BigBIRD: A Large-Scale 3D Database of Object Instances

Arjun Singh, James Sha, Karthik S. Narayan, Tudor Achim, Pieter Abbeel

***Abstract*— The state of the art in computer vision has rapidly advanced over the past decade largely aided by shared image datasets. However, most of these datasets tend to consist of assorted collections of images from the web that do not include 3D information or pose information. Furthermore, they target the problem of object category recognition—whereas solving the problem of object instance recognition might be sufficient for many robotic tasks. To address these issues, we present a high-quality, large-scale dataset of 3D object instances, with accurate calibration information for every image. We anticipate that "solving" this dataset will effectively remove many perception-related problems for mobile, sensing-based robots.**

The contributions of this work consist of: (1) BigBIRD, a dataset of 100 objects (and growing), composed of, for each object, 600 3D point clouds and 600 high-resolution (12 MP) images spanning all views, (2) a method for jointly calibrating a multi-camera system, (3) details of our data collection system, which collects all required data for a single object in under 6 minutes with minimal human effort, and (4) multiple software components (made available in open source), used to automate multi-sensor calibration and the data collection process. All code and data are available at http://rll.eecs.berkeley.edu/bigbird.

I. Introduction and Related Work

Object recognition, the task of identifying a given object in an image, remains an unsolved problem in computer vision. Researchers typically dichotomize object recognition into (1) category-level recognition, where various concrete objects are assigned a single label (e.g. "Pepsi can" and "Coke can" are both assigned the label "soda can") and (2) instance-recognition, where each concrete object is assigned a separate label (e.g. "Pepsi can" and "Coke can" are given separate labels). The computer vision and robotics approaches to the recognition problem differ fundamentally in that (1) robots in fixed environments typically need to interact with on the order of a few hundred objects and (2) robotic perception algorithms need to successfully localize and detect 3D object poses in addition to identifying the correct object. We believe that instance recognition suits many robotic tasks well, as joint object detection and pose estimation are the primary components of the instance recognition problem.

Though the advent of commodity RGB-D sensors, such as the Microsoft Kinect, aid in addressing 3D pose detection and localization by providing a depth channel in addition to a color channel, instance recognition systems still cannot reliably detect hundreds of objects [1], [2], [3]. We believe that the primary issue currently hampering progress towards reliable and robust instance recognition is the lack of a large-scale dataset containing high-quality 3D object data; this is because collecting such a dataset requires constructing a reliable and high-quality 3D scanning system, which is a significant undertaking.

Department of Electrical Engineering and Computer Science, University of California, Berkeley, Berkeley, CA 94709

Fig. 1. Our data-collection system. We place the object near the center of the turntable, and our software takes care of the rest. Note the chessboard on the turntable to merge clouds as the turntable moves. Sample objects from the dataset can be seen sitting on top of the Ortery Photobench.

A. Datasets

The last decade has witnessed rapid advances in computer vision largely due to fundamental image datasets, such as MNIST, Caltech-101, PASCAL, Labeled Faces in the Wild, PASCAL, and most recently, ImageNet [4], [5], [6], [7], [8], [9]. Unfortunately, the solution to most current 2D vision datasets would not constitute a solution to instance recognition as they currently target image retrieval tasks from arbitrary images drawn from the web. In particular, while some of these tasks emphasize detection, they do not directly address the problem of pose estimation, a component crucial to attaining high performance in instance recognition and robotic tasks. While there exist several 3D vision datasets, most datasets either (1) have few objects, (2) have low-quality objects, (3) provide only single views of objects or scenes, (4) don't contain calibration and pose information, or (5) provide low-resolution RGB data [10], [11], [12], [13], [14], [15]. While addressing all five aspects would improve the quality of instance recognition systems, aspect (5) would also provide a venue to explore synergies and comparisons

between Kinect-style and multi-view stereo approaches to 3D model construction [16], [17], [18].

Furthermore, although most recent instance recognition systems work with RGB-D data, there exist high-quality instance recognition systems that use only RGB images, such as MOPED, presented by Collet et al. [19]. However, these generally work with higher-quality RGB images than those provided by RGB-D sensors. Unfortunately, this makes it quite difficult to compare RGB-D instance recognition systems with RGB-only systems, as simply applying the RGB-only systems to the images from RGB-D datasets would yield unrepresentative results. Because we provide high-quality RGB images in addition to the RGB-D data, we can enable meaningful comparison of these systems.

