household objects.

[1] INSA Centre Val de Loire, PRISME Laboratory EA4229, F-18000 Bourges, France. mohamed.tahoun, omar.tahri@insa-cvl.fr

[2] Université Clermont Auvergne, CNRS, SIGMA Clermont, Institut Pascal, F-63000 ClermontFerrand, France. cmateoagul, juan-antonio.corrales-ramon, youcef.mezouar@sigma-clermont.fr

[3] AUROVA Lab, Department of Physics, Systems Engineering and Signal Theory, University of Alicante, 03690, Spain. pablo.gil@ua.es

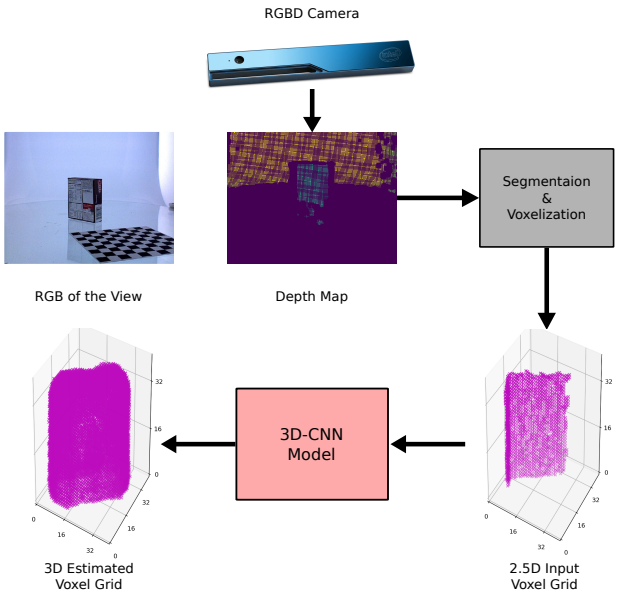

Fig. 1: Pipeline of reconstruction operation

The rest of the paper is structured as follows. Section II addresses the related works. Section III describes the proposed method based on CNN for reconstruction objects. The pipeline for processing the input data, the dataset, and the training methodology as well as the used evaluation metric are described in section IV. Followed by a discussion on the reconstruction results for both seen and novel objects in section V. Finally, we commented on the conclusions and future works in section VI.

II. RELATED WORKS

3D shape completion from depth maps or partial scans has been studied widely in robotics, computer vision, and computer graphics. There exist many researches on shape reconstruction from an incomplete point cloud. A detailed survey can be found in [2]. Traditional completion methods commonly use interpolation techniques to predict the underlying 3D structure, such as plane fitting [3] or Poisson surface estimation [4]. However, these methods are only suitable to refine reconstructed surfaces by filling holes and gaps on visible parts, but not the whole surface including occluded faces. In the past, several works focused on completing shapes were presented, for example, through shapes structure

978-1-7281-6321-5/19//$31.00 ©2019 IEEE

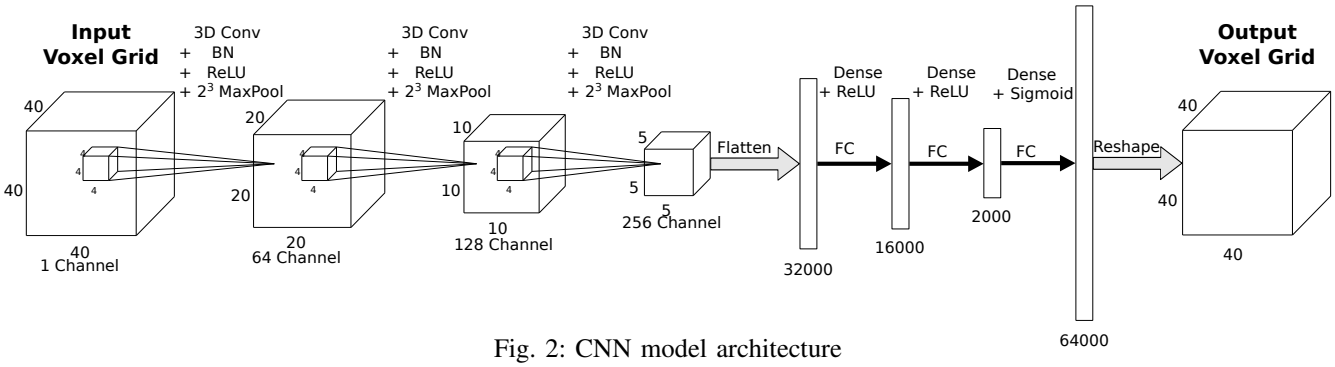

Fig. 2: CNN model architecture

and regularities [5] and based on the learning of shape priors [6] .

