

# Chapter 9

## RGB-D Object Recognition: Features, Algorithms, and a Large Scale Benchmark

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. We introduce a large-scale, hierarchical multi-view object dataset collected using an RGB-D camera. The dataset consists of two parts: The RGB-D Object Dataset containing views of 300 objects organized into 51 categories, and the RGB-D Scenes Dataset containing 8 video sequences of office and kitchen environments. The dataset has been made publicly available to the research community so as to enable rapid progress based on this promising technology. We describe the dataset collection procedure and present techniques for RGB-D object recognition and detection of objects in scenes recorded using RGB-D videos, demonstrating that combining color and depth information substantially improves quality of results.

---

K. Lai (✉) · L. Bo · D. Fox

Department of Computer Science & Engineering, University of Washington, Seattle, WA 98195, USA

e-mail: [kevinlai@cs.washington.edu](mailto:kevinlai@cs.washington.edu)

L. Bo

e-mail: [lfb@cs.washington.edu](mailto:lfb@cs.washington.edu)

D. Fox

e-mail: [fox@cs.washington.edu](mailto:fox@cs.washington.edu)

X. Ren

Intel Science and Technology on Pervasive Computing, Seattle, WA 98195, USA

e-mail: [xiaofeng.ren@intel.com](mailto:xiaofeng.ren@intel.com)

## 9.1 Introduction

The availability of public image repositories on the Web, such as Google Images, and visual recognition benchmarks like Caltech 101 [11], LabelMe [36], and ImageNet [9] has enabled rapid progress in visual object recognition in the past decade. 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 [22, 35]. This technology represents an opportunity to dramatically increase the capabilities of robotics object recognition, manipulation, navigation, and interaction. We describe the RGB-D Object Dataset, a large-scale, multi-view object data set collected using an RGB-D camera that was first introduced in [23]. The dataset and its accompanying software has been made publicly available to the research community to enable rapid progress based on this promising technology. The dataset and accompanying software tools are available at <http://www.cs.washington.edu/rgb-d-dataset>.

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. [37], which eight 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. The dataset also includes the RGB-D Scenes Dataset, which eight RGB-D video sequences of office and kitchen environments.

In addition to introducing a large RGB-D object and scene dataset, we also present techniques for object recognition in RGB-D data and detection of objects in scenes recorded using RGB-D videos. We demonstrate that combining color and depth information can substantially improve results on three object recognition tasks: (1) *Category-level* recognition involves classifying previously unseen objects as belonging in the same category as objects that have previously been seen (e.g., coffee mug). (2) *Instance-level* recognition is identifying whether an object is physically the same object that has previously been seen. We use the word *instance* to refer to an object with a particular appearance. (3) *Pose-level* recognition is estimating the orientation of the object relative to the camera. The ability to solve all three recognition tasks is important for applications such as service robotics. For example, identifying an object as a generic “coffee mug” or as “Amelia’s coffee mug” can have different implications depending on the context of the task. Determining the accurate pose of an object is necessary for manipulation.

## 9.2 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 are a subset of the categories in ImageNet [9]. Figure 9.1 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 belongs to one of the 51 leaf nodes in this hierarchy, with between 3 to 14 instances in each category. The leaf nodes are shaded blue in Fig. 9.1 and the number of object instances in each category is given in parentheses. Figure 9.2 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. 9.4). The segmentation procedure using combined visual and depth cues is described in Sect. 9.3.

The dataset is collected using a RGB-D camera manufactured by PrimeSense [35], whose optical hardware is identical to the Microsoft Kinect [22]. The RGB-D camera simultaneously records both color and depth images at  $640 \times 480$  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. Figure 9.3 (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.

Using this camera setup, we record video sequences of each object as it is spun around on a turntable at constant speed. The camera is placed around one meter from the turntable. We found this to be the minimum distance required for the RGB-D camera to return reliable depth readings. Data was recorded with the camera mounted at three different heights relative to the turntable, at approximately  $30^\circ$ ,  $45^\circ$  and  $60^\circ$  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  $[0, 360^\circ]$  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  $0^\circ$ .

