A Large-Scale Hierarchical Multi-View RGB-D Object Dataset

Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox

Abstract— **Over the last decade, the availability of public image repositories and recognition benchmarks has enabled rapid progress in visual object category and instance detection. Today we are witnessing the birth of a new generation of sensing technologies capable of providing high quality synchronized videos of both color and depth, the RGB-D (Kinect-style) camera. With its advanced sensing capabilities and the potential for mass adoption, this technology represents an opportunity to dramatically increase robotic object recognition, manipulation, navigation, and interaction capabilities. In this paper, we introduce a large-scale, hierarchical multi-view object dataset collected using an RGB-D camera. The dataset contains 300 objects organized into 51 categories and has been made publicly available to the research community so as to enable rapid progress based on this promising technology. This paper describes the dataset collection procedure and introduces techniques for RGB-D based object recognition and detection, demonstrating that combining color and depth information substantially improves quality of results.**

I. Introduction

The availability of public image repositories on the Web, such as Google Images and Flickr, as well as visual recognition benchmarks like Caltech 101 [9], LabelMe [24] and ImageNet [7] has enabled rapid progress in visual object category and instance detection in the past decade. Similarly, the robotics dataset repository RADISH [16] has greatly increased the ability of robotics researchers to develop and compare their SLAM techniques. Today we are witnessing the birth of a new generation of sensing technologies capable of providing high quality synchronized videos of both color and depth, the RGB-D (Kinect-style) camera [23], [19]. With its advanced sensing capabilities and the potential for mass adoption by consumers, driven initially by Microsoft Kinect [19], this technology represents an opportunity to dramatically increase the capabilities of robotics object recognition, manipulation, navigation, and interaction. In this paper, we introduce a large-scale, hierarchical multi-view object data set collected using an RGB-D camera. The dataset and its accompanying segmentation, and video annotation software has been made publicly available to the research community to enable rapid progress based on this promising technology. The dataset is available at http://www.cs.washington.edu/rgbd-dataset.

We thank Max LaRue for data collection and post-processing of a part of the dataset. This work was funded in part by an Intel grant, by ONR MURI grants N00014-07-1-0749 and N00014-09-1-1052, by the NSF under contract IIS-0812671, and through the Robotics Consortium sponsored by the U.S. Army Research Laboratory under Cooperative Agreement W911NF-10-2-0016.

Kevin Lai and Liefeng Bo are with the Department of Computer Science & Engineering, University of Washington, Seattle, WA 98195, USA. {kevinlai,lfb}@cs.washington.edu

Xiaofeng Ren is with Intel Labs Seattle, Seattle, WA 98105, USA. xiaofeng.ren@intel.com

Dieter Fox is with both the Department of Computer Science & Engineering, University of Washington, and Intel Labs Seattle. fox@cs.washington.edu

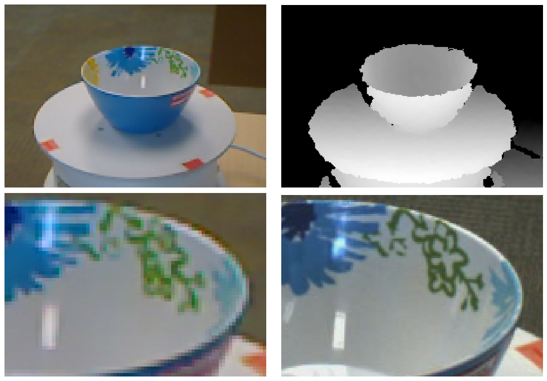

Fig. 1. (Top) Each RGB-D frame consists of an RGB image (left) and the corresponding depth image (right). (Bottom) A zoomed-in portion of the bowl showing the difference between the image resolution of the RGB-D camera and the Point Grey Grasshopper.

Unlike many existing recognition benchmarks that are constructed using Internet photos, where it is impossible to keep track of whether objects in different images are physically the same object, our dataset consists of multiple views of a set of objects. This is similar to the 3D Object Category Dataset presented by Savarese et al. [25], which contains 8 object categories, 10 objects in each category, and 24 distinct views of each object. The RGB-D Object Dataset presented here is at a much larger scale, with RGB and depth video sequences of 300 common everyday objects from multiple view angles totaling 250,000 RGB-D images. The objects are organized into a hierarchical category structure using WordNet hyponym/hypernym relations.

In addition to introducing a large object dataset, we introduce techniques for RGB-D based object recognition and detection and demonstrate that combining color and depth information can substantially improve the results achieved on our dataset. We evaluate our techniques at two levels. *Category level* recognition and detection involves classifying previously unseen objects as belonging in the same category as objects that have previously been seen (*e.g.*, coffee mug). *Instance level* recognition and detection is identifying whether an object is physically the same object that has previously been seen. The ability to recognize and detect objects at both levels is important if we want to use such recognition systems in the context of tasks such as service robotics. For example, identifying an object as a generic coffee mug or as Amelias coffee mug can have different implications depending on the context of the task. In this paper we use the word *instance* to refer to a single object.