Instead, this paper addresses the problem of shape prediction using a Deep Learning scheme to avoid a reconstruction process. Recently, Han et al. presented in [7] a detailed survey on 3D object reconstruction based on learning methods, where the authors highlight the fact that the most representative methods are based on a volumetric representation of surfaces, instead of meshes or point clouds.

In previous works such as [8] and [9], the authors used furniture objects as input, which are not realistic in robotic manipulation tasks, due to the object's size. Yang et al. [8] and Dai et al. [10] proposed complex models based on Generative Adversarial Network (GAN) to recover the occluded regions. In both works, an encoder-decoder model was used as the generator architecture with up to ten convolutional layers. In the proposed method, we used a lighter network which is important when the robot manipulator have to be adapted to work in scenarios with new objects. Additionally, our method takes a volumetric representation as it has been done in [8] but with less resolution requirements. Specifically, Wu et al. [9] presented a model to transfer the learning from synthetic data 2.5D to real objects. To do this, they trained their method with chairs representations on the Pascal 3d+ dataset. Unlike our work, they trained from a combination of depth map, normal vectors and silhoutte, instead of a volumetric representation as we did from YCB dataset [1]. Also, another method presented in [11] used a model based on autoencoder with two encoders and one decoder. RGB image and silhouette were used as input to each encoder and outputting several 2.5D surfaces that are needed to be aligned and fused. Conversely, we are outputting directly one complete 3D shape. Here, we propose a simpler model based on 3D CNN for manipulation of household objects. It is able to learn from real data as well as it requires less parametric adjustments. Similarly, Varley et al. [12] trained their 3D CNN model for object completion but using synthesized depth maps, which are free of noise. Whereas, our model is trained on real data to augment the learning generalization.

III. THE PROPOSED APPROACH

In this work, we present an approach to reconstruct the object surface from a single view obtained from an RGBD camera. In particular, we use a surface-based partial representation of 3D object shape to predict its whole shape. To do this, we use an approach based on deep learning techniques.

More specifically, we have implemented a 3D-CNN (3 Dimensional Convolutional Neural Network) which takes as input a 2.5D surface (depth map) observed from a viewpoint and generates as output a 3D predicted surface. Our model is shown in Fig. 2. Both, the input denoted as I and the output denoted as O, are voxel grids of a resolution 40^3. This resolution was empirically selected to get a good prediction with low memory requirements. In a grid, the occupied voxel contains 1 while the empty one contains 0. I represents the occupied voxels of the 2.5D view while O registers the occupied voxels of the prediction for the complete 3D object.

The proposed model consists of three 3D convolutional layers, each of which has a bank of 4x4x4 filters with strides of 1x1x1, then a Batch Normalization (BN), followed by a ReLU activation function and, a Max Pooling Layer which has 2x2x2 filters and strides of 2x2x2. The number of output channels for the max pooling layer starts with 64, doubling at each subsequent layer and ends up with 256. Followed by a three Fully Connected (FC) layers with a ReLU activation except for the last layer in which is used a sigmoid function as activation to restrict the output to be in this range (0, 1). Between the last convolutional layer and the first FC, we need to adapt the data. For that, we have used a Flatten layer to convert the multi-dimensionality of the output provided by the convolutional into a 1D.

IV. METHODOLOGY

A. Dataset

We employed the YCB dataset [1], which contains a wide variety of objects in size and shape that create the used dataset. Our dataset consists of 38 objects, where 63% of the objects were used for both training and validation and, the rest for testing (14 objects). This way, we assured that the testing objects were not previously seen by our CNN model during the training and validation process. Fig. 3 and Fig. 4 show the representation of the training and the testing subsets, respectively.

