

Yale-CMU-Berkeley Dataset for Robotic Manipulation Research

Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha Srinivasa, Pieter Abbeel, Aaron M. Dollar

Abstract— In this paper, we present an image and model dataset of the real-life objects from the Yale-CMU-Berkeley (YCB) Object Set, which is specifically designed for benchmarking in manipulation research. For each object, the dataset presents 600 high-resolution RGB images, 600 RGB-D images and five sets of textured 3D geometric models. Segmentation masks and calibration information for each image are also provided. These data are acquired using the BigBIRD Object Scanning Rig and Google Scanners. Together with the dataset, Python scripts and a Robot Operating System (ROS) node are provided to download the data, generate point clouds and create Unified Robot Description Files (URDFs). The dataset is also supported by our website, www.ycbbenchmarks.org, which serves as a portal for publishing and discussing test results along with proposing task protocols and benchmarks.

Index Terms—Benchmarking, manipulation, grasping, simulation

1 INTRODUCTION

IN this paper we present an image and model dataset of real-life objects for manipulation research. The dataset is available at <http://ycb-benchmarks.s3-website-us-east-1.amazonaws.com/>. Comparing to other object datasets in literature (including ones widely utilized by the robotics community (Goldfeder *et al.*, 2009; Kasper, Xue and Dillmann, 2012; Singh *et al.*, 2014); a comprehensive overview is given in (Berk Calli, Walsman, *et al.*, 2015)), our dataset has four major advantages. First, the objects are a part of the Yale-CMU-Berkeley (YCB) Object Set (Berk Calli, Walsman, *et al.*, 2015; Calli, Singh, *et al.*, 2015), which makes the physical objects available to any research group around the world upon request via our project website (YCB-Benchmarks, 2016b). Therefore, our dataset can be utilized both in simulations and in real-life model-based manipulation experiments. Second, the objects in the YCB set are specifically chosen for benchmarking in grasping and manipulation research, being tailored for designing many realistic and interesting manipulation scenarios with its shape and texture variety, as relationship to a range of tasks. Third, the quality of the data provided by our dataset is significantly greater than previous works; we supply high quality RGB images, RGB-D images and five sets of textured geometric models acquired by two state of the art systems (one at UC Berkeley and one at Google). Finally, our dataset is supported with a webportal (YCB-Benchmarks, 2016b),

which is designed as a hub to present and discuss results and to propose manipulation tasks and benchmarks.

The objects are scanned with two systems: the BigBIRD Object Scanning Rig (Singh *et al.*, 2014) (Section 2.1, Fig. 1, 2) and a Google scanner (Section 2.2, Fig. 3). The BigBIRD Object Scanning Rig provides 600 RGB and 600 RGB-D images for each object along with segmentation masks and calibration information for each image. Two kinds of textured mesh models are generated using these data by utilizing Poisson reconstruction (Kazhdan, Bolitho and Hoppe, 2006) and Truncated Signed Distance Function (TSDF) (Curless and Levoy, 1996) techniques. With the Google scanner, the objects are scanned in three resolution levels (16k, 64k, 512k mesh vertices). We believe that supplying these model sets with different quality and acquisition techniques will be useful for the community to determine best practices in manipulation simulations and also to assess the effect of model properties in the model-based manipulation planning techniques.












The scanned objects are listed in Table 1. We provide data for all the objects in YCB Set except for the ones that do not have a determinate shape (i.e. table cloth, t-shirt, rope, plastic chain), and the ones that are too small for the scanning systems (i.e. nails, washers).




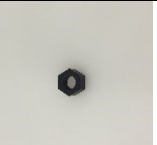







The dataset is hosted by Amazon Web Services Public Dataset Program (Amazon, 2016b). The program provides a tool called Amazon Simple Storage Service (Amazon S3) (Amazon, 2016a) to access the hosted data. Alternatively, the files can be downloaded via the links on our website (YCB-Benchmarks, 2016b). Along with these options, we also provide scripts for downloading and processing the data via simple parameter settings. Additionally, a Robot Operating System (ROS; (Quigley *et al.*, 2009)) node is available at (YCB-Benchmarks, 2016a) to manage the data and generate Unified Robot Description Files (URDFs) of the mesh models for easy integration to software platforms such as Gazebo (Koenig and Howard, 2004) and MoveIt (Chitta, Sucan and Cousins, 2012).