The closest work to ours is that of Kasper et al. [20]. They have a similar setup in which a laser scanner collects 3D data and a stereo pair collects data from 360 points from the viewing hemisphere. They also provide object meshes and calibrated RGB data. However, their 3D data collection setup is only semi-automated and their image collection setup takes an additional 20 minutes. Although they provide a relatively large number of objects (roughly 130 at the time of writing), scaling up to thousands may be infeasible at that speed. Our approach is fully automated after placing the object in the system, and data collection takes less than 5 minutes per object.

B. Data Collection

The chief obstacle to collecting a high-quality large-scale object dataset involves constructing a reliable 3D scanning system that can provide both high-quality depth and color information. Most commercial 3D scanners either provide only range sensor and low-resolution color information and/or are very expensive. Recent work demonstrates that KinectFusion variants can provide high-quality 3D reconstructions [21], [22], [23], [24]. However, some of these approaches don't provide calibrated RGB images, which are required by many instance recognition platforms, and those that do only provide low-resolution RGB images from the Kinect sensor. Further, the data collection process requires a human to slowly move a Kinect around the full object; even with an automated turntable, a single Kinect attached to an arm cannot image non-convex objects and translucent/transparent objects due to the inherent limitations of Kinect-style RGB-D sensors.

Using multiple Kinects and high-resolution DSLR cameras along with an automated turntable constitutes one possible approach to jointly reducing human effort while improving RGB-D mesh quality. The presence of multiple types of sensors determines highly accurate intrinsics for each sensor as well as relative transformations between pairs of sensors. Researchers have extensively studied this problem for both single and multiple 2D cameras and have recently explored it for single and multiple RGB-D sensors [25], [26], [27], [28], [29], [30], [31]. Typical approaches involve first calibrating each sensor individually to compute its intrinsics, computing stereo pairs between sensors to estimate each sensor's extrinsics, and then running a joint optimization procedure to refine each sensor's intrinsics and extrinsics. For calibrating RGB-D sensors, many approaches require additional hardware and/or setup from what is required for 2D cameras. For example, Herrera et al. [25] present a method that requires affixing a chessboard to a large, flat plane, whereas typical 2D approaches simply require a chessboard alone. Our method requires a source of infrared light, but no additional hardware setup.

Additionally, interference between IR patterns complicates constructing a data-collection system with multiple RGB-D sensors. Butler et al. [32] propose an approach for mitigating interference from multiple depth sensors. However, their approach requires affixing a vibrating motor to each device, which makes a static calibration procedure impossible and also introduces more complexity into the system. We employ time-multiplexing, another common approach, which involves turning off each camera when it is not taking a picture. Concretely, we turn off the infrared emitter, which is roughly two times faster than turning off the depth stream.

C. Contributions

To address the issues described above, we present the following contributions:

1) A dataset which addresses the various shortcomings of existing 2D and 3D datasets by providing the following data per object: (1) 600 Kinect-style RGB-D images, (2) 600 high-resolution images, (3) accurate calibration information for every image, (4) segmented objects per image, and (5) full-object meshes. We obtain 600 images by taking shots from 5 polar angles and 120 azimuthal angles, the latter equally spaced by 3°.
2) A method for jointly calibrating multiple RGB-D sensors and cameras.
3) Details of our data collection system, which can collect all required data for a single object in under 6 minutes, where the only human effort required involves placing an object on the turntable and running a single command.
4) Multiple software components, including software for calibrating a single depth sensor, software for jointly calibrating multiple sensors (RGB-D and 2D RGB), and tools to simplify the data collection process.

In addition to helping to solve the instance recognition problem, we believe that our dataset removes many obstacles associated with large-scale 3D data and serves as a unified dataset that bridges problems in graphics, computer vision, and robotics. Our dataset can be used for benchmarks in multiple areas, such as 3D mesh reconstruction (with and without RGB-D), instance recognition, and object categorization. We intend to continually add to our dataset, inviting others to request and/or send us objects for which we have not yet collected data. Test scenes and results will be made available as well.