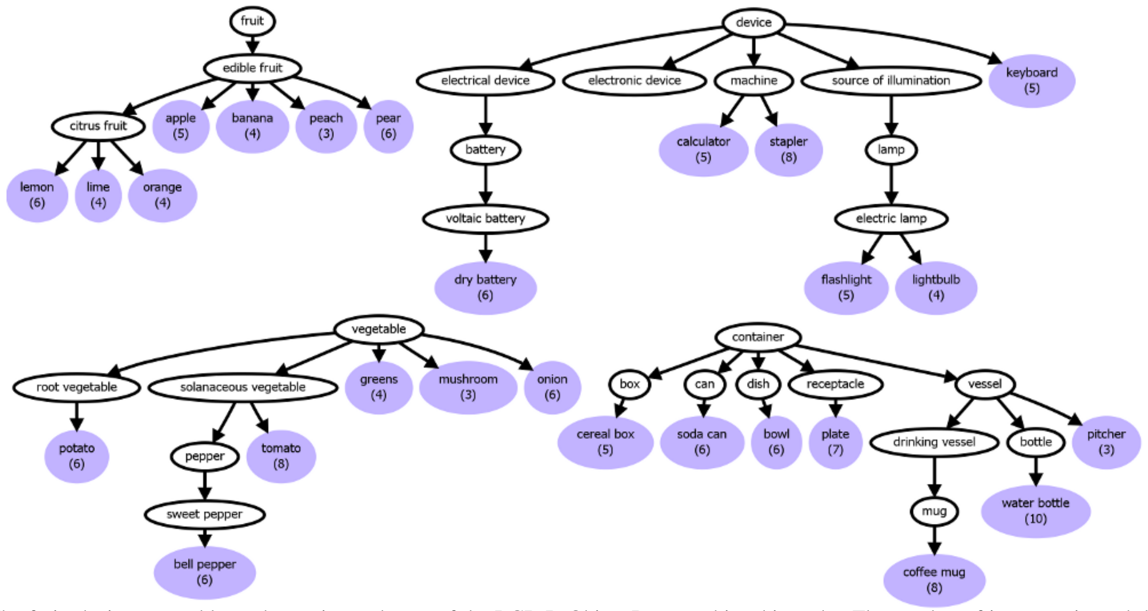

Fig. 2. The fruit, device, vegetable, and container subtrees of the RGB-D Object Dataset object hierarchy. The number of instances in each leaf category (shaded in blue) is given in parentheses.

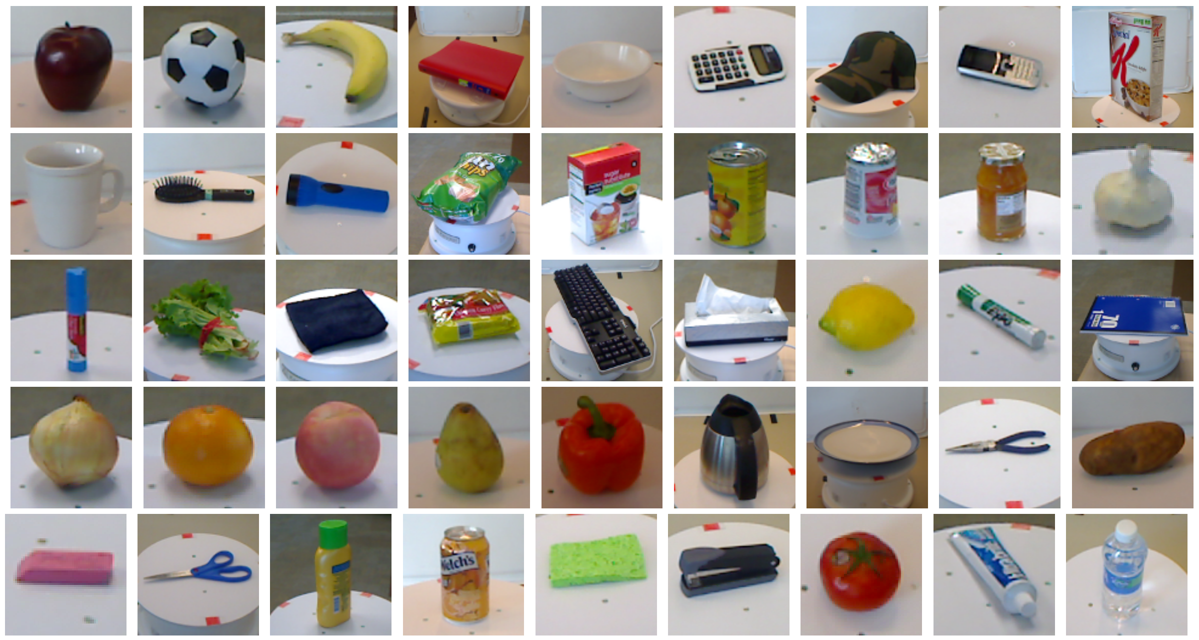

Fig. 3. Objects from the RGB-D Object Dataset. Each object shown here belongs to a different category.

II. RGB-D Object Dataset Collection

The RGB-D Object Dataset contains visual and depth images of 300 physically distinct objects taken from multiple views. The chosen objects are commonly found in home and office environments, where personal robots are expected to operate. Objects are organized into a hierarchy taken from WordNet hypernym/hyponym relations and is a subset of the categories in ImageNet [7]. Fig. 2 shows several subtrees in the object category hierarchy. *Fruit* and *Vegetable* are both top-level subtrees in the hierarchy. *Device* and *Container* are both subtrees under the *Instrumentation* category that covers a very broad range of man-made objects. Each of the 300 objects in the dataset belong to one of the 51 leaf nodes in this hierarchy, with between three to fourteen instances in each category. The leaf nodes are shaded blue in Fig. 2 and the number of object instances in each category is given in parentheses. Fig. 3 shows some example objects from the dataset. Each shown object comes from one of the 51 object categories. Although the background is visible in these images, the dataset also provides segmentation masks (see Fig. 4). The segmentation procedure using combined visual and depth cues is described in Section III.