- Berk Calli and Aaron M. Dollar are with the Mechanical Engineering and Material Science Department, Yale University, 9 Hillhouse Avenue, New Haven, CT, 06511. E-mail: {berk.calli;aaron.dollar}@yale.edu.
- Arjun Singh and Pieter Abbeel are with the Electrical Engineering and Computer Sciences Department, University of California, Berkeley, 746 Sutardja Dai Hall, Berkeley, CA, 94720-1758. E-mail: {arjun-singh;pabbeel}@berkeley.edu.
- Aaron Walsman and Siddhartha Srinivasa are with the Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213. E-mail: awalsman@andrew.cmu.edu; siddh@cs.cmu.edu.
- James Bruce and Kurt Konolige are with Google, 1600 Amphitheatre Pkwy, Mountain View, USA 94043. E-mail: {jimbruce;konolige}@google.com



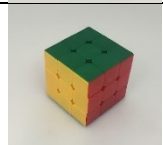
Table 1: The objects scanned in YCB object and model set. Note that the object IDs are kept consistent with (Berk Calli, Walsman, *et al.*, 2015); there are jumps since some objects are skipped as they don't have a determinate shape or too small for the scanners to acquire meaningful data. Some objects have multiple parts scanned as indicated by the letters next to their ID numbers.

ID	Picture	Object name	Mass (g)	ID	Picture	Object name	Mass (g)	ID	Picture	Object name	Mass (g)
1		Chips Can	205	11		Banana	66	21		Bleach Cleanser	1,131
2		Master Chef Can	414	12		Strawberry	18	22		Windex Bottle	1,022
3		Cracker Box	411	13		Apple	68	23		Wine-glass	133
4		Sugar Box	514	14		Lemon	29	24		Bowl	147
5		Tomato Soup Can	349	15		Peach	33	25		Mug	118
6		Mustard Bottle	603	16		Pear	49	26		Spong	6.2
7		Tuna Fish Can	171	17		Orange	47	27		Skillet	950
8		Pudding Box	187	18		Plump	25	28		Skillet Lid	652
9		Gelatin Box	97	19		Pitcher Base	178	29		Plate	279
10		Potted Meat Can	370	20		Pitcher Lid	66	30		Fork	34

31		Spoon	30
32		Knife	31
33		Spatula	51.5
35		Power Drill	895
36		Wood Block	729
37		Scissors	82
38		Padlock	304
39		Keys	10.1
40		Large Marker	15.8
41		Small Marker	8.2
42		Adjustable Wrench	252

43		Philips Screwdriver	97
44		Flat Screwdriver	98.4
46		Plastic Bolt	3.6
47		Plastic Nut	1
48		Hammer	665
49		Small Clamp	19.2
50		Medium Clamp	59
51		Large Clamp	125
52		Extra-large Clamp	202
53		Mini Soccer Ball	123
54		Softball	191

55		Baseball	148
56		Tennis Ball	3.6
57		Racquetball	1
58		Golf Ball	665
61		Foam Brick	59
62		Dice	125
63 (a-e)		Marbles	202
65 (a-k)		Cups	123
70 (a-b)		Colored Wood Bloc	729
71 (a-b)		Nine-hole Peg Test	82
72 (a-k)		Toy Airplane	304

73 (a-m)		Lego	10.1
76		Timer	8.2
77		Rubik's Cube	252

This paper provides a complete explanation of the dataset, its acquisition methods, its usage and the supplementary programs. Nevertheless, for detailed information about the YCB benchmarking effort in general, we refer the reader to (B. Calli, Walsman, *et al.*, 2015), which focuses on the criteria for choosing the objects and their utilization for benchmarking in robotics.