All data and code are available and regularly updated at the following URL: `http://rll.eecs.berkeley.edu/bigbird`. Available code includes a robust checkerboard

Fig. 2. Carmine mounted to Canon T3 using RGBDToolkit mount.

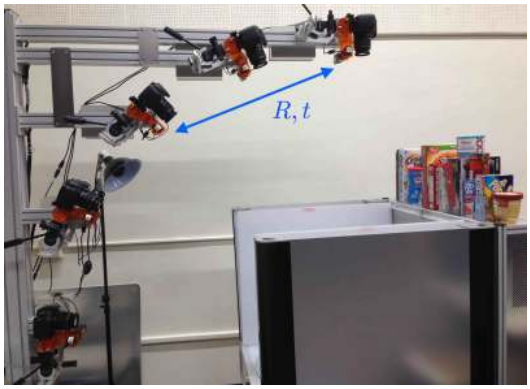

Fig. 3. Side view of all Carmines mounted to respective Canon T3s, pointed at the Ortery Photobench. The dimensions of the Photobench are 31" D x 26"H x 26" W.

detector, calibration software for a single RGB-D sensor, multi-sensor calibration, and various utilities for working with depth sensors and point clouds in Python.

II. System Overview

The sensors in our system comprise of 5 high resolution (12.2 MP) Canon Rebel T3 cameras and five PrimeSense Carmine 1.08 depth sensors. We mount each Carmine to one of the T3s using a mount designed by RGBDToolkit [33], as shown in Figure 2. Each T3 is then mounted to the Ortery MultiArm 3D 3000.

We place each object on the turntable in the Ortery Photobench 260. The Photobench contains a glass turntable, which can be rotated in units of 0.5 degrees. It also has four lights, consisting of 4000 LEDs, located at the bottom, the back wall, the front corners, and the back corners. Using a reverse-engineered driver, we can programmatically control the lighting and rotation of the turntable.

To obtain calibrated data, we place a chessboard on the turntable; the chessboard is always fully visible in at least one of the cameras, specifically the Canon and Carmine directly above the turntable (see 3). We refer to Carmine as the *reference camera*. After calibrating all of the cameras to find the transformations from each camera to the reference camera, we can provide a good estimate of the pose for every image.

For each object, we capture images with each camera at each turntable position. We rotate the turntable in increments of 3 degrees, yielding a total of 600 point clouds from the Carmines and 600 high-resolution RGB images from the Canon T3s. We then estimate poses for each camera, segment each cloud and generate segmentation masks for each of the 600 views, and produce a merged cloud and mesh model.

Automation and speed are crucial to enabling large-scale data collection; a significant amount of engineering is required to make the process as fast as possible.

Our system runs the following steps when collecting data for a single object:

1) Start the depth and color stream for each Carmine. Turn off the infrared emitter for each Carmine.
2) Repeat for each turntable orientation (every 3 degrees, 120 total orientations):
   a) Start a thread for each Canon T3 that captures an image.
   b) Start a thread for each Carmine that captures a color image.
   c) Start a single thread that loops through each Carmine, turning on the infrared emitter, capturing a depth map, and turning off the infrared emitter in sequence.
   d) Once all of the above threads are done executing in parallel, rotate the turntable by 3 degrees.

Using all Carmines simultaneously causes the projected infrared patterns to interfere, leading to severe degradations in data quality. One option involves stopping the depth stream for each device not taking a depth image, and restarting the depth stream immediately before taking an image. However, stopping and starting a depth stream takes roughly 0.5s, imposing a 5 minute minimum bound on collecting 120 images with each of the 5 cameras. Rather than stopping the entire stream, we modified the OpenNI2 library to allow turning off the infrared emitter while keeping the depth stream open, which takes 0.25s. We present detailed timing breakdowns in Table I.

We now discuss how we jointly calibrate the sensors.