The dataset is collected using a sensing apparatus consisting of a prototype RGB-D camera manufactured by PrimeSense [23] and a firewire camera from Point Grey Research. The RGB-D camera simultaneously records both color and depth images at 640×480 resolution. In other words, each 'pixel' in an RGB-D frame contains four channels: red, green, blue and depth. The 3D location of each pixel in physical space can be computed using known sensor parameters. The RGB-D camera creates depth images by continuously projecting an invisible infrared structured light pattern and performing stereo triangulation. Compared to passive multi-camera stereo technology, this active projection approach results in much more reliable depth readings, particularly in textureless regions. Fig. 1 (top) shows a single RGB-D frame which consists of both an RGB image and a depth image. Driver software provided with the RGB-D camera ensures that the RGB and depth images are aligned and time-synchronous. In addition to the RGB-D camera, we also recorded data using a Point Grey Research Grasshopper camera mounted above the RGB-D camera, providing RGB images at a higher resolution (1600×1200). The two cameras are calibrated using the *Camera Calibration Toolbox for Matlab* [2]. To synchronize images from the two cameras, we use image timestamps to associate each Grasshopper image with the RGB-D frame that occurs closest in time. The RGB-D camera collects data at 20 Hz, while the Grasshopper has a lower framerate of arond $\sim$ 12 Hz. Fig. 1 (bottom) shows a zoomed-in portion of the bowl showing the difference between the image resolution of the RGB-D camera and the Point Grey Grasshopper.

Using this camera setup, we record video sequences of each object as it is spun around on a turntable at constant speed. The cameras are placed about one meter from the turntable. This is the minimum distance required for the RGB-D camera to return reliable depth readings. Data was recorded with the cameras mounted at three different heights relative to the turntable, at approximately 30°, 45° and 60° above the horizon. One revolution of each object was recorded at each height. Each video sequence is recorded at 20 Hz and contains around 250 frames, giving a total of 250,000 RGB + Depth frames in the RGB-D Object Dataset. The video sequences are all annotated with ground truth object pose angles between $[0, 2\pi]$ by tracking the red markers on the turntable. A reference pose is chosen for each category so that pose angles are consistent across video sequences of objects in a category. For example, all videos of coffee mugs are labeled such that the image where the handle is on the right is 0°.

III. Segmentation

Without any post-processing, a substantial portion of the RGB-D video frames is occupied by the background. We use visual cues, depth cues, and rough knowledge of the configuration between the turntable and camera to produce fully segmented objects from the video sequences.

The first step in segmentation is to remove most of the background by taking only the points within a 3D bounding box where we expect to find the turntable and object, based on the known distance between the turntable and the camera. This prunes most pixels that are far in the background, leaving only the turntable and the object. Using the fact that the object lies above the turntable surface, we can performing RANSAC plane fitting [11] to find the table plane and take points that lie above it to be the object. This procedure gives very good segmentation for many objects in the dataset, but is still problematic for small, dark, transparent, and reflective objects. Due to noise in the depth image, parts of small and thin objects like rubber erasers and markers may get merged into the table during RANSAC plane fitting. Dark, transparent, and reflective objects cause the depth estimation to fail, resulting in pixels that contain only RGB but no depth data. These pixels would be left out of the segmentation if we only used depth cues. Thus, we also apply vision-based background subtraction to generate another segmentation. The top row of Fig. 4 shows several examples of segmentation based on depth. Several objects are correctly segmented, but missing depth readings cause substantial portions of the water bottle, jar and the marker cap to be excluded.

To perform vision-based background subtraction, we applied the adaptive gaussian mixture model of KaewTraKulPong et al. [18] and used the implementation in the OpenCV library. Each pixel in the scene is modeled with a mixture of K gaussians that is updated as the video sequence is played frame-by-frame. The model is adaptive and only depends on a window W of the most recent frames. A pixel in the current frame is classified as foreground if its value is beyond σ standard deviations from all gaussians in the mixture. For our object segmentation we used $K = 2$, $W = 200$, and $\sigma = 2.5$. The middle row of Fig. 4 shows several examples of visual background subtraction. The method is very good at segmenting out the edges of objects and can segment out parts of objects where depth failed to do so. However, it tends to miss the centers of objects that are uniform in color, such as the peach in Fig. 4, and pick up the moving shadows and markers on the turntable.

Since depth-based and vision-based segmentation each excel at segmenting objects under different conditions, we combine the two to generate our final object segmentation. We take the segmentation from depth as the starting point. We then add pixels from the visual segmentation that are not in the background nor on the turntable by checking their depth values. Finally a filter is run on this segmentation mask to remove isolated pixels. The bottom row of Fig. 4 shows the resulting segmentation using combined depth and visual segmentation. The combined procedure provides high quality segmentations for all the objects.