2 SYSTEM DESCRIPTION

This section presents specifications of the BigBIRD Object Scanning Rig and the Google Scanner used in data acquisition.

2.1 BigBIRD Object Scanning Rig:

The rig has five 12.2 megapixel Canon Rebel T3 RGB cameras and five PrimeSense Carmine 1.08 RGB-D sensors. The Canon Rebel T3s have APS-C sensors (22.0mm x 14.7mm), a pixel size of 5.2 μ m, and are equipped with EF-S 18-55mm f/3.5-5.6 IS lenses. Before imaging each object, the cameras are focused using autofocus. However, while imaging the objects, the autofocus is turned off. This means that the focus can vary between objects but is the same for all images for a single object.

Each RGB camera is paired with an RGB-D sensor as shown in Fig. 1. These pairs are arranged in a quarter-circular arc focusing to a motorized turntable placed at a photobench as depicted in Fig. 2. To obtain calibrated data, a chessboard is placed on the turntable in such a way that it is always fully visible in at least one of the cameras. For consistent illumination, light sources are arranged at the bottom wall, the back wall, the front corners and the back corners of the photobench.

To calibrate the cameras, we require an external infrared light and a calibration chessboard. We take pictures of the chessboard with the high-resolution RGB camera and the RGB-D sensor's infrared camera and RGB cameras, as well as a depth map. We then detect the chessboard corners in all of the images. Note that we turn off the infrared emitter before collecting infrared images, and turn it back on before collecting depth maps.

After collecting data, we first initialize the intrinsic matrices and transformations for all fifteen cameras (five Canon T3s, five Carmines with an RGB camera and IR



Fig. 1: A Canon Rebel T3 RGB camera and PrimeSense Carmine 1.08 RGB-D sensor pair mounted together.



Fig. 2: Bigbird scanning rig and viewpoints of the 5 cameras.

camera each) using OpenCV’s camera calibration routines. We also initialize the relative transformations between cameras using OpenCV’s solvePnP. We then construct an optimization problem to jointly optimize the intrinsic parameters and extrinsic parameters for all the sensors. The details of the optimization are given in (Singh *et al.*, 2014).

Each object was placed on a computer-controlled turntable, which was rotated by three degrees at a time, yielding 120 turntable orientations. Together, this yields 600 RGB images and 600 RGB-D images. The process is completely automated, and the total collection time for each object is under five minutes.

The collected images are used in surface reconstruction procedure to generate watertight meshes. Two sets of textured mesh models are obtained using Poisson reconstruction (Kazhdan, Bolitho and Hoppe, 2006) and Truncated Signed Distance Functions (TSDF) (Bylow *et al.*, 2013) methods. While Poisson method provides watertight meshes, TSDF models are not guaranteed to be watertight.

Together with the images and models, we also provide calibration information and segmentation masks for each image. The segmentation masks are obtained by projecting the Poisson models onto the RGB images using the calibration data.

Note that both Poisson and TSDF methods fail on objects with missing depth data due to the transparent or reflective regions: For objects 22, 30, 31, 32, 38, 39, 42, 43 and 44 the mesh models are partially distorted and for the objects 23 and 28 no meaningful model could be generated with the adopted methods. The system also fails to obtain models for very thin or small objects such as 46, 47 and 63-b-f. For objects with missing models, we still provide RGB and RGB-D images, which, together with the calibration information, can be used to implement other methods of model reconstruction.