III. Calibration

The 10 sensors are situated in a quarter-circular arc, with each Carmine mounted to a Canon T3, and each Canon T3 mounted to the arm. One of the overhead cameras, referred to as the reference camera, can always see the chessboard affixed to the turntable; specifically, we use the overhead Carmine. In order to recover the pose of all of the other sensors, we must estimate the transformation from each sensor to the reference camera.

Kinect-style RGB-D sensor calibration involves estimating the intrinsic matrix for the infrared (IR) camera, the intrinsic matrix for the RGB camera, and the extrinsic rigid transform from the RGB camera to the infrared camera. Highly accurate calibration is crucial to achieving strong depth-to-color

Step	Time (s)
Startup	
Ortery Photobench Startup	3.5
Carmine Startup (depth and color)	9.3
Capture at each turntable position (done 120 times)	
Capture images – performed in parallel	1.82
Capture Canon T3 images (all 5 in parallel)	1.2
Capture Carmine color (all 5 in parallel)	0.07
Capture Carmine depth (all 5 in sequence)	1.82
Rotate turntable	0.48
Total capture time	276
Shutdown	0.49
Total time for one object	289

TABLE I

TIMING INFORMATION FOR THE DATA-COLLECTION PROCESS. NOTE THAT THE THREE IMAGE CAPTURE THREADS ALL RUN IN PARALLEL, WHICH MEANS THAT THE IMAGE CAPTURE STEP TAKES AS LONG AS THE LONGEST PROCESS.

registration. In our system, we not only need to calibrate the intrinsics of each individual RGB-D sensor, but also the extrinsics which yield the relative transformations between each of the 10 sensors, both RGB-D and RGB.

Accurate calibration also enables registering depth maps to different RGB images, including the higher-resolution 1280x1024 image provided by the Carmine (hardware registration only works when the color stream is at the same resolution as the 640x480 depth stream). Although this is a relatively well-studied problem [30], [25], obtaining strong results is still nontrivial due to multiple details about the Carmines that are not well documented.

Our method requires an external infrared light and a calibration chessboard. At a high level, we take pictures of the chessboard with the high-resolution RGB camera and the RGB-D sensor's infrared camera and RGB cameras[1], 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 transformations for all fifteen cameras (five Canon T3s, five Carmines with an RGB camera and IR camera each) using OpenCV's camera calibration routines, based on the simple calibration method proposed by Zhang [28]. 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 sensors.

A. Joint Optimization

We use an approach similar to that given by Le and Ng [29]. Their approach requires that all sensors be grouped into *3D systems*. A stereo pair of cameras (RGB or IR) yields one kind of 3D system (a stereo system), and a RGB-D sensor's infrared camera and projector yield the other (a RGBD system). Each 3D system has intrinsic parameters, used to produce 3D points, and extrinsic parameters, used to transform 3D points into another system's coordinate frame. We construct and solve the optimization problem using Ceres Solver [34].

The calibrator optimizes the intrinsic and extrinsic parameters such that 1) each 3D system produces 3D points that match the physical characteristics of the chessboard (e.g. the points are all planar, the points on a given chessboard row are linear, and the distance between generated 3D points match up with the true distance on the chessboard) and 2) all 3D systems agree with each other on the locations of the chessboard corners.

The intrinsic parameters of a RGBD 3D system consist of the intrinsic matrix K and distortion parameters of the sensor's IR camera. The intrinsic parameters of a stereo 3D system consist of the intrinsic matrices and distortion parameters of each camera, along with the rotation and translation from one camera to the other.

The loss function is given by

$$\mathcal{G} = \sum_{s \in \mathcal{S}} \sum_{u \in \mathcal{U}} I(s,u) + \sum_{s_1,s_2 \in \mathcal{S}} E(s_1,s_2,u)$$

where I denotes the intrinsic cost, E denotes the extrinsic cost, $\mathcal{S}$ denotes the set of all 3D systems and U denotes the calibration data (i.e. the chessboard corners).

Let $Q(s,u_i)$ be a function that produces a 3D point for the corner u_i using the intrinsic parameters of system s. For a stereo system, this entails triangulation, and for an RGBD system, this is simply converting image coordinates to world coordinates using the depth value provided by the sensor.