Yale dataset provides 600 partial views of each object with 5 cameras (120 views for each camera) as described in [13], each of them consists of an RGB image, a depth map, a camera calibration parameters, the object point cloud and a

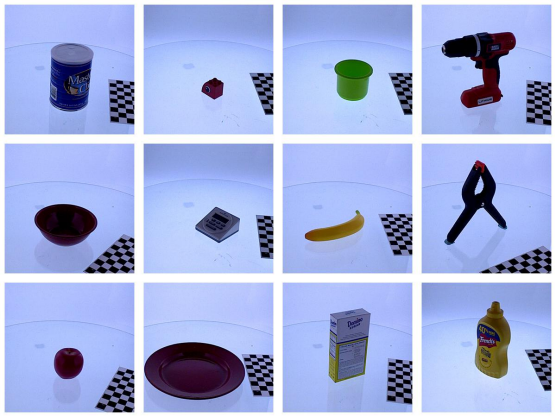

Fig. 3: Sample of training objects (big can, bowl, apple, small Lego, timer, plate, wide cup, banana, sugar box, power drill, large clamp, mustard bottle)

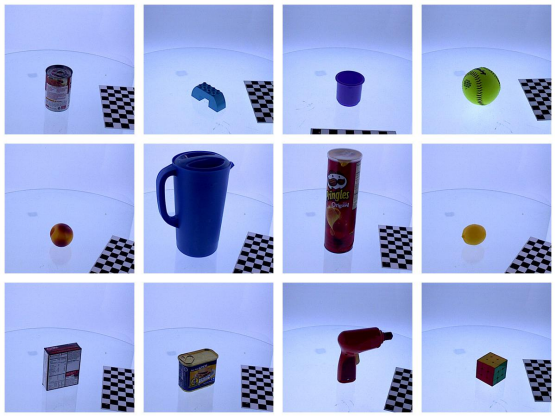

Fig. 4: Samples of testing objects (small can, peach, short box, big Lego, pitcher base, squared rounded can, small cup, chips can, toy drill, tennis ball, lemon, Rubik's cube)

mask to filter the object from the scene. We did not use the RGB in this work, while our Ground-Truth is the object point cloud.

We manually selected only 68 views for each object to make sure they were quite different, and they did not provide similar information for the training model. So, redundancies of the closest views have been avoided. Therefore, each training object is composed of 15 views acquired from four cameras plus 8 views from the overhead camera.

B. Pre-processing training data

As shown in Fig.2, the input of the model should be a 40^3 3D voxel grid as explained in section III. So, we need to pre-process the depth map of each input view to obtain the 3D voxel grid of the occluded view. To do this, we follow the next steps.

First of all, we have to filter the input depth map of the scene to obtain the cluster which represents the object. Object depth maps have different reference frames because they were captured from five different cameras in five different locations. In this case, we used the calibration information for each camera, which was supplied in the dataset, to change the reference frame from the camera to a point on the table. In this way, we accomplished that all object views have the same reference frame to obtain a partial point cloud $\mathcal{V}$. Also, it is the same reference frame used by the whole point cloud supplied in the dataset, which is used as Ground-Truth, $\mathcal{C}$.

Second, the partial cloud $\mathcal{V}$ is filtered to eliminate noisy points because we are dealing with real depth information. Then, we used a down-sampling filter of a size equal to $0.002m$ on $\mathcal{C}$ to get a better performance in the matching process with $\mathcal{V}$. As the filter size is increased, more details will be lost, inversely, decreasing it will be computationally expensive during the matching process. Later, we compared $\mathcal{C}$ and $\mathcal{V}$ by applying a distance filter which allows a difference of $0.005m$ between both clouds.

Finally, we got the occupied voxels in the volumetric grid $\mathbb{V}_{\mathcal{C}}$ (40^3) which comprises the merged cloud $\mathcal{C}$. We normalized the merged point cloud inside the occupancy grid $\mathbb{V}_{\mathcal{C}}$, by calculating the maximum $\bar{x}_{\max}$ and the minimum number of steps $\bar{x}_{\min}$ in each dimension. As well as the step value N based on the grid dimension, applying the following equations:

$$N = \frac{\max\left(\bar{x}_{\max} - \bar{x}_{\min}\right)}{40} \tag{1}$$

$$n_{\bar{x}} = \left\lfloor \frac{\bar{x}_{\max} - \bar{x}_{\min}}{N} \right\rfloor \tag{2}$$

where $\bar{x} = (x, y, z)$, are the 3D coordinates of the point cloud, while $\bar{x}_{\max}$ and $\bar{x}_{\min}$ the maximum and minimum in each coordinate.

Once, we got the occupied voxels of the $\mathcal{C}$, we fit $\mathcal{V}$ inside the volumetric grid $\mathbb{V}_{\mathcal{V}}$ with the same $\mathbb{V}_{\mathcal{C}}$ voxels size. Figure 1 provides an example of the input and the output voxel grids.