2.2 Google Scanner

The Google research scanner consists of a mostly light-sealed enclosure, three “scanheads” and a motorized turntable (Fig. 3). Each scanhead is a custom structured light (Scharstein and Szeliski, 2003) capture unit, consisting of a consumer DLP projector, two monochrome cameras in a stereo pair and a color camera to capture fine detail texture/color information (in total 3 cameras per scanhead). The consumer projector is an ACER P7500 with 1920x1080 resolution and 4000 lumens. The monochrome cameras are Point Grey Grasshopper3 with Sony IMX174 CMOS sensors and 1920x1200 resolution. The lenses on the monochrome cameras are Fujinon HF12.5SA-1 with 12.5mm focal length. The color camera is a Canon 5dMk3 with 5760 x 3840 pixels resolution with a Canon EF 50mm f 1:1.4 lens. Using a consumer projector allows us to select a model with very high brightness and contrast, which helps when scanning dark or shiny items, but complicates synchronization, which now must be done in software.

Processing is split into an online and an offline stage. In the online stage, structured light patterns are decoded

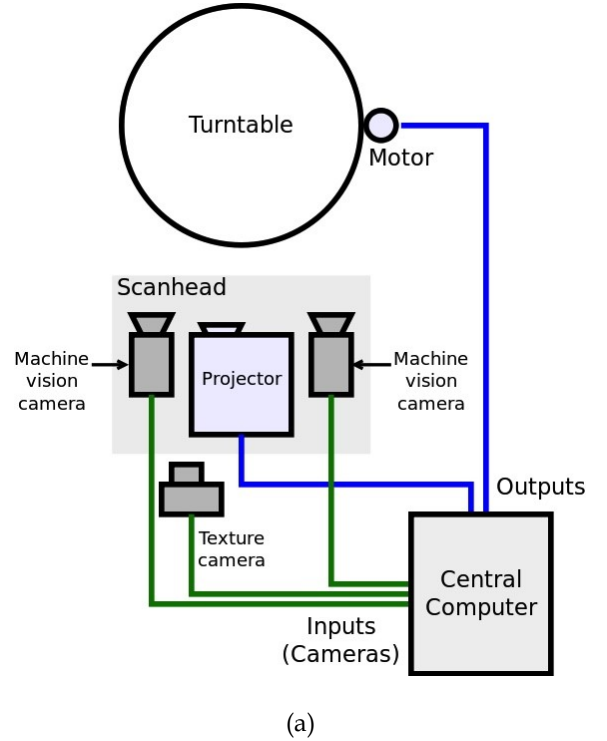


Fig. 3: Google Scanner, (a) the overall scheme, (b) projector coupled with two monochrome cameras and a DSLR camera

and triangulated into a depth map. An image from the DSLR is saved with each view. The online stage is followed by offline post-processing, where the individual depth maps are aligned, merged, and textured. Scanning and post-processing are a fully automated processes. The scanner is calibrated using CMVision (Bruce, Balch and Veloso, 2000) for pattern detection and Ceres-Solver (Sameer and Mierle, 2012) is used to optimize camera parameters for all 9 cameras in parallel. The objects were scanned using 8 turntable stops for a total of 24 views. For each object, three mesh models are generated with 16k, 64k, 514k mesh vertices.

Again due to object properties, the models generated by this scanner are cannot be generated for objects 23, 39, 46, 47 and 63-c-f, and a partially distorted model is generated for object 22.

Table 2: Parameters to set in the ycb_downloader.py Python script.

Parameter name	Type	•Values
objects_to_download	String array	Object full names with id and underscores e.g. "001_chips_can", "063-a_marbles"
files_to_download	String array	Any combination of "berkeley_rgb-highres", "berkeley_rgbd", "berkeley_processed", "google_16k", "google_64k", "google_512k"
extract	Bool	'True' if the downloaded compressed file is to be extracted. 'False' if otherwise.

Table 3: Parameters to set in the ycb_generate_point_cloud.py Python script.

Parameter name	Type	Values
target_object	String	Object full names with id and underscores e.g. "001_chips_can"
viewpoint_camera	String	Camera from which the viewpoint is generated. Either of "NP1", "NP2", "NP3", "NP4" or "NP5"
viewpoint_angle	String	An integer number between 0 and 357 and a multiple of 3. E.g. "0", "3", "9" ... "357"
ycb_data_folder	String	The folder that contains the ycb data.