IV. Video Scene Annotation

In addition to the views of objects recorded using the turntable, the RGB-D Object Dataset also includes 8 video sequences of natural scenes. The scenes cover common indoor environments, including office workspaces, meeting rooms, and kitchen areas. The video sequences were recorded by holding the RGB-D camera at approximately human eye-level while walking around in each scene. Each video sequence contains several objects from the RGB-D Object Dataset. The objects are visible from different viewpoints and distances and may be partially or completely occluded

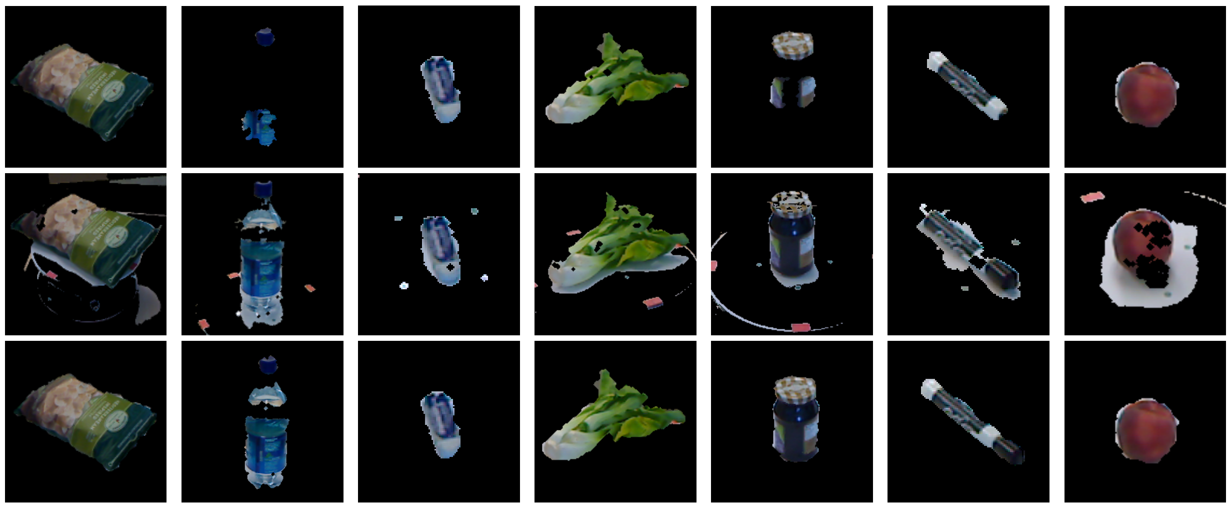
Fig. 4. Segmentation examples, from left to right: bag of chips, water bottle, eraser, leaf vegetable, jar, marker and peach. Segmentation using depth only (top row), visual segmentation via background subtraction (middle row), and combined depth and visual segmentation (bottom row).

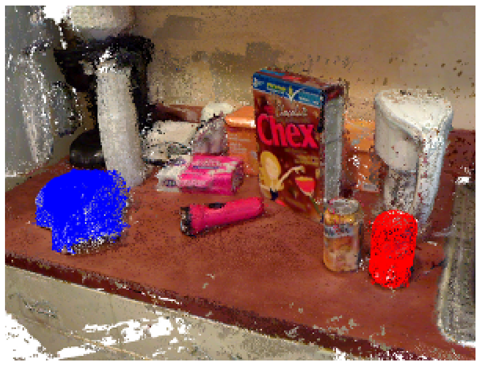
Fig. 5. 3D reconstruction of a kitchen scene with a cap highlighted in blue and a soda can in red using the labeling tool.

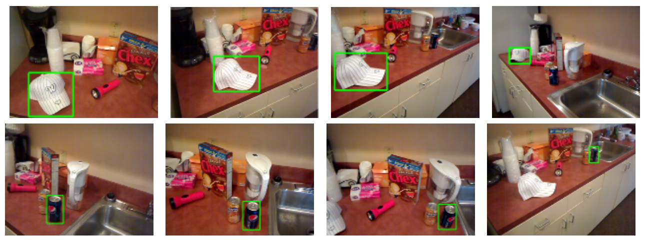
Fig. 6. Ground truth bounding boxes of the cap (top) and soda can (bottom) obtained by labeling the scene reconstruction in Fig. 5.

Video Sequence	# of Frames	# of Objects
Desk_1	1748	3
Desk_2	1949	3
Desk_3	2328	4
Kitchen_small_1	2359	8
Meeting_small_1	3530	13
Table_1	2662	8
Table_small_1	2037	4
Table_small_2	1776	3

Fig. 7. Number of frames and objects in the eight annotated videos of natural scenes in the RGB-D Object Dataset.

in some frames. Fig. 7 summarizes the number of frames and number of objects in each video sequence. In Section VI we demonstrate that the RGB-D Object Dataset can be used as training data for performing object detection in these natural scenes. Here we will first describe how we annotated these natural scenes with the ground truth bounding boxes of objects in the RGB-D Object Dataset. Traditionally, the computer vision community has annotated video sequences one frame at a time. A human must tediously segment out objects in each image using annotation software like the LabelMe annotation tool [24] and more recently, *vatic* [26]. Temporal interpolation across video frames can somewhat alleviate this, but is only effective across a small sequence of frames if the camera trajectory is complex. Crowd-sourcing (e.g. Mechanical Turk) can also shorten annotation time, but does so merely by distributing the work across a larger number of people. We propose an alternative approach. Instead of labeling each video frame, we first stitch together the video sequence to create a 3D reconstruction of the entire scene, while keeping track of the camera pose of each video frame. We label the objects in this 3D reconstruction by hand. Fig. 5 shows the reconstruction of a kitchen scene with a cap labeled in blue and a soda can labeled in red. Finally, the labeled 3D points are projected back into the known camera poses in each video frame and this segmentation can be used to compute an object bounding box. Fig. 6 shows some bounding boxes obtained by projecting the labeled 3D points in Fig. 5 into several video frames.

