Yale-CMU-Berkeley dataset for robotic manipulation research

**Berk Calli[1], Arjun Singh[2], James Bruce[4], Aaron Walsman[3], Kurt Konolige[4],
Siddhartha Srinivasa[3], Pieter Abbeel[2] and Aaron M Dollar[1]**

Abstract

*In this paper, we present an image and model dataset of the real-life objects from the Yale-CMU-Berkeley 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 three-dimensional 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 node are
provided to download the data, generate point clouds and create Unified Robot Description Files. The dataset is also sup-
ported by our website, www.ycbbenchmarks.org, which serves as a portal for publishing and discussing test results along
with proposing task protocols and benchmarks.*

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/. Compared to other object datasets in
the literature (including ones widely utilized by the robotics
community (Goldfeder et al., 2009; Kasper et al., 2012;
Singh et al., 2014; a comprehensive overview is given by
Calli et al., 2015b), our dataset has four major advantages.
Firstly, the objects are a part of the Yale-CMU-Berkeley
(YCB) Object Set (Calli et al., 2015a, 2015b), 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. Secondly, 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 sce-
narios with shape and texture variety, and relationships
with a range of tasks. Thirdly, the quality of the data pro-
vided by our dataset is significantly greater than that of
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,
Figures 1 and 2) and a Google scanner (Section 2.2, Figure
3). The BigBIRD Object Scanning Rig provides 600 RGB
and 600 RGB-D images for each object along with segmen-
tation masks and calibration information for each image.
Two kinds of textured mesh models are generated using
these data by utilizing Poisson reconstruction (Kazhdan
et al., 2006) and Truncated Signed Distance Function
(TSDF) (Curless and Levoy, 1996) techniques. With the
Google scanner, the objects are scanned at three resolution
levels (16k, 64k, 512k mesh vertices). We believe that sup-
plying these model sets with different quality and acquisi-
tion techniques will be useful for the community to

[1]Mechanical Engineering and Material Science Department, Yale
University, USA
[2]Electrical Engineering and Computer Sciences Department, University of
California, Berkeley, USA
[3]Robotics Institute, Carnegie Mellon University, USA
[4]Google, USA

Corresponding author:
Berk Calli, 9 Hillhouse Avenue, New Haven, CT-06511, USA.
E-mail: berk.calli@yale.eu

determine best practices in manipulation simulations and also to assess the effect of model properties on 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 those that do not have a determinate shape (i.e. table cloth, T-shirt, rope, plastic chain), and those that are too small for the scanning systems (i.e. nails, washers).

The dataset is hosted by the 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. In addition, 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 et al., 2012).

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 Calli et al. (2015b), 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 (22.0 mm × 14.7 mm), a pixel size of 5.2 μm, 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 Figure 1. These pairs are arranged in a quarter-circular arc focusing on a motorized turntable placed at a photobench, as depicted in Figure 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.

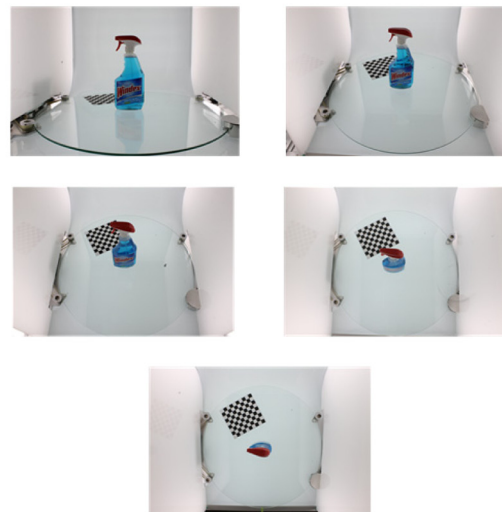**Fig. 1.** A Canon Rebel T3 RGB camera and PrimeSense Carmine 1.08 RGB-D sensor pair mounted together.

Fig. 2. BigBIRD Scanning Rig and viewpoints of the five cameras.

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

Table 1. The objects scanned in the Yale-CMU-Berkeley object and model set. Note that the object IDs are kept consistent with Calli et al. (2015b); there are jumps since some objects are skipped as they do not have a determinate shape or are 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	1131
2		Master chef can	414	12		Strawberry	18	22		Windex bottle	1022
3		Cracker box	411	13		Apple	68	23		Wineglass	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		Sponge	6.2
7		Tuna fish can	171	17		Orange	47	27		Skillet	950
8		Pudding box	187	18		Plum	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