3 DATA STRUCTURE AND USAGE

This section explains the data organization, the supplementary programs and the YCB benchmarking project website.

3.1 Structure

The data are ordered by object ID, followed by the name of the objects. For each object four compressed files are supplied:

- 'berkeley_processed' file contains
 - A point cloud in .ply extension obtained by merging the data acquired from all the viewpoints.
 - Poisson meshes.
 - TSDF meshes.

- 'berkeley_rgb_highres' file contains
 - 600 image with 12.2 megapixel resolution in JPEG format,
 - pose of the RGB camera for each image in Hierarchical Data Format (HDF5; (The HDF Group, 2016)) with '.h5' extension and in JSON format with '.json' extension
 - camera intrinsic parameters in HDF5 and JSON format,
 - segmentation masks in '.pbm' format.
- 'berkeley_rgbd' file contains
 - 600 RGB-D images in HDF5 format,
 - pose of the RGB-D camera in HDF5 and JSON format for each image,
 - camera intrinsic parameters in HDF5 and JSON format,
 - segmentation masks in '.pbm' format.
- 'google_16k' file contains meshes with 16 thousand vertices.
- 'google_64k' file contains meshes with 64 thousand vertices.
- 'google_512k' contain meshes with 512 thousand vertices.

For all the mesh:

- Textureless meshes are provided in '.xml', '.stl', '.ply' formats.
- Textured meshes are provided in '.mtl' and '.obj' formats
- Texture maps are provided in '.png' format.
- Point clouds are generated in '.ply' format.

3.2 Supplementary Programs

In order to make the usage of the dataset easier, we provide the following programs available at (YCB-Benchmarks, 2016a):

- A Python script for downloading the data.
- A Python script for generating point clouds from the RGB-D data.
- A ROS package for managing the data, generating point clouds and URDF files.

3.2.1 Python Script for Downloading Data:

The Python script "ycb_downloader.py" can be used for downloading any data in the dataset. The user needs to set objects_to_download and files_to_download parameters as explained in Table 2.

3.2.2 Python Script for Generating Point Clouds:

The script "ycb_pointclouds.py" generates a point cloud file in '.pcd' format for a given RGB-D data and calibration files. The viewpoint of the camera for generating the point cloud and the turn table angle should be selected as indicated in Table 3.

3.2.3 YCB Benchmarks ROS Package:

The package provides ROS service interface for downloading data, deleting data, generating point clouds and generating URDF files. Fields of the services are summarized in Table 4. For the generated URDF files, the object mass attribute is automatically written in the corre-

Table 4: Service request and response fields of the services provided by ycb_benchmarks ROS node.

	Parameter name	Type	Values
Service Request	object_id	String array	<ul style="list-style-type: none"> Any combinations of ids in Table 1. Alternatively, full name of the object can be given as "002_master_chef_can". <p>To download the files for all the objects, set this parameter to "all".</p>
	data_type	String array	Any combination of "berkeley_rgb-highres", "berkeley_rgbd", "berkeley_processed", "google_16k", "google_64k", "google_512k"
	view-point_camera	Integer array	Camera from which the viewpoint is generated. Integers should be between 1 and 5.
Service Response	success	Bool	Returns 'True' if operation is successful 'False' if otherwise.
	error_message	String	Returns empty string if operation is successful. Returns the reason of failure otherwise.

sponding field, but the inertia matrix is written as identity as this information is not yet available for the objects.

3.3 YCB Project Website

The YCB project website (YCB-Benchmarks, 2016b) is designed as a hub for robotic manipulation community. Via this website, researchers can present, compare and discuss results obtained by using the YCB dataset. Additionally researchers can also propose protocols and benchmarks for manipulation research. We believe that this interaction within the community will help to make the dataset a commonly used tool, and therefore, substantiate its value for benchmarking in manipulation research.

Acknowledgements

The authors would like to thank to David Butterworth and Shushman Choudhury from Carnegie Mellon University for their help restructuring the data files.