TABLE I: Different CNN Models

	Channels	**Kernel size**	**Padding**	**BN**
CNN_1	64x64x64	4x4x4	valid	True
CNN_2	16x32x64x128	4x4x4	valid	True
CNN_3	16x32x64	4x4x4	valid	True
CNN_4	64x128x256	4x4x4	valid	True
CNN_5	64x128x256	4x4x4	same	True
CNN_6	64x128x256	5x4x3	valid	True
CNN_7	64x128x256	4x4x4	same	False

C. Training and tuning

In order to demonstrate the performance of our model in reconstruction tasks, we have tested different configurations for the proposed CNN model as specified in table I. During the configuration phase, we fixed the number of epochs to 20, the batch size to 16, and the learning rate to $5e^{-5}$. We selected these parameters after several trials of different learning rates, batch sizes, and epochs. In figure 5, we found that the best learning rate for our model should be between $1e^{-5}$ and $1e^{-4}$.

Then, we started to test each model shown in Table I and get the average of the Mean Squared Error (MSE) on the testing objects for each model as shown in Figure 6. From

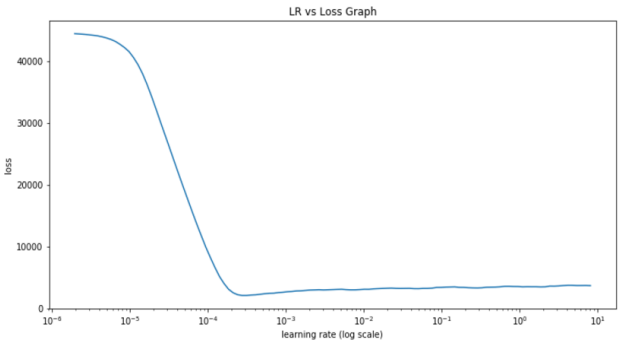

Fig. 5: Different learning rates vs. Loss

this comparison, we found that the best model which has the least MSE is model CNN_5 in Table I. We used BN between the CNN layers to normalize the inputs of each layer, where the input from prior layers could be changed after the weights update. In addition, using BN accelerate the training process because it helps the model to train in fewer epochs.

In Fig. 7, we present the achieved loss rate during training and validation steps for the problem of learning to reconstruct surfaces via generating voxels with the best configuration. We used the binary cross-entropy error loss function:

$$Cost = E(\bar{y}, \bar{z}) = -(\bar{y}\log(\bar{z}) + (1-\bar{y})\log(1-\bar{z})) \quad (3)$$

where $\bar{y} \in \mathbb{R}^m$ is the flattened version of $\mathbb{V}_{\mathcal{C}} \in \mathbb{R}^{n^3}$ after applying a mapping function $f : \mathbb{V} \rightarrow \bar{y}$. Similarly, $\bar{z}$ is the flattened version of the predicted occupancy grid $\mathbb{V}_{\mathcal{P}}$. Hence, we have that $m = 64000$ elements of the flattened 1D vector and $n = 40$ voxels in each dimension. While, each value y_i of $\bar{y}$ can be $\{0, 1\}$, and the values of z_i of $\bar{z}$ are real numbers bounded in $(0, 1)$. The cost function boosts the output to tends to be either 1 for the occupied goal voxels and 0 for the empty ones. The chosen optimizer was Adam [14] with the default hyper-parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^-8$), and the learning rate was set to $5e^-5$. Additionally, we used he normal initializer [15] to initialize the model kernel weights.

After several tests (Fig. 6), we set the number of epochs to 10 according to the loss function shown in (Fig. 7) to avoid

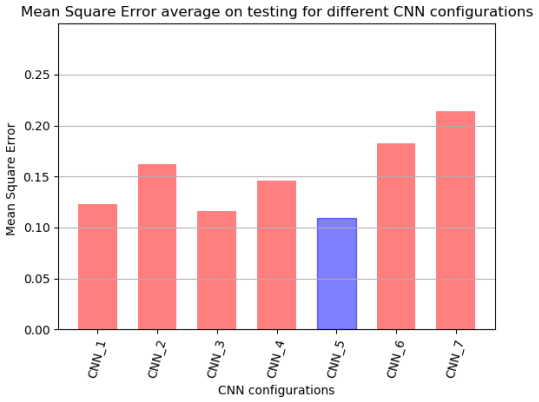

Fig. 6: Comparison of the MSE obtained varying the parameters of our model with the parameters from Table I

