Shape Completion Enabled Robotic Grasping

Jacob Varley, Chad DeChant, Adam Richardson, Joaquín Ruales, and Peter Allen

Abstract— **This work provides an architecture to enable robotic grasp planning via shape completion. Shape completion is accomplished through the use of a 3D convolutional neural network (CNN). The network is trained on our own new open source dataset of over 440,000 3D exemplars captured from varying viewpoints. At runtime, a 2.5D pointcloud captured from a single point of view is fed into the CNN, which fills in the occluded regions of the scene, allowing grasps to be planned and executed on the completed object. Runtime shape completion is very rapid because most of the computational costs of shape completion are borne during offline training. We explore how the quality of completions vary based on several factors. These include whether or not the object being completed existed in the training data and how many object models were used to train the network. We also look at the ability of the network to generalize to novel objects allowing the system to complete previously unseen objects at runtime. Finally, experimentation is done both in simulation and on actual robotic hardware to explore the relationship between completion quality and the utility of the completed mesh model for grasping.**

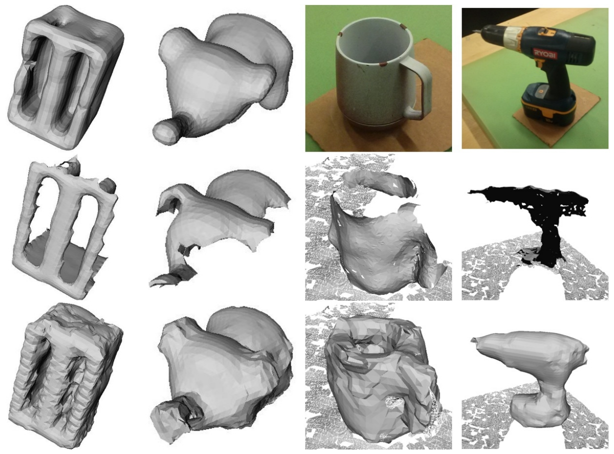


Fig. 1: Ground Truth, Partials, and Completions (T to B). The left two are completions of synthetic depth images of holdout models. The mug and drill are completions of Kinect-captured depth images of physical objects.

I. INTRODUCTION

Grasp planning based on raw sensory data is difficult due to incomplete scene geometry information. In this work 3D CNNs enable stable robotic grasp planning via shape completion. The 3D CNN is trained to do shape completion from a single pointcloud of a target object, essentially filling in the occluded portions of objects. This ability to infer occluded geometries can be applied to a multitude of robotic tasks. It can assist with path planning for both arm motion and robot navigation where an accurate understanding of whether occluded scene regions are occupied or not results in better trajectories. It also allows a traditional grasp planner to generate stable grasps via the completed shape.

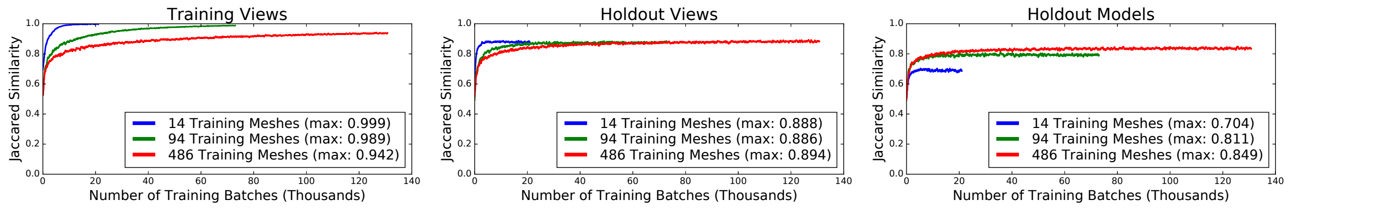
During training, the CNN is shown occupancy grids created from thousands of synthetically rendered depth images of different mesh models. These occupancy grids are captured from a single point of view, and occluded regions are marked as empty. For each training example, the ground truth occupancy grid (the occupancy grid for the entire 3D volume) is also generated for the given mesh. From these pairs of occupancy grids the CNN learns to quickly complete mesh models at runtime using only information from a single point of view. Several example completions are shown in Fig. 1. This setup is beneficial for robotics applications as the majority of the computation time is during offline training, so that at runtime an object's partial-view pointcloud can be run through the CNN and completed in under a tenth of a second on average and then quickly meshed.