Table 1. Continued

ID	Picture	Object name	Mass (g)	ID	Picture	Object name	Mass (g)	ID	Picture	Object name	Mass (g)
31		Spoon	30	43		Philips screwdriver	97	55		Baseball	148
32		Knife	31	44		Flat screwdriver	98.4	56		Tennis ball	3.6
33		Spatula	51.5	46		Plastic bolt	3.6	57		Racquetball	1
35		Power drill	895	47		Plastic nut	1	58		Golf ball	665
36		Wood block	729	48		Hammer	665	61		Foam brick	59
37		Scissors	82	49		Small clamp	19.2	62		Dice	125
38		Padlock	304	50		Medium clamp	59	63 (a-e)		Marbles	202
39		Keys	10.1	51		Large clamp	125	65 (a-k)		Cups	123
40		Large marker	15.8	52		Extra-large Clamp	202	70 (a-b)		Colored wood bloc	729
41		Small marker	8.2	53		Mini soccer ball	123	71 (a-b)		Nine-hole Peg Test	82
42		Adjustable wrench	252	54		Softball	191	72 (a-k)		Toy airplane	304

Table 1. Continued

ID	Picture	Object name	Mass (g)	ID	Picture	Object name	Mass (g)	ID	Picture	Object name	Mass (g)
								73 (a-m)		Lego	10.1
								76		Timer	8.2
								77		Rubik's cube	252

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 15 cameras (five Canon T3s, five Carmines with an RGB camera and infrared 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 by 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 5 minutes.

The collected images are used in the surface reconstruction procedure to generate meshes. Two sets of textured mesh models are obtained using Poisson reconstruction (Kazhdan et al., 2006) and TSDF (Bylow et al., 2013) methods. While the Poisson method provides watertight meshes, the 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 the 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 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 (Figure 3). Each scanhead is a custom structured light (Scharstein and Szeliski, 2003) capture unit, consisting of a consumer digital light processing (DLP) projector, two monochrome cameras in a stereo pair and a color camera to capture fine detail texture/color information (in total three cameras per scanhead). The consumer projector is an ACER P7500 with 1920 × 1080 resolution and 4000 lumens. The monochrome cameras are a Point Grey Grasshopper3 with Sony IMX174 CMOS sensors and 1920 × 1200 resolution. The lenses on the monochrome cameras are Fujinon HF12.5SA-1 with 12.5 mm focal length. The color camera is a Canon 5dMk3 with 5760 × 3840 pixel 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 and triangulated into a depth map. An image from the digital single-lens reflex (DSLR) camera 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 fully automated processes. The scanner is calibrated using CMVision (Bruce et al., 2000) for pattern detection and Ceres-Solver
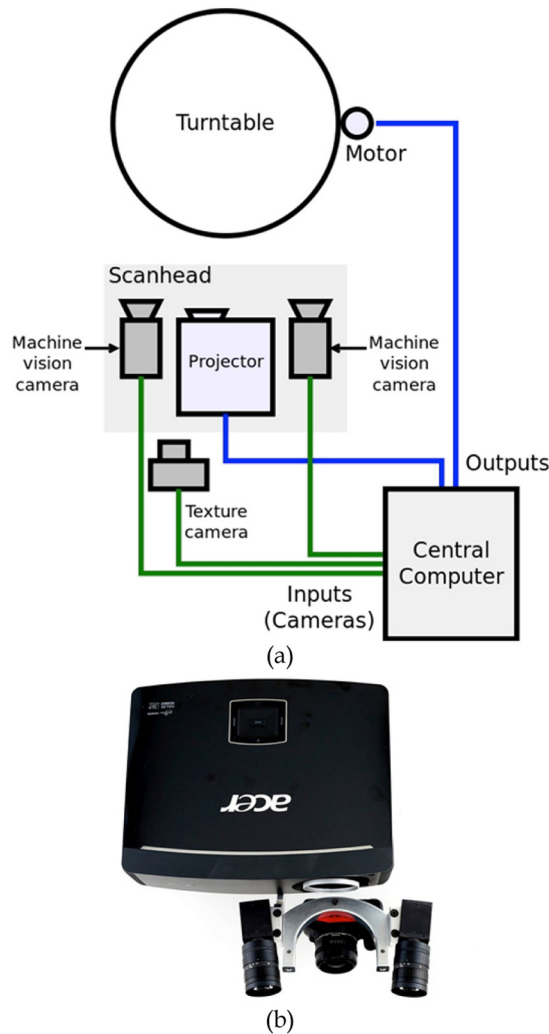
Fig. 3. Google scanner: (a) the overall scheme; (b) projector coupled with two monochrome cameras and a digital single-lens reflex camera.

(Sameer and Mierle, 2012) is used to optimize the camera parameters for all nine cameras in parallel. The objects were scanned using eight 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 cannot be generated with this scanner for objects 23, 39, 46, 47 and 63-c-f, and a partially distorted model is generated for object 22.

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 as follows.