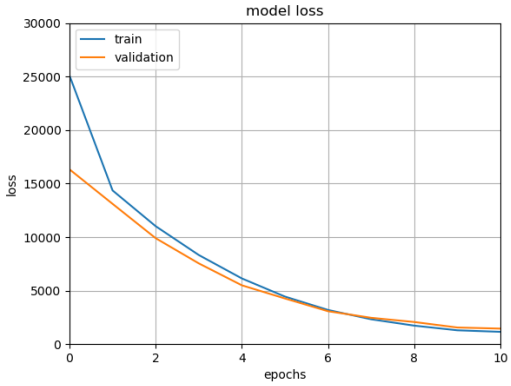

Fig. 7: The training and validation loss for the best configuration of parameters shown in Table I

overfitting problems, and the batch size to 16. During the training process, we applied k-fold cross validation which improves the Monte-Carlo estimation in relation to solely perform the tests with a single random partition. Therefore, our training dataset was consequently divided into mutually exclusive sub-sets, using each view from an object only in one partition. In this way, the views of the same object cannot be repeated in both sub-sets, training, and validation. Although, the observed objects during training were always seen from different viewpoints. For each fold, we used one of the partitions for validation (20% of the samples) and the rest for training (80%). Both training and validation processes were repeated for $k = 5$ times, using different partitions of our dataset. Later, the average result and its standard deviation are provided.

D. Evaluation metrics

We have used the Mean Squared Error (MSE) metric to evaluate the performance of our proposal. MSE basically measures the average squared error of our predictions. This calculates the square difference between the predictions and the target, then computes the average of these values for each voxel as follows,

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y_i - z_i)^2. \quad (4)$$

If MSE value starts decreasing, it means that the performance of the model will be better. Oppositely, the performance becomes worse if it increases.

V. RESULTS AND EXPERIMENTATION

This model was implemented using Keras 2.2.5 based on TensorFlow 1.14 library. To do this experiment, all the computations have been performed using the NVIDIA Tesla P40 GPU. We have used the MSE to evaluate the performance of our model as explained section IV-D.

A. Prediction on seen objects

In this experiment, we tested our model on unseen views of the training objects (Fig. 3). We calculated the MSE average of each object estimation to compare the reconstruction error

between different objects as shown in Fig. 8. While, Fig. 9, visualises the completion results of three different objects (a sugar box, a mustard bottle and a tuna can). Where these object are different in geometry and size, and were viewed from different camera poses. In both cases, the error is smaller than 0.025 over 1 when we trained with all views for the same objects. Although the tuna can was the worst rebuilt, the error is acceptable. Perhaps, it was because of its input viewpoint is worse than others.

B. Prediction on novel objects

In order to show the generalization capabilities of our proposal, we present the reconstruction results obtained from testing on 14 novel objects (Fig. 4). Fig. 10 shows the MSE average of the predicted objects from 21 different views for each object.

The visual perception of the prediction can be seen in Fig. 11, in which we show some examples for objects with different geometry like tomato soup can, Rubik's cube, lemon and Lego part. Although our method used complicated views as input, it predicted fairly good the cylindrical objects such as the soup can with an error around 0.10 over 1. This can be extrapolated to other revolution objects like the lemon with a spherical shape. Although, the Rubik's cube was observed from better viewpoints than others, this experiments grew in the MSE. Since edges between faces were more difficult to generalize than spherical (convex and concave) shapes.

C. Performance Analysis

To measure the performance, we have analyzed the behavior of our model according to shape, size, and viewpoint. To do this, we have considered the validation and testing experiments, separately. The performance which depends on shape, has been studied using four categories such as cylindrical, spherical, rectangular and irregular; plus two categories for size, these are big and small; and three final categories for the observation angles such as top, bottom, lateral. Then, we compare the average of MSE for each category as shown in Table II.