For a 3D system, the intrinsic cost is given by

$$\begin{aligned} I(s,u_i) = &\sum_{u_j \in \mathcal{U}} (||Q(s,u_i) - Q(s,u_j)|| - d_{ij})^2 \\ &+ \sum_{l \in \mathcal{L}} d(Q(s,u_i), l) \\ &+ d(Q(s,u_i), p) \end{aligned}$$

where d_{ij} is the ground-truth 3D distance between points i and j on the chessboard, $\mathcal{L}$ is the set of lines that corner u_i belongs to, p is the plane that corner u_i belongs to, and $d(Q(s,u_i),p)$ measures the distance from the generated 3D point to the plane.

The extrinsic cost is given by

$$E(s_1,s_2,u_i) = ||R_{12}Q(s_2,u_i) + t_{12} - Q(s_2,u_i)||^2$$

where R_{12} and t_{12} represent the rotation and translation needed to transform a point from the coordinate frame of 3D system s_2 to s_1.

The major difference between our approach and that of Le and Ng is that we add one additional term to the cost function for stereo pairs; specifically, we enforce that

[1]It is vital that the Carmine and chessboard remain completely still while both images are captured, as it is not possible to simultaneously take a color and infrared image.

epipolar constraints are satisfied by adding an additional term to the stereo intrinsic cost function:

$$I(s,u) = ||u_1^T F u_2||^2,$$

where F is the fundamental matrix implied by the current values of the stereo pair's intrinsic parameters, u_1 are the homogeneous coordinates of the calibration datum in the first camera, and u_2 are the homogeneous coordinates of the calibration datum in the second camera.

We obtain the depth intrinsic matrix K_{Depth} from the infrared intrinsic matrix by subtracting off the offset between the depth image and infrared image due to the convolution window used by the internal algorithm. We found the values suggested by Konolige and Mihelich [35] of -4.8 and -3.9 pixels the x and y directions, respectively, worked well. Figure 4 shows the results of registering the depth map to the RGB image using our calibration and also using hardware registration.

Fig. 4. Comparison of hardware and software registration. The left image shows a hardware-registered point cloud. Note the bleeding of the cardboard in the background onto the Pringles can and the low resolution of the color data. The right image shows a software-registered point cloud using our calibration. Most of the bleeding of the cardboard onto the can has been fixed, and we can use higher-resolution color data.

IV. 3D Model Generation

After calibrating each camera to the reference camera, we proceed with model generation. At a high level, we:

1) Collect data from each Carmine and Canon as the turntable rotates through 120 3° increments.
2) Filter each Carmine depth map to remove depth discontinuities (Section IV-A).
3) Generate point clouds for each Carmine view using calibration intrinsics.
4) Merge the 5 point clouds for each of the 120 scenes using calibration extrinsics.
5) Segment the object from the merged cloud (Section IV-C).
6) Improve the object cloud quality for each of the 120 scenes through plane equalization (Section IV-B).
7) Merge the 120 scenes together to form a single cloud using calibration extrinsics.
8) Create a mesh via Poisson Reconstruction [36], [37].

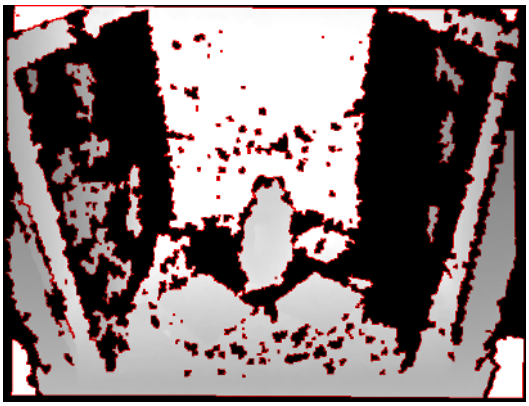

Fig. 5. Applying depth discontinuity filtering. Pixels marked in red are considered unreliable due to either a discontinuity or neighboring pixels that were not measured by the Carmine depth sensor. Before proceeding, we discard depth measurements associated with the red pixels.