Our labeling tool uses the technique proposed by Henry et al. [15] to construct 3D scene models from the RGB-D video frames. The RGB-D mapping technique consists of two key components: 1) spatial alignment of consecutive video frames, and 2) globally consistent alignment of the complete video sequence. Successive frames are aligned by jointly optimizing over both appearance and shape matching. Appearance-based alignment is done with RANSAC over SIFT features annotated with 3D position (3D SIFT). Shape-based alignment is performed through Iterative Closest Point

Classifier	Shape	Vision	All
	Category		
LinSVM	53.1 ± 1.7	74.3 ± 3.3	81.9 ± 2.8
kSVM	64.7 ± 2.2	74.5 ± 3.1	83.8 ± 3.5
RF	66.8 ± 2.5	74.7 ± 3.6	79.6 ± 4.0
	Instance (Alternating contiguous frames)		
LinSVM	32.4 ± 0.5	90.9 ± 0.5	90.2 ± 0.6
kSVM	51.2 ± 0.8	91.0 ± 0.5	90.6 ± 0.6
RF	52.7 ± 1.0	90.1 ± 0.8	90.5 ± 0.4
	Instance (Leave-sequence-out)		
LinSVM	32.3	59.3	73.9
kSVM	46.2	60.7	74.8
RF	45.5	59.9	73.1

Fig. 8. Category and instance recognition performance of various classifiers on the RGB-D Object Dataset using shape features, visual features, and with all features. LinSVM is linear SVM, kSVM is gaussian kernel SVM, RF is random forest.

(ICP) using a point-to-plane error metric [5]. The initial alignment from 3D SIFT matching is used to initialize ICP-based alignment. Henry et al. [15] show that this allows the system to handle situations in which only vision or shape alone would fail to generate good alignments. Loop closures are performed by matching video frames against a subset of previously collected frames using 3D SIFT. Globally consistent alignments are generated with TORO, a pose-graph optimization tool developed for robotics SLAM [13].

The overall scene is built using small colored surface patches called *surfels* [22] as opposed to keeping all the raw 3D points. This representation enables efficient reasoning about occlusions and color for each part of the environment, and provides good visualizations of the resulting model. The labeling tool displays the scene in this surfel representation. When the user selects a set of surfels to be labeled as an object, they are projected back into each video frame using transformations computed during the scene reconstruction process. Surfels allow efficient occlusion reasoning to determine whether the labeled object is visible in the frame and if so, a bounding box is generated.

V. Object Recognition Using the RGB-D Object Dataset

The goal of this task is to test whether combining RGB and depth is helpful when the well segmented or cropped object images are available. To the best of our knowledge, the RGB-D Object Dataset presented here is the largest multi-view dataset of objects where both RGB and depth images are provided for each view. To demonstrate the utility of having both RGB and depth information, in this section we present object recognition results on the RGB-D Object dataset using several different classifiers with only shape features, only visual features, and with both shape and visual features.

In object recognition the task is to assign a label (or class) to each query image. The possible labels that can be assigned are known ahead of time. State-of-the-art approaches to tackling this problem are usually supervised learning systems. A set of images are annotated with their ground truth labels and given to a classifier, which learns a model for distinguishing between the different classes. We evaluate object recognition performance at two levels: category recognition and instance recognition. In category level recognition, the system is trained on a set of objects. At test time, the system is presented with an RGB and depth image pair containing an object that was not present in training and the task is to assign a category label to the image (e.g. coffee mug or soda can). In instance level recognition, the system is trained on a subset of views of each object. The task here is to distinguish between object instances (e.g. Pepsi can, Mountain Dew can, or Aquafina water bottle). At test time, the system is presented with an RGB and depth image pair that contains a previously unseen view of one of the objects and must assign an instance label to the image.

We subsampled the turntable data by taking every fifth video frame, giving around 45000 RGB-D images. For category recognition, we randomly leave one object out from each category for testing and train the classifiers on all views of the remaining objects. For instance recognition, we consider two scenarios:

- Alternating contiguous frames: Divide each video into 3 contiguous sequences of equal length. There are 3 heights (videos) for each object, so this gives 9 video sequences for each instance. We randomly select 7 of these for training and test on the remaining 2.
- Leave-sequence-out: Train on the video sequences of each object where the camera is mounted 30° and 60° above the horizon and evaluate on the 45° video sequence.

We average accuracies across 10 trials for category recognition and instance recognition with alternating contiguous frames. There is no randomness in the data split for leave-sequence-out instance recognition so we report numbers for a single trial.