At runtime, a pointcloud is captured, segmented, and has regions corresponding to graspable objects extracted. Occupancy grids of these regions are created, and passed separately through the trained CNN. The outputs from the CNN are occupancy grids, where the CNN has labeled all the occluded regions of the input as either occupied or empty for each object. These new occupancy grids are either run through a fast marching cubes algorithm, or further post-processed if they are to be grasped. Whether the object is completed or completed and post-processed results in either 1) fast completions suitable for path planning and scene understanding or 2) detailed meshes suitable for grasp planning, where the higher resolution visible regions are incorporated into the reconstruction. This framework is extensible to crowded scenes with multiple objects as each object is completed individually. It is also applicable to different domains because it can learn from arbitrary training data, and further shows the ability to generalize to unseen views of objects or even entirely novel objects.

The contributions of this work include: 1) A novel CNN architecture for shape completion; 2) A fast mesh completion method, resulting in meshes able to quickly fill the planning scene; 3) A second CUDA enabled completion method that creates detailed meshes suitable for grasp planning by integrating fine details from the observed pointcloud; 4) A open-source dataset of over 440,000 40^3 voxel grid pairs used for

*This work is supported by NSF Grant IIS-1208153. Thanks to the NVIDIA Corporation for the Titan X GPU grant.

Authors are with Columbia University, New York, NY 10027, USA `(jvarley,dechant,allen)@cs.columbia.edu`, `(ajr2190, jar2262)@columbia.edu`



(a) (b) (c)

Fig. 2: Jaccard similarity for three CNNs trained with a variable number of mesh models (blue: 14, green: 94, red: 486). While training, the CNNs were evaluated on a) inputs from training (Training Views), b) novel inputs from meshes they were trained on (Holdout Views) and c) novel inputs from never before seen meshes (Holdout Models).

training. This dataset and the related code are freely available at **http://shapecompletiongrasping.cs.columbia.edu**. The website makes it easy to browse and explore the thousands of completions and grasps related to this work; 5) Results from both simulated and live experiments comparing our method to other approaches and demonstrating its improved performance in grasping tasks.

II. Related Work

Typical approaches to general shape completion to enable robotic grasping [1][2][3] use symmetry and extrusion heuristics, and are reasonable for objects well represented by geometric primitives. Our approach differs in that it learns to complete arbitrary objects based upon training exemplars, rather than requiring objects to conform to heuristics.

A alternative to general shape completion in the robotics community is object recognition and 3D pose detection [4][5][6]. In these approaches objects are recognized from an object database and the pose is estimated. These techniques fill a different use case: the number of encountered objects is small, and known ahead of time. Our approach differs in that it extends to novel objects.

The computer vision community is also interested in shape completion. [7][8], use a deep belief network and Gibbs sampling, and [9], which uses Random Forests. [10] uses an exemplar based approach. Others [11][12] have developed algorithms to learn 3D occupancy grids directly from 2D images. [13] uses a database of models for completion.

It is difficult to apply many of these works directly to robotic manipulation as no large dataset of renderings of handheld objects needed for robotic manipulation tasks existed until now. Also, [11][12] use pure RGB rather than the RGBD images prevalent in robotics, making the problem more difficult as the completed shape must somehow be positioned in the scene and the process does not utilize available depth information. Most create results with resolutions too low for use with current grasp planners which require meshes. Our work creates a new dataset specifically designed for completing objects useful for manipulation tasks using the 2.5-D range sensors prevalent in robotics, and provides a technique to integrate the high resolution observed view of the object with our relatively high resolution CNN output, creating a completion suitable for grasp planning.

Our work differs from [14][15], both of which require a complete mesh to query a model database and retrieve grasps used on similar objects. These approaches could be used in tandem with our framework where the completed model would act as the query mesh. While grasps can be planned using partial meshes where the object is not completed (see [16]), they still have their limitations and issues. Shape completion can be used to alleviate this problem.