A. Depth Discontinuity Filtering

After collecting data from each Carmine and Canon sensor, we run a depth discontinuity filtering step as suggested by Whelan et al. [38], since depth map discontinuities tend to yield imprecise depth and color measurements. To do so, we associate each 3×3 patch p in the depth image with a value $\max\{(\max p - p_{mid}), (\min p - p_{mid})\}$ where p_{mid} refers to the center pixel's depth. We keep all pixels whose associated patch has a value lesser than some threshold. See Figure 5 for an example of the pixels eliminated by depth discontinuity filtering.

B. Plane Equalization

After obtaining a preliminary 3D mesh, we produce a cleaner cloud through a procedure called *plane equalization*. As we collect point clouds, recall that we compute the transform from the turntable chessboard to the reference camera via OpenCV's `solvePnP`. Experimentally, we notice slight depth ambiguities when computing these transforms, evidenced by the non-aligned plane normals and inconsistent depths presented in Figure 6. Since we know that the turntable chessboard revolves about a circle roughly horizontal to the ground, we refine each transform's rotational component and translational component by (1) computing a new vector normal to be shared across all chessboards and (2) enforcing the centers of each chessboard to lie on a circle.

Concretely, given a set $\mathcal{T} = \{(R_1, t_1), \ldots, (R_n, t_n)\}$ of chessboard poses, we produce a refined set $\mathcal{T}' = \{(R'_1, t'_1), \ldots, (R'_n, t'_n)\}$ of chessboard poses. Note that an R_i operates on a plane with unit normal $\hat{k}$ yielding a plane with unit normal $R_i[3]$, the third column of R_i. Ultimately, we would like all plane normals to match; to do this, we compute a unit vector $\hat{u}$ so as to minimize $\sum_{i=1}^{n} (\hat{u} \cdot R_i[3])^2$. We solve for $\hat{u}$ exactly by setting it to be the least eigenvector of the covariance of all the $R_i[3]$s. We then compute each R'_i by multiplying each R_i by the transform that takes $R_i[3]$ to $\hat{u}$ via rotation about the axis $R_i[3] \times \hat{u}$. We next compute each t'_i by projecting each t_i onto the least squares circle determined by $\{t_1, \cdots, t_n\}$; this problem can be solved quickly by

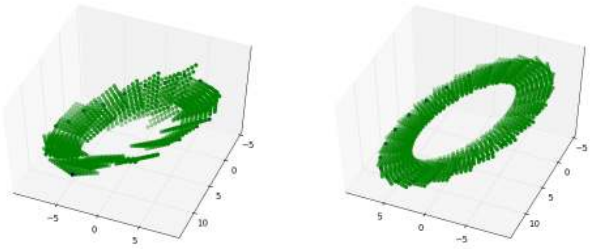

Fig. 6. The chessboard poses for each turntable location are shown in the frame of the reference camera. On the left, the chessboard poses are determined by solvePnP. On the right, we refine these pose estimates using the plane equalization method described in Section IV-B. The refined board poses are significantly cleaner.

projecting $\{t_1, \cdots, t_n\}$ onto a plane, computing the least squares circle in the plane's basis, and projecting each point onto the resulting circle. In practice, plane equalization runs in negligible time (< 0.1 s) for $n = 120$ and yields higher quality point clouds (see Figure 7).

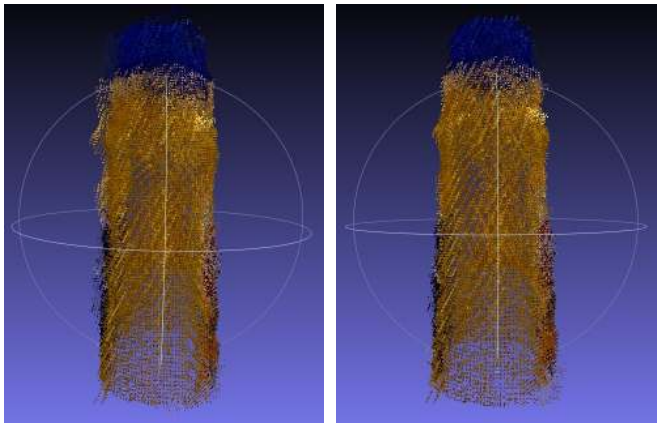

Fig. 7. Constructed point clouds for one object. On the left, the cloud is constructed using the raw solvePnP poses; the cloud has multiple shifted copies of the object due to misalignment. On the right, the cloud is constructed with the output of the plane equalization procedure; the cloud is much cleaner and well-aligned.