Each image is a view of an object and we extract one set of features capturing the shape of the view and another set capturing the visual appearance. We use state-of-the-art features including spin images [17] from the shape retrieval community and SIFT descriptors [21] from the computer vision community. Shape features are extracted from the 3D locations of each depth pixel in physical space, expressed in the camera coordinate frame. We first compute spin images for a randomly subsampled set of 3D points. Each spin image is centered on a 3D point and captures the spatial distribution of points within its neighborhood. The distribution, captured in a two-dimensional 16×16 histogram, is invariant to rotation about the point normal. We use these spin images to compute efficient match kernel (EMK) features using random fourier sets as proposed in [1]. EMK features are similar to bag-of-words (BOW) features in that they both take a set of local features (here spin images) and generate a fixed length feature vector describing the bag. EMK features approximate the gaussian kernel between local features and gives a continuous measure of similarity. To incorporate spatial information, we divide an axis-aligned bounding cube around each view into a $3 \times 3 \times 3$ grid. We compute a 1000-dimensional EMK feature in each of the 27 cells separately. We perform principal component analysis (PCA) on the EMK features in each cell and take the first 100 components. Finally, we include as shape features the width, depth and height of a 3D bounding box around the view. This gives us a 2703-dimensional shape descriptor.

To capture the visual appearance of a view, we extract SIFT on a dense grid of 8×8 cells. To generate image-level features and capture spatial information we compute EMK features on two image scales. First we compute a 1000-dimensional EMK feature using SIFT descriptors from the entire image. Then we divide the image into a 2×2 grid and compute EMK features separately in each cell from only the SIFT features inside the cell. We perform PCA on each cell and take the first 300 components, giving a 1500-dimensional EMK SIFT feature vector. Additionally, we extract texton histograms [20] features, which capture texture information using oriented gaussian filter responses. The texton vocabulary is built from an independent set of images on LabelMe. Finally, we include a color histogram and also use the mean and standard deviation of each color channel as visual features.

We evaluate the category and instance recognition performance of three state-of-the-art classifiers: linear support vector machine (LinSVM), gaussian kernel support vector machine (kSVM) [8], [4], random forest (RF) [3], [12]. Fig. 8 shows the classification performance of these classifiers using only shape features, only visual features, and using both shape and visual features. Overall visual features are more useful than shape features for both category level and instance level recognition. However, shape features are relatively more useful in category recognition, while visual features are relatively more effective in instance recognition. This is exactly what we should expect, since a particular object instance has a fairly constant visual appearance across views, while objects in the same category can have different texture and color. On the other hand, shape tends to be stable across a category in many cases. The most interesting and significant conclusion is that combining both shape and visual features gives higher overall category-level performance regardless of classification technique. The features compliment each other, which demonstrates the value of a large-scale dataset that can provide both shape and visual information. For alternating contiguous frames instance recognition, using visual features alone already gives very high accuracy, so including shape features does not increase performance. The leave-sequence-out evaluation is much more challenging, and here combining shape and visual features significantly improves accuracy.

We also ran a nearest neighbor classifier under the same experimental setup and using the same set of features and found that it performs much worse than learning-based approaches. For example, its performance on leave-sequence-out instance recognition when using all features is 43.2%, much worse than the accuracies reported in Fig. 8.

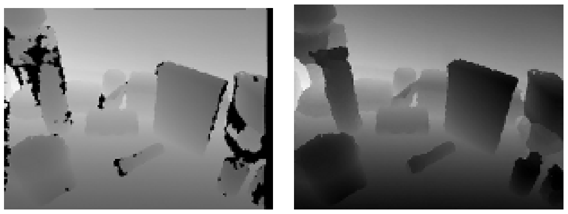

Fig. 9. Original depth image (left) and filtered depth image using a recursive median filter (right). The black pixels in the left image are missing depth values.

VI. Object Detection Using the RGB-D Object Dataset

We now demonstrate how to use the RGB-D OBject Dataset to perform object detection in real-world scenes. Given an image, the object detection task is to identify and localize all objects of interest. Like in object recognition, the objects belong to a fixed set of class labels. The object detection task can also be performed at both the category and the instance level. Our object detection system is based on the standard sliding window approach [6], [10], [14], where the system evaluates a score function for all positions and scales in an image, and thresholds the scores to obtain object bounding boxes. Each detector window is of a fixed size and we search across 20 scales on an image pyramid. For efficiency, we here consider a linear score function (so convolution can be applied for fast evaluation on the image pyramid). We perform non-maximum suppression to remove multiple overlapping detections.

Let H be the feature pyramid and p the position of a subwindow. p is a three-dimensional vector: the first two dimensions is the top-left position of the subwindow and the third one is the scale of the image. Our score function is

$$s_w(p) = w^\top \phi(H, p) + b \tag{1}$$

where w is the filter (weights), b the bias term, and $\phi(H, p)$ the feature vector at position p. We train the filter w using a linear support vector machine (SVM):

$$L(w) = \frac{w^\top w}{2} + C \sum_{i=1}^{N} \max(0, 1 - y_i(w^\top x_i + b)) \tag{2}$$

where N is the training set size, $y_i \in \{-1, 1\}$ the labels, x_i the feature vector over a cropped image, and C the trade-off parameter.