This framework uses the YCB[17] and Grasp Database[18] mesh model datasets. We chose these as many robotics labs all over the world have physical copies of the YCB objects, and the Grasp Database contains objects specific to robotic manipulation. We augmented 18 of the higher quality YCB meshes with the 590 Grasp Database meshes.

TABLE I: CNN Architecture

Layer	Operation	Resulting Shape
Input	-	40^3 x 1 channel
1	4^3 conv + Relu	37^3 x 64 channels
2	4^3 conv + Relu + 2^3 max pool	17^3 x 64 channels
3	4^3 conv + Relu + 2^3 max pool	7^3 x 64 channels
4	Flatten	21952
5	Dense + Relu	5000
6	Dense + Sigmoid	64000
7	Reshape	40^3 x 1 channel

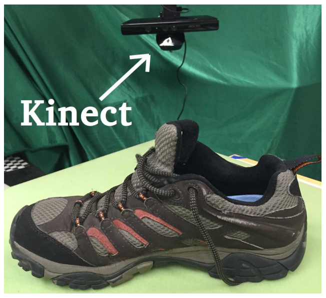
III. Training

A. Data Generation

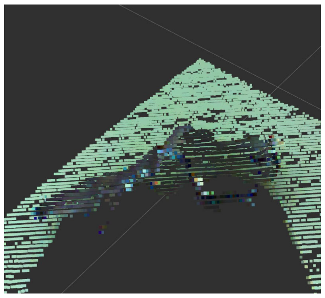
To train a network to reconstruct a diverse range of objects, meshes were collected from the YCB and Grasp Database. The models were run through binvox[19] generating 256^3 occupancy grids. Then in Gazebo[20] 726 depth images were generated for each object subject to different uniformly sampled rotations. The depth images are used to create occupancy grids for the portions of the mesh visible to the simulated camera, and then all the occupancy grids generated by binvox are transformed to correctly overlay the depth image occupancy grids. Both sets of occupancy grids are then down-sampled to 40^3 to create a large number of training examples. The input set (X) contains occupancy grids that are filled only with the regions of the object visible to the camera, and the output set (Y) contains the ground truth occupancy grids for the space occupied by the entire model.

B. Model Architecture and Training

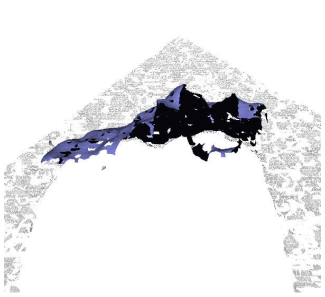
The CNN architecture in Table I was trained using a cross-entropy error cost function, and optimizer Adam[21].