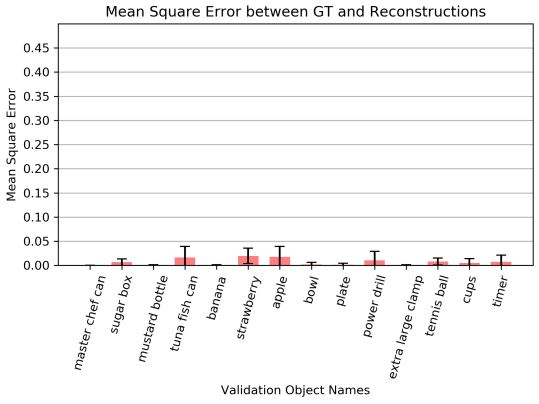

Fig. 8: Average of Squared Mean Error (SME) of the reconstruction of selection from seen objects but unseen views

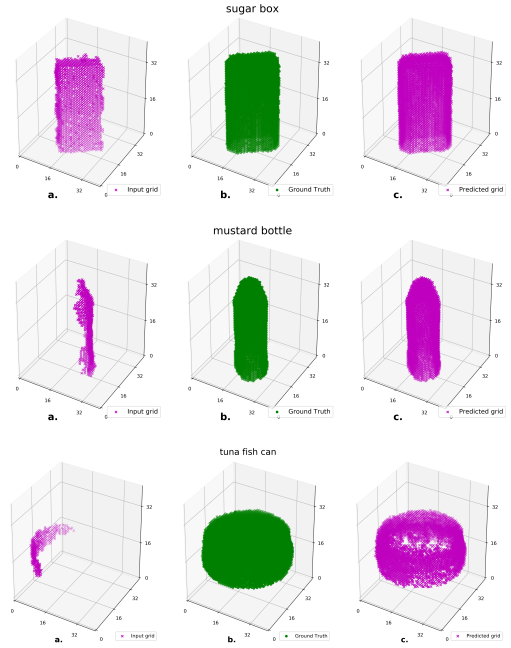

Fig. 9: Prediction example on validation objects from unknown views. a) Original object. b) Ground-Truth. c) Shape prediction.

In this experiment, we have noticed that the estimation of cylindrical, spherical and rectangular objects is better than the estimation of irregular objects due to the lack of symmetry in the last one. In addition, big objects estimation surpass the estimation of small objects because the edges and corners are more detailed on bigger objects. According to the observation viewpoint, the lateral views are resulting in better estimation than the top and bottom views because those partially appear in the lateral views. In contrast, the top view could be identical for long and short objects which leads to predicting different shapes. While in some bottom views, it is difficult to recognize the shape of the object whether it is spherical, cylindrical or rectangular. Finally, as shown in table II, the performance of the model on the

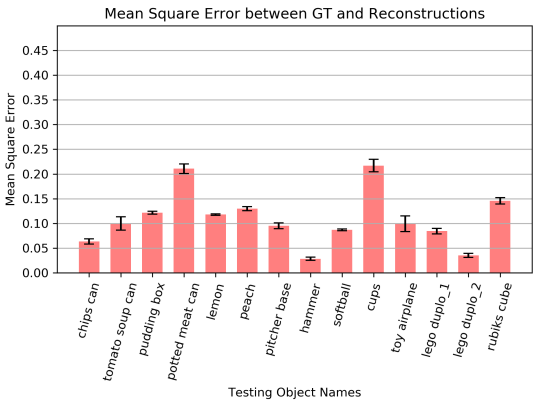

Fig. 10: Average of Squared Mean Error (SME) of the reconstruction from novel objects

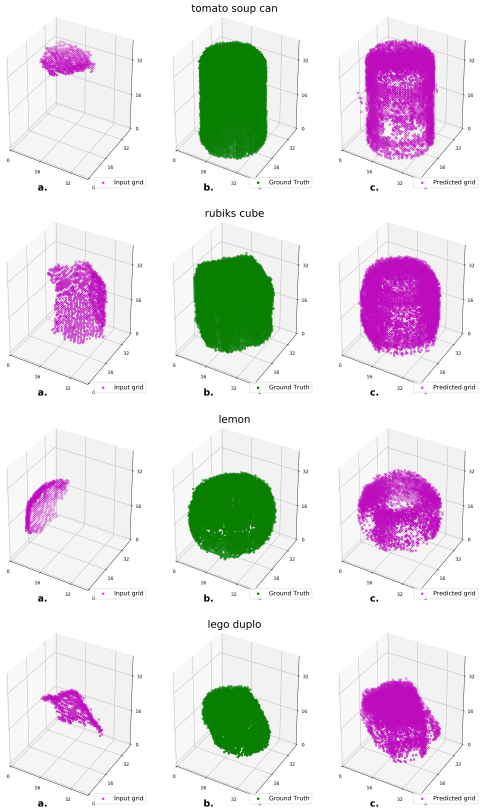

Fig. 11: Prediction example on unknown objects from different views. a) Original object. b) Ground-Truth. c) Shape prediction.

unknown objects is slightly worse than the performance on the unknown views. Mainly, this is due to the fact that new objects contain unseen surface shapes.