C. Object segmentation

As discussed above, for a given turntable angle, we merge the 5 Carmine point clouds into a single cloud using calibration extrinsics. To segment the object from this cloud, we first discard all points outside of the Ortery PhotoBench. We then discard all points below the turntable plane (which was identified in the previous step), and lastly conduct agglomerative clustering to remove tiny clusters of points.

D. Accuracy

Although we use a naive approach for building 3D models, their accuracy is better than the models used by Xie et al. [1] to obtain state-of-the-art RGBD instance recognition results. In Figure 8, we give a rough idea of the accuracy of our 3D models by projecting a representative mesh onto an image from one of the Canon cameras (which is not used to build the mesh), showing that the system is well calibrated and produces reasonable meshes. We expect that more sophisticated algorithms can produce higher-fidelity 3D models.

Fig. 8. The 3D mesh is projected onto one of the Canon images.

E. Limitations

Our approach relies solely on point cloud data from the Carmines when building the 3D mesh models. However, Kinect-style RGB-D sensors are known to perform poorly for certain objects, including transparent and highly-reflective objects, such as the bottle shown in Figure 9. For these objects, the 3D models may be missing or of poor quality. However, by incorporating methods that also use RGB data, we anticipate being able to provide high-quality 3D models for many of these objects in the future.

Fig. 9. An example object for which Kinect-style RGB-D sensors yield poor-quality point clouds.

V. Dataset Usage

We anticipate our dataset to be used for multiple related computer vision problems, including object instance recognition, object category recognition, and 3D object model generation. The dataset, and all code used to generate it, can be obtained at our website (http://rll.eecs.berkeley.edu/bigbird).

A. Obtaining the Dataset

Due to the large size (and many uses) of the dataset (each object has roughly three gigabytes of data), it is impractical to provide a single downloadable file for the entire dataset, and inconvenient to have a single downloadable file per object. On our website, we provide an automated way to download the data for various use-cases. Instructions for downloading the data are provided on the website. The settings can be configured to download whichever subset of the following components are desired:

1) High-resolution (12MP) images (.jpg)
2) Low-resolution Carmine images (.jpg)
3) Raw point clouds (.pcd)
4) Depth maps (.h5)
5) Segmented point clouds (.pcd)
6) Segmentation masks (.pbm)
7) 3D mesh model (.ply)

B. Instance Recognition

A set of test scenes using the objects from the dataset will be made available on the website. As we collect more objects, we will collect more test scenes. In order to facilitate meaningful comparison of similar algorithms, we plan to collect and record results of different methods on the same subset of the test data on the website.

C. 3D Model Generation

Although we have used a very naive approach to generating 3D models (i.e. concatenating all point clouds and running Poisson reconstruction), the data is well-suited for evaluating algorithms for generating 3D models of objects from RGB and RGB-D sources. The current release of the data does not contain ground-truth object models; however, we plan to obtain 3D-printed models and provide data for them. As more sophisticated algorithms are used on our data, we plan to provide better 3D models as well.

VI. Conclusions

We believe this dataset will significantly accelerate progress in robotic perception, especially the instance recognition problem. We also believe it can lead to benchmarks for a variety of areas from computer graphics, computer vision, and robotics, including 3D object reconstruction, recognition, and grasping.

All of our code and data, including calibration data, object instance data, and test scenes are available at the following URL: `http://rll.eecs.berkeley.edu/bigbird`. We plan to continue adding to our dataset; we invite others to request and/or send us objects which we have not yet scanned. Please contact us regarding such requests.

Acknowledgements

This work is supported in part by ONR Grant #N00014-12-1-0756 and by ONR YIP Award #N00014-13-1-0570. Arjun Singh is supported by an NDSEG Fellowship. Karthik Narayan is supported by an NSF Graduate Fellowship. We thank Ortery Technologies for their support.

References

[1] Ziang Xie, Arjun Singh, Justin Uang, Karthik S. Narayan, and Pieter Abbeel. Multimodal blending for high-accuracy instance recognition. In *IROS*, 2013.