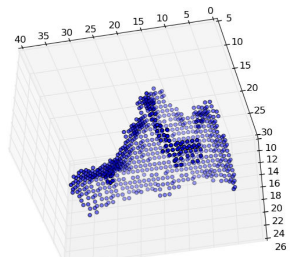
(a) Image of Occluded Side



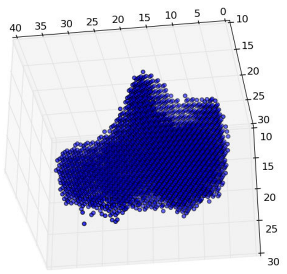
(b) Point Cloud



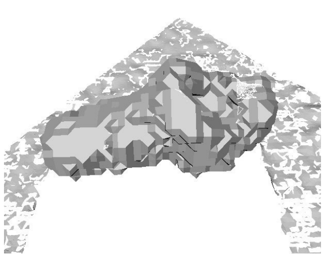
(c) Segmented and Meshed



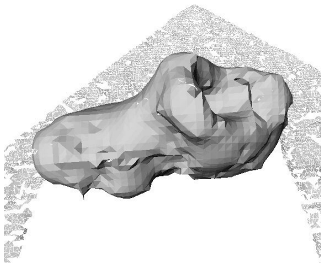
(d) CNN Input



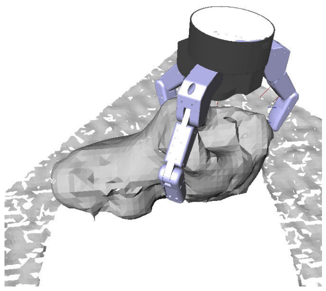
(e) CNN Output



(f) Fast Mesh



(g) Detailed Mesh



(h) Grasp Planning

Fig. 3: Runtime Pipeline with novel object CNN was not trained on: Images are not shown from the angle captured in order to visualize the occluded regions. (a): Scene with an object to be grasped. (b): Scene pointcloud. (c): The pointcloud is segmented and meshed. (d): A partial mesh is selected, voxelized and passed into the 3D shape completion CNN. (e): CNN output. (f): The resulting occupancy grid can be run through a marching cubes algorithm to obtain a mesh quickly. (g): Or, for better results, the output of the CNN can be combined with the observed pointcloud and preprocessed for smoothness before meshing. (h): Grasps are planned on the smoothed completed mesh.

Weights were initialized following [22][23], and trained with batch size 32 on an NVIDIA Titan X GPU.

We used the Jaccard similarity (intersection over union) to compare generated voxel occupancy grids and the ground truth. During training, this measure is computed for input meshes in the training data (**Training Views**), novel views of meshes within the training data (**Holdout Views**), and meshes of objects not in the training data (**Holdout Models**).

C. Training Results

To explore how the quality of the reconstruction changes as the number of models in the training set is adjusted, we trained three networks with identical architectures using variable numbers of mesh models. One was trained with partial views from 14 YCB models, another with 94 models (14 YCB + 80 Grasp Database), and the third with 486 models (14 YCB models + 472 Grasp Database). Each network was allowed to train until learning plateaued. The remaining 4 YCB and 118 Grasp Dataset models were kept as a holdout set. Fig. 2 shows how the Jaccard similarity measures vary as the networks' train. We note that the networks trained with fewer models perform better shape completion when they are tested on views of objects they have seen during training than networks trained on a larger number of models. This suggests that the network is able to completely learn the training data for the smaller number of models but struggles to do so when trained on larger numbers. Conversely, the models trained on a larger number of objects perform better than those trained on a smaller number when asked to complete novel objects. Because, as we have seen, the networks trained on larger numbers of objects are unable to learn all of the models seen in training, they may be forced to learn a more general completion strategy that will work for a wider variety of objects, allowing them to better generalize to objects not seen in training.

Fig. 2(a) shows the performance of the three CNNs on training views. In this case, the fewer the mesh models used during training, the better the completion results. Fig. 2(b) shows how the CNNs performed on novel views of the mesh objects used during training. Here the CNNs all did approximately the same. Fig. 2(c) shows the completion quality of the CNNs on objects they have not seen before. In this case, as the number of mesh models used during training increases, performance improves as the system has learned to generalize to a wider variety of inputs.

IV. Runtime

At runtime the pointcloud for the target object is acquired from a 3D sensor, scaled, voxelized and passed through the CNN. The output of the CNN, a completed voxel grid of the object, goes through a post processing algorithm that returns a completed mesh. Finally, a grasp can be planned and executed based on the completed mesh model. Fig. 3 demonstrates the runtime pipeline on a novel object.

1) Acquire Target Pointcloud: A pointcloud is captured using a Microsoft Kinect, then segmented using PCL's[24] implementation of euclidean cluster extraction. A segment corresponding to the object to be completed is selected.
2) Complete via CNN: The selected pointcloud is used to create an occupancy grid with resolution 40^3. This occupancy grid is input to the CNN whose output is an equivalently sized occupancy grid for the completed shape. To fit the pointcloud to the 40^3 grid, it is scaled down so the bounding box of the pointcloud fits in a 32^3 voxel cube, and then centered in the 40^3 grid such that the center of the bounding box is at (20, 20, 18). Finally all voxels occupied by points from this scaled and transformed pointcloud are marked as such. Placing the pointcloud slightly off-center in the z dimension leaves more space in the back half of the grid for the network to fill.
3a) Create Fast Mesh: If the object being completed is not going to be grasped, then the voxel grid output by the CNN is run through marching cubes, and the resulting mesh is added to the planning scene, filling in occlusions.
3b) Create Detailed Mesh: Alternatively, if this object is going to be grasped, then post-processing occurs. The purpose of this post-processing is to integrate the points from the visible portion of the object with the output of the CNN. This partial view is of much higher density than the 40^3 grid and captures significantly finer detail for the visible surface. This merge is made difficult by the large disparity in point densities between the captured cloud and 40^3 CNN output which can lead to holes and discontinuities if the points are naively merged.