VI. CONCLUSIONS

In this work, we proposed a novel 3D CNN model trained to predict the complete shape of an observed object from an arbitrary single view. We trained and tested our model on real depth maps not on synthesized ones.Our proposal has the capability of the generalization from novel views of models that are already seen during training, as well as unseen objects. In addition, it provides a prediction in a few milliseconds. Therefore, this enables it to be employed as an input for grasp planning.

Currently, we are working to add touch data using tactile sensing. In this case, the fusion between contact and visual information will allow us to improve the whole shape estimation of complex objects, especially when the visible part is different from the hidden part. In our future study, other evaluations metrics such as Jaccard similarity (Intersection-Over-Union) and chamber's distance will be used for a deep monitoring on the model performance.

VII. ACKNOWLEDGMENT

To acknowledge the provided support through using the supercomputer of Mésocentre Clermont-Auvergne using one NVIDIA Tesla P40 GPU for our experimentation. https://mesocentre.uca.fr/

TABLE II: Average of MSE of each class per group (smaller is better)

Group	Class	Unknown views	Unknown objects
Shape	Cylindrical	**0.006**	0.131
	Spherical	0.010	**0.114**
	Rectangular	0.010	0.125
	Irregular	0.013	0.192
Size	Big objects	**0.004**	**0.103**
	Small object	0.011	0.114
View	Top view	0.006	0.110
	Lateral view	**0.002**	**0.108**
	Bottom view	0.016	0.109

REFERENCES

[1] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The ycb object and model set: Towards common benchmarks for manipulation research," in *2015 International Conference on Advanced Robotics (ICAR)*, July 2015, pp. 510–517.

[2] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, J. A. Levine, A. Sharf, and C. T. Silva, "State of the Art in Surface Reconstruction from Point Clouds," in *Eurographics 2014 - State of the Art Reports*, S. Lefebvre and M. Spagnuolo, Eds. The Eurographics Association, 2014.

[3] O. Sorkine-Hornung and D. Cohen-Or, "Least-squares meshes," *Proceedings Shape Modeling Applications, 2004.*, pp. 191–199, 2004.

[4] M. M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Trans. Graph.*, vol. 32, pp. 29:1–29:13, 2013.

[5] S. Thrun and B. Wegbreit, "Shape from symmetry," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2, Oct 2005, pp. 1824–1831 Vol. 2.

[6] M. Firman, O. M. Aodha, S. Julier, and G. J. Brostow, "Structured prediction of unobserved voxels from a single depth image," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 5431–5440.

[7] X. Han, H. Laga, and M. Bennamoun, "Image-based 3d object reconstruction: State-of-the-art and trends in the deep learning era," *CoRR*, vol. abs/1906.06543, 2019. [Online]. Available: http://arxiv.org/abs/1906.06543

[8] B. Yang, S. Rosa, A. Markham, N. Trigoni, and H. Wen, "Dense 3d object reconstruction from a single depth view," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2018.

[9] J. Wu, Y. Wang, T. Xue, X. Sun, W. T. Freeman, and J. B. Tenenbaum, "Marrnet: 3d shape reconstruction via 2.5d sketches," vol. abs/1711.03129, 2017. [Online]. Available: http://arxiv.org/abs/1711.03129

[10] A. Dai, C. R. Qi, and M. Nießner, "Shape completion using 3d-encoder-predictor cnns and shape synthesis," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6545–6554, 2016.

[11] D. Shin, C. C. Fowlkes, and D. Hoiem, "Pixels, voxels, and views: A study of shape representations for single view 3d object shape prediction," *CoRR*, vol. abs/1804.06032, 2018. [Online]. Available: http://arxiv.org/abs/1804.06032

[12] J. Varley, C. DeChant, A. Richardson, A. Nair, J. Ruales, and P. K. Allen, "Shape completion enabled robotic grasping," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2442–2447, 2016.

[13] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel, "Bigbird: A large-scale 3d database of object instances," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 509–516, 2014.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 1026–1034.