[2] J. Tang, S. Miller, A. Singh, and P. Abbeel. A textured object recognition pipeline for color and depth image data. In *ICRA*, 2012.

[3] N. Vaskevicius, K. Pathak, A. Ichim, and A. Birk. The jacobs robotics approach to object recognition and localization in the context of the icra'11 solutions in perception challenge. In *ICRA*, 2012.

[4] Y. LeCun, C. Cortes, and C. J. C. Burges. The mnist database, 1998.

[5] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Comput. Vis. Image Underst.*, 106(1):59–70, 2007.

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[7] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, 2007.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[9] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *ICCV*, 2005.

[10] Willow Garage. Solutions in perception challenge, May 2011.

[11] A. Janoch, S. Karayev, Y. Jia, J. T. Barron, M. Fritz, K. Saenko, and T. Darrell. A category-level 3-d object dataset: Putting the kinect to work. 2011.

[12] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *ICRA*, 2011.

[13] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[14] J. Sturm, J. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *IROS*, 2012.

[15] D. Cremers and K. Kolev. Multiview stereo and silhouette consistency via convex functionals over convex domains. 33, 2011.

[16] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. *PAMI*, 32(8), 2010.

[17] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Towards internet-scale multi-view stereo. In *CVPR*, 2010.

[18] J. Guillemaut and A. Hilton. Joint multi-layer segmentation and reconstruction for free-viewpoint video applications. *Int. J. Comput. Vision*, 93(1):73–100, 2011.

[19] Alvaro Collet, Manuel Martinez, and Siddhartha S. Srinivasa. The MOPED framework: Object Recognition and Pose Estimation for Manipulation. 2011.

[20] Alexander Kasper, Zhixing Xue, and Rüdiger Dillmann. 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, 2012.

[21] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *UIST*, 2011.

[22] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011.

[23] Kaess M. Fallon M. F. Johannsson H. Leonard J. J. McDonald J. B. Whelan, T. Kintinuous: Spatially extended kinectfusion. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012.

[24] Q. Zhou, S. Miller, and V. Koltun. Elastic fragments for dense scene reconstruction. In *ICCV*, 2013.

[25] C. D. Herrera, J. Kannala, and J. Heikkilä. Accurate and practical calibration of a depth and color camera pair. In *CAIP*, 2011.

[26] C. Zhang and Z. Zhang. Calibration between depth and color sensors for commodity depth cameras. In *ICME*, 2011.

[27] A. Geiger, F. Moosmann, O. Car, and B. Schuster. A toolbox for automatic calibration of range and camera sensors using a single shot. In *ICRA*, 2012.

[28] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *ICCV*, 1999.

[29] Q. V. Le and A. Y. Ng. Joint calibration of multiple sensors. In *IROS*, 2009.

[30] Jan Smisek, Michal Jancosek, and Tomas Pajdla. 3d with kinect. In *Consumer Depth Cameras for Computer Vision*, pages 3–25. Springer, 2013.

[31] Michael Warren, David McKinnon, and Ben Upcroft. Online Calibration of Stereo Rigs for Long-Term Autonomy. In *International Conference on Robotics and Automation (ICRA)*, Karlsruhe, 2013.

[32] D Alex Butler, Shahram Izadi, Otmar Hilliges, David Molyneaux, Steve Hodges, and David Kim. Shake'n'sense: reducing interference for overlapping structured light depth cameras. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, pages 1933–1936. ACM, 2012.

[33] James George, Alexander Porter, Jonathan Minard, and Mike Heavers. Rgbd toolkit, 2013.

[34] Sameer Agarwal, Keir Mierle, and Others. Ceres solver.

[35] K. Konolige and P. Mihelich. Technical description of kinect calibration, 2013.

[36] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *SGP*, 2006.

[37] P. Cignoni, M. Corsini, and G. Ranzuglia. Meshlab: an open-source 3d mesh processing system. *ERCIM News*, 2008(73), 2008.

[38] Thomas Whelan, Hordur Johannsson, Michael Kaess, John J Leonard, and John McDonald. Robust tracking for real-time dense rgb-d mapping with kintinuous. 2012.