Algorithm 1 Shape Completion

```
1:  procedure MESH(cnn_out, observed_pc)
2:      //cnn_out: $40^3$ voxel output from CNN
3:      //observed_pc: captured pointcloud of object
4:      if FAST then return mCubes(cnn_out)
5:      d_ratio ← densityRatio(observed_pc, cnn_out)
6:      upsampled_cnn ← upsample(cnn_out, d_ratio)
7:      vox ← merge(upsampled_cnn, observed_pc)
8:      vox_no_gap ← fillGaps(vox)
9:      vox_weighted ← CUDA_QP(vox_no_gap)
10:     mesh ← mCubes(vox_weighted)
11:     return mesh
```

Alg. 1 shows how we integrated the dense partial view with our 40^3 voxel grid via the following steps. (Alg.1:L5) In order to merge with the partial view, the output of the CNN is converted to a point cloud and its density is compared to the density of the partial view point cloud. The densities are computed by randomly sampling $\frac{1}{10}$ of the points and averaging the distances to their nearest neighbors. (Alg.1:L6) The CNN output is up-sampled by d_ratio to match the density of the partial view. For each new voxel, the L1 distance to 8 original closest voxels are computed and the 8 distances are summed, weighted by 1 if the original voxel is occupied and -1 otherwise. The new voxel is occupied if its weighted sum is nonnegative. This has the effect of creating piecewise linear separating surfaces similar marching cubes and mitigates up-sampling artifacts. (Alg.1:L7) The upsampled output from the CNN is then merged with the point cloud of the partial view and the combined cloud is voxelized at the new higher resolution of $(40 * d\_ratio)^3$. For most objects d_ratio is 2 or 3, depending on the physical size of the object, resulting in a voxel grid of either 80^3 or 120^3. (Alg.1:L8) Any gaps in the voxel grid between the upsampled CNN output and the partial view cloud are filled. This is done by finding the first occupied voxel in every z-stack. If the distance to the next occupied voxel is less than $d\_ratio + 1$ the intermediate voxels are filled. (Alg.1:L9) The voxel grid is smoothed using our CUDA implementation of the convex quadratic optimization from [25]. (Alg.1:L10) The weighted voxel grid is run through marching cubes.
4) Grasp completed mesh: A grasp is planned on the reconstructed mesh in GraspIt![26], and the reachability of the planned grasps are checked in MoveIt![27], and the highest quality reachable grasp is executed.

View Type	Partial	Mirror	Ours
Training Views	0.1182	0.2325	**0.7771**
Holdout Views	0.1307	0.2393	**0.7486**
Holdout Models	0.0931	0.1921	**0.6496**

TABLE II: Jaccard Similarity Results (Larger is better). This measures the intersection over union of two voxelized meshes as described in Section V-A.

View Type	Partial	Mirror	Ours
Training Views	11.4	7.5	**3.6**
Holdout Views	12.3	8.2	**4.0**
Holdout Models	13.6	10.7	**5.9**

TABLE III: Hausdorff Distance Results (Smaller is better). This measures the mean distance in millimeters from points on one mesh to another as described in Section V-A.

V. Experimental Results

We created a test dataset by randomly sampling 50 training views (**Training Views**), 50 holdout views (**Holdout Views**), and 50 views of holdout models (**Holdout Models**). The Training Views and Holdout Views were sampled from the 14 YCB training objects. The Holdout Models were sampled from holdout YCB and Grasp Dataset objects. We compared the accuracy of the completion methods using: Jaccard similarity, Hausdorff distance, and geodesic divergence.