References

- Amazon (2016a) *Amazon Simple Storage Service (Amazon S3)*. Available at: <https://aws.amazon.com/s3/>.
- Amazon (2016b) *Amazon Web Services Public Dataset Program*. Available at: <https://aws.amazon.com/public-data-sets/>.
- Bruce, J., Balch, T. and Veloso, M. (2000) 'Fast and Inexpensive Color Image Segmentation for Interactive Robots', in *Proceedings of 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2061–2066.
- Bylow, E., Sturm, J., Kerl, C., Kahl, F. and Cremers, D. (2013) 'Real-Time Camera Tracking and 3D Reconstruction Using Signed Distance Functions', in *Robotics: Sci. and Sys. (RSS) Conference*.
- Calli, B., Singh, A., Walsman, A., Srinivasa, S., Abbeel, P. and Dollar, A. M. (2015) 'The YCB Object and Model Set: Towards Common Benchmarks for Manipulation Research', in *Proceedings of 2015 Int. Conf. on Advanced Rob. (ICAR)*. Istanbul, pp. 510–517.
- Calli, B., Walsman, A., Singh, A., Srinivasa, S., Abbeel, P. and Dollar, A. M. (2015) 'Benchmarking in Manipulation Research: Using the Yale-CMU-Berkeley Object and Model Set', *IEEE Robotics and Automation Magazine*, 22(3), pp. 36–52.
- Calli, B., Walsman, A., Singh, A., Srinivasa, S., Abbeel, P. and Dollar, A. M. (2015) 'Benchmarking in Manipulation Research Using the Yale-CMU-Berkeley Object and Model Set', *Robotics & Automation Magazine*, 22(3), pp. 36–52.
- Chitta, S., Sucan, I. and Cousins, S. (2012) 'MoveIt!', *IEEE Robotics & Automation Magazine*, 19(1), pp. 18–19.
- Curless, B. and Levoy, M. (1996) 'A volumetric method for building complex models from range images', in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pp. 303–312.
- Goldfeder, C., Ciocarlie, M., Dang, H. and Allen, P. K. (2009) 'The Columbia Grasp Database', in *Proceedings of 2009 IEEE International Conference on Robotics and Automation*, pp. 1710–1716.
- Kasper, A., Xue, Z. and Dillmann, R. (2012) 'The KIT Object Models Database: An Object Model Database for Object Recognition, Localization and Manipulation in Service Robotics', *The International Journal of Robotics Research*, 31(8), pp. 927–934.
- Kazhdan, M., Bolitho, M. and Hoppe, H. (2006) 'Poisson Surface Reconstruction', in *Proceedings of Fourth Eurographics Symposium on Geometry Processing*, pp. 61–70.
- Koenig, N. and Howard, A. (2004) 'Design and Use Paradigms for Gazebo, an Open-Source Multi-Robot Simulator', in *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2149–2154.
- Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R. and Ng, A. Y. (2009) 'ROS: An Open-Source Robot Operating System', in *ICRA Workshop on Open Source Software*.
- Sameer, A. and Mierle, K. (2012) 'Ceres solver: Tutorial & reference', *Google Inc.*
- Scharstein, D. and Szeliski, R. (2003) 'High-accuracy Stereo Depth Maps Using Structured Light', in *Proceedings of 2003 IEEE Comp. Society Conf. on Com. Vision and Pattern Rec.*, pp. 195–202.
- Singh, A., Sha, J., Narayan, K. S., Achim, T. and Abbeel, P. (2014) 'BigBIRD: A large-scale 3D database of object instances', in *Proceedings of 2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 509–516.
- The HDF Group (2016) *Hierarchical Data Format, version 5*. Available at: <http://www.hdfgroup.org/HDF5/>.
- YCB-Benchmarks (2016a) *YCB Benchmarks Amazon Page*. Available at: <http://ycb-benchmarks.s3-website-us-east-1.amazonaws.com/>.
- YCB-Benchmarks (2016b) *YCB Benchmarks Main Page*. Available at: <http://ycbbenchmarks.org/>.