### 9.3 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

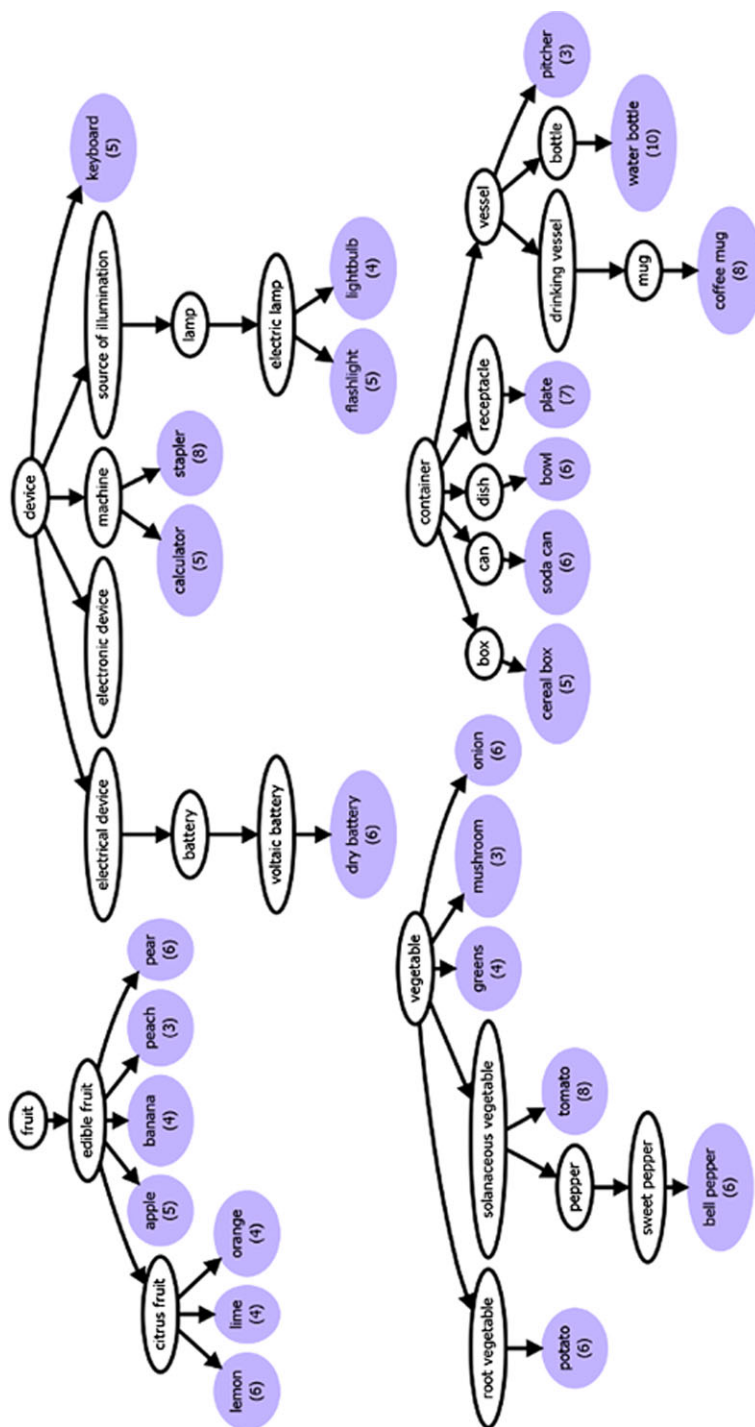
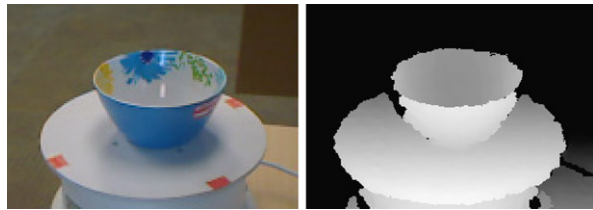