A. General Completion Results

We first compared several general completion methods: putting the partial view through marching cubes and Meshlab's Laplacian smoothing (**Partial**), mirroring completion[1] (**Mirror**), our method (**Ours**). Our CNN was trained on the 484 objects from the YCB + Grasp Dataset and the weights come from the point of peak performance on holdout models (red line in Fig. 2(c)).

The Jaccard similarity was used to compare the final resulting meshes from several completion strategies (Table II).

View Type	Partial	Mirror	Ours
Training Views	0.3770	0.2905	**0.0867**
Holdout Views	0.4944	0.3366	**0.0934**
Holdout Models	0.3407	0.2801	**0.1412**

TABLE IV: Geodesic Divergence Results (Smaller is better). This measures the Jenson-Shannon probabilistic divergence between two meshes as described in Section V-A.

The completed meshes were voxelized at 80^3, and compared with the ground truth. Our proposed method results in higher similarity to the ground truth meshes than the partial and mirroring approaches for all tested views.

The Hausdorff distance was computed with Meshlab's[28] implementation in both directions. Table III shows mean values of the symmetric Hausdorff distance per completion method. The CNN completions are significantly closer to the ground truth than the partial and mirrored completions.

The completions are also compared using a measure of geodesic divergence[29]. A probability density function is computed for each mesh. The probability density functions for each completion are compared with the ground truth mesh using the Jenson-Shannon divergence. Table IV shows the mean of the divergences for each completion method. Here, our method outperforms all other completion methods.

Across all metrics, our method results in more accurate completions than the other general completion approaches.

B. Comparison to Database Driven Methods

We evaluated a RANSAC-based approach[5] on the Training Views of the YCB dataset using the same metrics. This corresponds to a highly constrained environment containing only a very small number of objects which are known ahead of time. It is not possible to load 484 objects into the RANSAC framework, so a direct comparison to our method involving the large number of objects we train on is not possible. In fact, the inability of RANSAC-based methods to scale to large databases of objects is one of the motivations of our work. However, we compared our method to a very small RANSAC using only 14 objects, and our method performs comparably to the RANSAC approach even on objects in its database, while having the additional abilities to train on far more objects and generalize to novel objects: Jaccard (Ours: 0.771, RANSAC: **0.8566**), Hausdorff (Ours: 3.6, RANSAC: **3.1**), geodesic (Ours: **0.0867**, RANSAC: 0.1245). Our approach significantly outperforms the RANSAC approach when encountering an object that neither method has seen before (Holdout Models): Jaccard (Ours: **0.6496**, RANSAC: 0.4063), Hausdorff (Ours: **5.9**, RANSAC: 20.4), geodesic (Ours: **0.1412**, RANSAC: 0.4305). The RANSAC based approach's performance on the Holdout Models is also worse than that of the mirrored or partial completion methods on both the geodesic and Hausdorff metrics.

C. Simulation Based Grasp Comparison

To evaluate our framework's ability to enable grasp planning, the system was tested in simulation using the same

View	Error	Completion Type			
		Partial	**Mirror**	**Ours**	**RANSAC**
Training View	**Joint (°)**	6.09°	4.20°	**1.75°**	1.83°
	Pose (mm)	16.0	11.5	**4.3**	7.3
Holdout View	**Joint (°)**	6.27°	4.05°	1.80°	**1.69°**
	Pose (mm)	20.8	15.6	**6.7**	7.4
Holdout Model	**Joint (°)**	7.59°	5.82°	**4.56°**	6.86°
	Pose (mm)	18.3	15.0	**13.2**	29.25

TABLE V: Simulation results comparing planned to realized grasps by Joint Error (mean differences per joint) and Pose Error (mean difference in pose). Smaller is better.