- The 'berkeley_processed' file contains the following:
  - a point cloud in .ply extension obtained by merging the data acquired from all the viewpoints;
  - Poisson meshes;
  - TSDF meshes.
- The 'berkeley_rgb_highres' file contains the following:
  - 600 images with 12.2 megapixel resolution in JPEG format;
  - the 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 formats;
  - segmentation masks in '.pbm' format.
- The 'berkeley_rgbd' file contains the following:
  - 600 RGB-D images in HDF5 format;
  - the pose of the RGB-D camera in HDF5 and JSON format for each image;
  - camera intrinsic parameters in HDF5 and JSON formats;
  - segmentation masks in '.pbm' format.
- The 'google_16k' file contains meshes with 16,000 vertices.
- The 'google_64k' file contains meshes with 64,000 vertices.
- The 'google_512k' contain meshes with 512,000 vertices.

For all the mesh:

- textureless meshes are provided in '.xml', '.stl' and '.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 and 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.

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.

Table 4. Service request and response fields of the services provided by the ycb_benchmarks Robot Operating System node.

	Parameter name	Type	Values
Service Request	object_id	String array	Any combinations of IDs in Table 1. Alternatively, full name of the object can be given as '002_master_chef_can'. To download the files for all the objects, set this parameter to 'all'.
	data_type	String array	Any combination of 'berkeley_rgb-highres', 'berkeley_rgbd', 'berkeley_processed', 'google_16k', 'google_64k', 'google_512k'
	viewpoint_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.

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

3.2.3 YCB benchmarks ROS package. The package provides a ROS service interface for downloading data, deleting data, generating point clouds and generating URDF files. The fields of the services are summarized in Table 4. For the generated URDF files, the object mass attribute is automatically written in the corresponding 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 the robotic manipulation community. Via this website, researchers can present, compare and discuss the results obtained by using the YCB dataset. In addition, 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

References

Amazon (2016a) Amazon Simple Storage Service (Amazon S3). Available at: https://aws.amazon.com/s3/ (accessed 16 March 2017).

Amazon (2016b) Amazon Web Services Public Dataset Program. Available at: https://aws.amazon.com/public-data-sets/ (accessed 16 March 2017).

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*, Takamatsu, 31 October–5 November 2000, vol. 3, pp.2061–2066.

Bylow E, Sturm J, Kerl C, et al. (2013) Real-time camera tracking and 3D reconstruction using signed distance functions. In: *Robotics: Science and systems (RSS) conference*, Berlin, 24–28 June 2013.

Calli B, Singh A, Walsman A, et al. (2015a) The YCB Object and Model Set: towards common benchmarks for manipulation research. In: S Kalkan and U Saranli (eds.) *Proceedings of 2015 international conference on advanced robots (ICAR)*, Istanbul, 27–31 July 2015, pp.510–517.

Calli B, Walsman A, Singh A, et al. (2015b) Benchmarking in manipulation research: using the Yale-CMU-Berkeley Object and Model Set. *IEEE Robotics and Automation Magazine* 22(3): 36–52.

Chitta S, Sucan I and Cousins S (2012) MoveIt! *IEEE Robotics & Automation Magazine* 19(1): 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)*, New Orleans, 4–9 August 1996, pp.303–312. New York: ACM Press.

Goldfeder C, Ciocarlie M, Dang H, et al. (2009) The Columbia Grasp Database. In: *Proceedings of 2009 IEEE international conference on robotics and automation*, Kobe, 12–17 May 2009. 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): 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)*, Sendai, 28 September–2 October 2004, pp.2149–2154.

Quigley M, Conley K, Gerkey B, et al. (2009) ROS: an open-source robot operating system. In: Hirukawa H and Knoll A (eds) *ICRA workshop on open source software*, Kobe, 17 May 2009.

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: Danielle M (ed.) *Proceedings of 2003 IEEE Comp. Society conference on com. vision and pattern rec*, Los Alamitos, 18–20 June 2003, pp.195–202. USA: Victor Graphics.

Singh A, Sha J, Narayan KS, et al. (2014) BigBIRD: a large-scale 3D database of object instances. In: *Proceedings of 2014 IEEE international conference on robotics and automation (ICRA)*, Hong Kong, 31 May–7 June 2014, pp.509–516.

The HDF Group (2016) Hierarchical Data Format, version 5. Available at: http://www.hdfgroup.org/HDF5/ (accessed 16 March 2017).

YCB-Benchmarks (2016a) YCB Benchmarks Amazon Page. Available at: http://ycb-benchmarks.s3-website-us-east-1.amazonaws.com/ (accessed 16 March 2017).

YCB-Benchmarks (2016b) YCB Benchmarks Main Page. Available at: http://ycbbenchmarks.org/ (accessed 16 March 2017).