Fig. 9.1 The fruit, device, vegetable, and container subtrees of the RGB-D Object Dataset object hierarchy. The number of instances in each leaf category (shaded in blue) is given in parentheses



**Fig. 9.2** Objects from the RGB-D Object Dataset. Each object shown here is in a different category

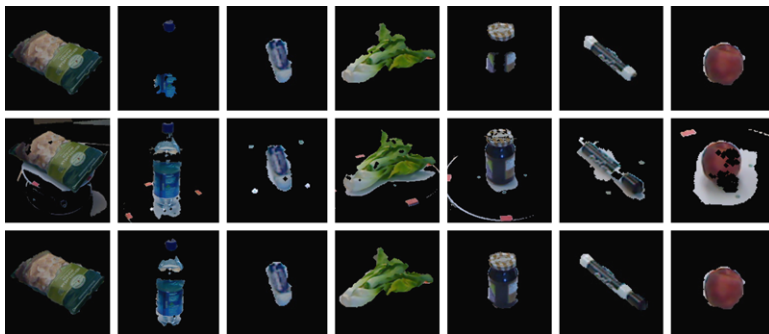
**Fig. 9.3** Each RGB-D frame consists of an RGB image (*left*) and a depth image (*right*)



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 perform RANSAC plane fitting [13] 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. 9.4 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 and Bowden [21], using the implementa-



**Fig. 9.4** 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*)

tion provided by 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  $\sigma$  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. 9.4 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. 9.4, 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 an image erosion filter is run on this segmentation mask to remove isolated pixels. The bottom row of Fig. 9.4 shows the resulting segmentation using combined depth and visual segmentation. The combined procedure provides high quality segmentations for all the objects.

## 9.4 Video Scene Annotation

In addition to the views of objects recorded using the turntable (the RGB-D Object Dataset), we also eight video sequences of natural scenes, which we call the RGB-D Scenes Dataset. 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

**Table 9.1** Number of frames and objects in the eight annotated videos of natural scenes in the RGB-D Scenes Dataset

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. 9.5** (Left) 3D scene reconstruction of a kitchen scene with a cap highlighted in blue and a soda can in red using the labeling tool. (Right) Ground truth bounding boxes of the cap (top) and soda can (bottom) obtained by labeling the reconstruction

the RGB-D Object Dataset. The objects are visible from different viewpoints and distances and may be partially or completely occluded in some frames. Table 9.1 summarizes the number of frames and number of objects in each video sequence. In Sect. 9.6 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 [36] and more recently, *vatic* [39]. 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. Amazon 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. Figure 9.5 (left) 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. Figure 9.5 (right) shows some bounding boxes obtained by projecting the labeled 3D points into several video frames.

Our labeling tool uses the technique proposed by Henry et al. [19] to reconstruct 3D scenes 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 (ICP) using a point-to-plane error metric [7]. The initial alignment from 3D SIFT matching is used to initialize ICP-based alignment. Henry et al. [19] 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 [17].

The overall scene is built using small colored surface patches called *surfels* [34] 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.

## 9.5 RGB-D Object Recognition

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 on two tasks: category recognition and instance recognition. In category 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 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.

Two important problems to address for object recognition using RGB-D cameras are designing the appropriate feature representation for RGB-D data, and devising



the appropriate classification method. In Sect. 9.5.1 we describe the experimental setup for using the RGB-D Object Dataset to evaluate recognition techniques. We then describe our work on distance learning in Sect. 9.5.2, demonstrating that it outperforms existing state-of-the-art classifiers. In Sect. 9.5.3, we present Kernel Descriptors, a novel family of features for RGB-D Object Recognition, and show that it outperforms existing state-of-the-art features for image and 3D point cloud recognition. Finally, in Sect. 9.5.4 we present a technique for efficiently performing RGB-D object recognition and pose estimation jointly.