Completion Method	Grasp Success Rate (%)	Joint Error (degrees)	Completion Time (s)
Partial	71.43	9.156°	**0.545**
Mirror	73.33	8.067°	1.883
Ours	**93.33**	**7.276°**	2.426

TABLE VI: Grasp Success Rate shows percentage of successful grasps. Joint Error shows the mean difference in degrees between the planned and executed grasp joint values. Completion Time shows how long the method required.

completions from Sec V-A. GraspIt! was used to plan over 24,000 grasps on all of the completions of the objects. In order to simulate a real-world grasp execution, the completion was removed from GraspIt! and the ground truth object was inserted in its place. Then the hand was placed 20cm backed off from the ground truth object along the approach direction of the grasp. The spread angle of the fingers was set, and the hand was moved along the approach direction of the planned grasp either until contact was made or the grasp pose was reached. At this point, the fingers closed to the planned joint values, and continued until either contact occurred or joint limits were reached. Visualizations of the simulation results for the entire YCB and Grasp Datasets are available at **http://shapecompletiongrasping.cs.columbia.edu**

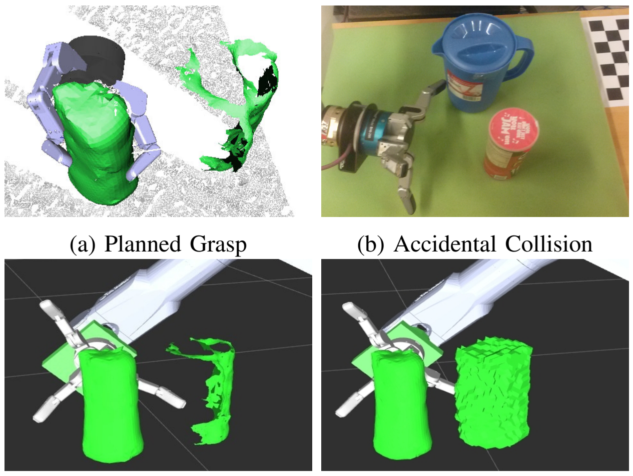
Table V shows the differences between the planned and realized joint states as well as the difference in pose of the base of the end effector between the planned and realized grasps. Using our method caused the end effector to end up closer to its intended location in terms of both joint space and the palm's cartesian position.

D. Performance on Real Hardware

The system was used in an end-to-end manner with a Barrett Hand and StaubliTX60 Arm to execute grasps planned via the aforementioned completion methods for 15 YCB objects. For each object, we ran the arm once per completion method. Results in Table VI show our method enabled a 20% improvement over the general shape completion methods in terms of grasp success rate, and resulted in executed grasps closer to the planned grasps shown by the lower joint error.

E. Crowded Scene Completion

Scenes often contain objects that are not to be manipulated and only require completion in order to be avoided. In this case, our CNN output can be run directly through


(a) Planned Grasp (b) Accidental Collision

(c) Unfilled Planning Scene (d) Filled Planning Scene

Fig. 4: The arm fails to execute the planned grasp (a), resulting in collision (b). The collision with the non-target object occurred due to a poor planning scene (c). Our fast mesh method can fill the planning scene so this configuration is marked invalid (d).

marching cubes without post-processing to quickly create a mesh. Fig. 4 highlights the benefit of reasoning about non target objects. Our system has the ability to quickly fill occluded regions of the scene, and selectively spend more time generating detailed completions on specific objects to be manipulated. Average scene completion times from 15 runs of crowded scenes are Scene Segmentation (0.119s), Target Object Completion (2.136s), and Non Target Object Completion (0.142s).

VI. Conclusion

This work utilizes a CNN to complete and mesh an object observed from a single point of view, and then plan grasps on the completed object. The completion system is fast, with completions available in a matter of milliseconds, and post processed completions suitable for grasp planning available in several seconds. The dataset and code are open source and available for other researchers to use. It has also been demonstrated that our completions are better than more naive approaches in terms of a variety of metrics including those specific to grasp planning. In addition, grasps planned on completions generated using our method are more often successful and result in executed grasps closer to the intended hand configuration than grasps planned on completions from the other methods.