The performance of the classifier heavily depends on the data used to train it. For object detection, there are many potential negative examples. A single image can be used to generate 10^5 negative examples for a sliding window classifier. Therefore, we follow a bootstrapping hard negative mining procedure. The positive examples are object windows we are interested in. The initial negative examples are randomly chosen from background images and object images from other categories/instances. The trained classifier is used to search images and select the false positives with the highest scores (hard negatives). These hard negatives are then added to the negative set and the classifier is retrained. This procedure is repeated 5 times to obtain the final classifier.

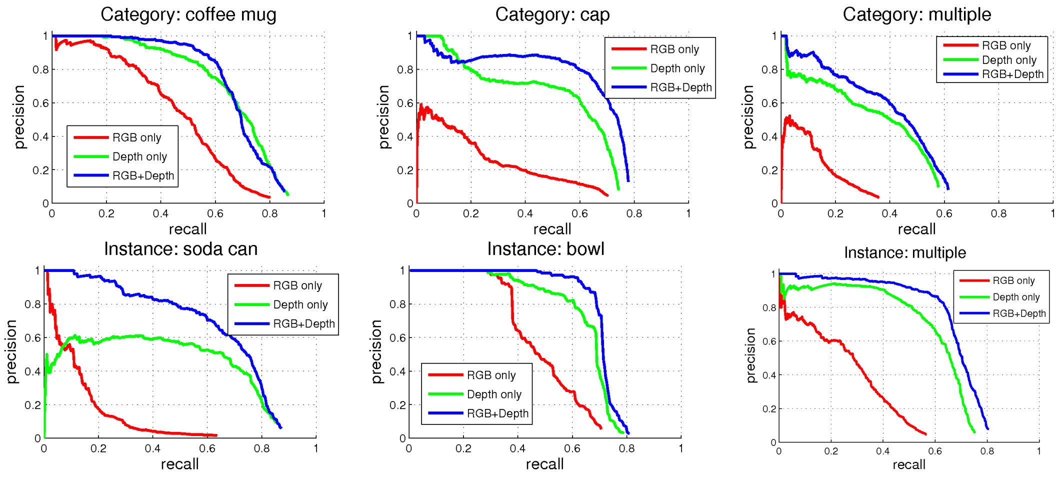

Fig. 10. Precision-recall curves comparing performance with image features only (red), depth features only (green), and both (blue). The top row shows category-level results. From left to right, the first two plots show precision-recall curves for two binary category detectors, while the last plot shows precision-recall curves for the multi-category detector. The bottom row shows instance-level results. From left to right, the first two plots show precision-recall curves for two binary instance detectors, while the last plot shows precision-recall curves for the multi-instance detector.

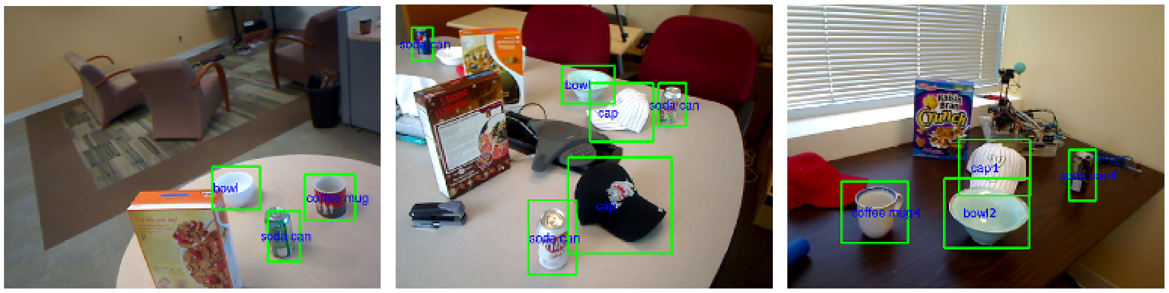

Fig. 11. Three detection results in multi-object scenes. From left to right, the first two images show multi-category detection results, while the last image shows multi-instance detection results.

As features we use a variant of histogram of oriented gradients (HOG) [10], which has been found to work slightly better than the original HOG. This version considers both contrast sensitive and insensitive features, where the gradient orientations in each cell (8×8 pixel grid) are encoded using two different quantization levels into 18 ($0° - 360°$) and 9 orientation bins ($0° - 180°$), respectively. This yields a $4 \times (18 + 9) = 108$-dimensional feature vector. A 31-D analytic projection of the full 108-D feature vectors is used. The first 27 dimensions correspond to different orientation channels (18 contrast sensitive and 9 contrast insensitive). The last 4 dimensions capture the overall gradient energy in four blocks of 2×2 cells.

Aside from HOG over RGB image, we also compute HOG over depth image where each pixel value is the actual object-to-camera distance. Before extracting HOG features, we need to fill up the missing values in the depth image. Since the missing values tend to be grouped together, we here develop a recursive median filter. Instead of considering all neighboring pixel values, we take the median of the non-missing values in a 5×5 grid centered on the current pixel. We apply this median filter recursively until all missing values are filled. An example original depth image and the filtered depth image are shown in Fig. 9.