References

[1] J. Bohg, M. Johnson-Roberson, B. León, J. Felip, X. Gratal, N. Bergström, D. Kragic, and A. Morales, "Mind the gap-robotic grasping under incomplete observation," in *ICRA*. IEEE, 2011, pp. 686–693.

[2] A. H. Quispe, B. Milville, M. A. Gutiérrez, C. Erdogan, M. Stilman, H. Christensen, and H. B. Amor, "Exploiting symmetries and extrusions for grasping household objects," in *ICRA*, 2015, pp. 3702–3708.

[3] D. Schiebener, A. Schmidt, N. Vahrenkamp, and T. Asfour, "Heuristic 3d object shape completion based on symmetry and scene context," in *IROS*. IEEE, 2016, pp. 74–81.

[4] C. Rennie, R. Shome, K. E. Bekris, and A. F. De Souza, "A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1179–1185, 2016.

[5] C. Papazov and D. Burschka, "An efficient ransac for 3d object recognition in noisy and occluded scenes," in *Asian Conference on Computer Vision*. Springer, 2010, pp. 135–148.

[6] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *ICCV), 2011*. IEEE, 2011, pp. 858–865.

[7] Z. Wu, S. Song, A. Khosla, X. Tang, and J. Xiao, "3D shapenets for 2.5D object recognition and next-best-view prediction," *arXiv preprint arXiv:1406.5670*, 2014.

[8] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *CVPR*, 2015, pp. 1912–1920.

[9] M. Firman, O. Mac Aodha, S. Julier, and G. J. Brostow, "Structured prediction of unobserved voxels from a single depth image," in *CVPR*, 2016, pp. 5431–5440.

[10] J. Rock, T. Gupta, J. Thorsen, J. Gwak, D. Shin, and D. Hoiem, "Completing 3d object shape from one depth image," in *CVPR*, 2015, pp. 2484–2493.

[11] S. Tulsiani, A. Kar, J. Carreira, and J. Malik, "Learning category-specific deformable 3d models for object reconstruction," *IEEE transactions on pattern analysis and machine intelligence*, 2016.

[12] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction," in *ECCV*. Springer, 2016, pp. 628–644.

[13] Y. Li, A. Dai, L. Guibas, and M. Nießner, "Database-assisted object retrieval for real-time 3d reconstruction," in *Computer Graphics Forum*, vol. 34, no. 2. Wiley Online Library, 2015, pp. 435–446.

[14] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg, "Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards," in *ICRA, 2016*. IEEE, 2016, pp. 1957–1964.

[15] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen, "The columbia grasp database," in *ICRA*. IEEE, 2009, pp. 1710–1716.

[16] J. Varley, J. Weisz, J. Weiss, and P. Allen, "Generating multi-fingered robotic grasps via deep learning," in *IROS*, 2015, pp. 4415–4420.

[17] 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 *Advanced Robotics (ICAR), 2015 International Conference on*. IEEE, 2015, pp. 510–517.

[18] D. Kappler, J. Bohg, and S. Schaal, "Leveraging big data for grasp planning," in *ICRA*. IEEE, 2015, pp. 4304–4311.

[19] P. Min, "Binvox, a 3d mesh voxelizer," 2004.

[20] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *IROS*, vol. 3. IEEE, 2004, pp. 2149–2154.

[21] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *arXiv preprint arXiv:1502.01852*, 2015.

[23] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the 13th AISTATS*, 2010, pp. 249–256.

[24] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *ICRA*, Shanghai, China, May 9-13 2011.

[25] V. Lempitsky, "Surface extraction from binary volumes with higher-order smoothness," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 1197–1204.

[26] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *IEEE R&A Magazine*, vol. 11, no. 4, pp. 110–122, 2004.

[27] I. A. Sucan and S. Chitta, "Moveit!" *http://moveit.ros.org*, 2013.

[28] P. Cignoni, M. Corsini, and G. Ranzuglia, "Meshlab: an open-source 3d mesh processing system," *Ercim news*, vol. 73, pp. 45–46, 2008.

[29] A. B. Hamza and H. Krim, "Geodesic object representation and recognition," in *International conference on discrete geometry for computer imagery*. Springer, 2003, pp. 378–387.