Finally, we also compute a feature capturing the scale (true size) of the object. We make the observation that the distance d of an object from the camera is inversely proportional to its scale, o. For an image at a particular scale s, we have $c = \frac{o}{s}d$, where c is constant. In the sliding window approach the detector window is fixed, meaning that o is fixed. Hence, $\frac{d}{s}$, which we call the normalized depth, is constant. Since the depth is noisy, we use a histogram of normalized depths over 8×8 grid to capture scale information. For each pixel in a given image, d is fixed, so the normalized depth histogram can choose the correct image scale from the image pyramid. We used a histogram of 20 bins with each bin having a range of 0.15m. Helmer et al. [14] used depth information to define a score function. However, the method of exploiting depth information is very different from our approach: Helmer et al. used depth information as a prior while we construct a scale histogram feature from normalized depth values.

We evaluated the above object detection approach on the 8 natural scene video sequences described in Section IV. Since consecutive frames are very similar, we subsample the video data and run our detection algorithm on every 5th frame. We constructed 4 category-level detectors (bowl, cap, coffee mug,and soda can) and 20 instance-level detectors from the same categories. We follow the PASCAL Visual Object Challenge (VOC) evaluation metric. A candidate detection is considered correct if the size of the intersection of the predicted bounding box and the ground truth bounding box is more than half the size of their union. Only one of multiple successful detections for the same ground truth is considered correct, the rest are considered as false positives. We report precision-recall curves and average precision, which is computed from the precision-recall curve and is an approximation of the area under this curve. For multiple category/instance detections, we pool all candidate detection across categories/instances and images to generate a single precision-recall curve.

In Fig. 10 we show precision-recall curves comparing detection performance with a classifier trained using image features only (red), depth features only (green), and both (blue). We found that depth features (HOG over depth image and normalized depth histograms) are much better than HOG over RGB image. The main reason for this is that in depth images strong gradients are mostly from true object boundaries (see Fig. 9), which leads to much less false positives compared to HOG over RGB image, where color change can also lead to strong gradients. The best performance is attained by combining image and depth features. The combination gives higher precision across all recall levels than image only and depth only, if not comparable. In particular, combining image and depth features gives much higher precision when high recall is desired.

Fig. 11 shows multi-object detection results in three scenes. The leftmost scene contains three objects observed from a viewpoint significantly different than was seen in the training data. The multi-category detector is able to correctly detect all three objects, including a bowl that is partially occluded by a cereal box. The middle scene shows category-level detections in a very cluttered scene with many distracter objects. The system is able to correctly detect all objects except the partially occluded white bowl that is far away from the camera. Notice that the detector is able to identify multiple instances of the same category (caps and soda cans). The rightmost scene shows instance-level detections in a cluttered scene. Here the system was able to correctly detect both the bowl and the cap, even though the cap is partially occluded by the bowl. Our current single-threaded implementation takes approximately 10 seconds to run the four object detectors to label each scene. Both feature extraction over a regular grid and evaluating a sliding window detector are easily parallelizable. We are confident that a GPU-based implementation of the the described approach can perform multi-object detection in real-time.

VII. Discussion

In this paper, we have presented a large-scale, hierarchical multi-view object dataset collected using an RGB-D camera. We have shown that depth information is very helpful for background subtraction, video ground truth annotation via 3D reconstruction, object recognition and object detection. The RGB-D Object Dataset and a set of tools, which are fully integrated into the Robot Operating System (ROS), for accessing and processing the dataset is publicly available at `http://www.cs.washington.edu/rgbd-dataset`.

References

[1] L. Bo and C. Sminchisescu. Efficient Match Kernel between Sets of Features for Visual Recognition. In *Advances in Neural Information Processing Systems (NIPS)*, December 2009.

[2] Jean-Yves Bouguet. Camera calibration toolbox for matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc/`.

[3] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[4] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001.

[5] Y. Chen and M. Gérard. Object modelling by registration of multiple range images. *Image Vision Comput.*, 10(3):145–155, 1992.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[7] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[8] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research (JMLR)*, 9:1871–1874, 2008.

[9] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(4):594–611, 2006.

[10] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[11] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[12] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning (ICML)*, pages 148–156, 1996.

[13] G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard. Estimation of accurate maximum likelihood maps in 3d. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2007.

[14] Scott Helmer and David G. Lowe. Using stereo for object recognition. In *IEEE International Conference on Robotics & Automation (ICRA)*, pages 3121–3127, 2010.

[15] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D Mapping: Using depth cameras for dense 3D modeling of indoor environments. In *the 12th International Symposium on Experimental Robotics (ISER)*, December 2010.

[16] A. Howard and N. Roy. The robotics data set repository (radish), 2003.

[17] A. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 21(5), 1999.

[18] P. Kaewtrakulpong and R. Bowden. An improved adaptive background mixture model for realtime tracking with shadow detection. In *European Workshop on Advanced Video Based Surveillance Systems*, 2001.

[19] Microsoft Kinect. `http://www.xbox.com/en-us/kinect`.

[20] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *Int. J. Comput. Vision*, 43(1):29–44, June 2001.

[21] David G. Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision (ICCV)*, 1999.

[22] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 2000.

[23] PrimeSense. `http://www.primesense.com/`.

[24] B. Russell, K. Torralba, A. Murphy, and W. Freeman. Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1-3), 2008.

[25] S. Savarese and Li Fei-Fei. 3d generic object categorization, localization and pose estimation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.

[26] C. Vondrick, D. Ramanan, and D. Patterson. Efficiently scaling up video annotation with crowdsourced marketplaces. In *European Conference on Computer Vision (ECCV)*, 2010.