### 9.5.1 *Experimental Setup*

The experiments performed in this section use the turntable data containing cropped and segmented views of objects in the RGB-D Object Dataset. The video sequences in the RGB-D Scenes Dataset are not used. 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 three contiguous sequences of equal length. There are three heights (videos) for each object, so this gives nine video sequences for each instance. We randomly select seven of these for training and test on the remaining two.
- Leave-sequence-out: Train on the video sequences of each object where the camera is mounted  $30^\circ$  and  $60^\circ$  above the horizon and evaluate on the  $45^\circ$  video sequence.

We do not make use of the WordNet organization of the dataset in our experiments. 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.

### 9.5.2 *Distance Learning for RGB-D Object Recognition*

In its simplest form, nearest neighbor classifiers place a set of examples with known labels in a euclidean feature space and a test example is classified based on the labels of the  $k$  nearest known examples (the  $k$ -nearest neighbor classifier). The idea behind distance learning is that nearest neighbor classification can be improved by learning a distance function because euclidean feature distances may not be the best measure of similarity, particularly when different types of feature are combined [40, 41]. Instead of learning a single global distance metric, recently researchers have looked into local distance learning, which learns different distance functions for different

regions of the feature space [38]. Local distance learning has been extensively studied and demonstrated for object recognition, both for color images [15, 16, 31] and 3D shapes [27]. A key property of these approaches is that they are non-parametric, meaning that they can learn decision boundaries whose shape and complexity is determined by the data.

### 9.5.2.1 Instance Distance Learning

We proposed instance distance learning in [25]. Many existing distance learning approaches for image classification are designed for recognition of image collections on the web, such as Flickr and Google Images. In these applications it is impossible to tell whether two images in the collection are of the exact same object. In contrast, there are applications, particularly in robotics, where the data come from a known set of objects and consist of a collection of views taken from different camera positions, as is the case in the RGB-D Object Dataset. Our proposed approach exploits this structure by learning a distance function for each object instance. An object instance is an object with a particular appearance, and we assume that we have a collection of views of each object. We learn view-to-instance distances that measure the similarity between a query view of an object and each of the 300 object instances in the RGB-D Object Dataset.

Given a set of  $M$  features, let  $\mathbf{d}(x, y)$  be the  $M$ -dimensional vector of  $L_2$  distances between two views  $x$  and  $y$  computed for each feature separately. We define the distance between a view  $x$  and an instance  $Y$  as the weighted average of feature distances from  $x$  to all the views  $y$  that constitute instance  $Y$ :

$$f(x, Y) = \frac{1}{|Y|} \sum_{y \in Y} \mathbf{w}_y^\top \mathbf{d}(x, y) + b \quad (9.1)$$

where  $\mathbf{w}_y$ , the vector of weights for  $y$ , and  $b$ , the bias term, are learned parameters. Classification of a test view is performed by computing the above distance to every object instance in the training set and taking the label of the nearest neighbor.

Parameter learning is formulated as a convex optimization problem with a margin-based loss function. Group-Lasso regularization [33] is used to encourage sparsity across views by encouraging  $\mathbf{w}_y$  to not be the zero vector for only a small subset of  $y \in Y$ . In other words, the approach can, via supervised learning, choose to retain a small subset of views that provide good coverage of the visual variation of each object. On the RGB-D Object Dataset, instance distance learning can learn distance functions that depend only on 30 % of the views in the training data without compromising classification accuracy, outperforming uniform random downsampling at equal levels of data sparsification [25].

### 9.5.2.2 RGB-D Feature Set

To evaluate Instance Distance Learning we used existing state-of-the-art features developed separately for RGB images and 3D point clouds.

*Shape features* are extracted from 3D point clouds of each view. The 3D point cloud is obtained from the depth image using known camera intrinsic parameters. We first compute spin images [20] for a randomly subsampled set of 3D points. We generate a feature vector for the view using this collection of local descriptors using efficient match kernels (EMK) [4]. To incorporate spatial information, we divide each view using a  $3 \times 3 \times 3$  axis-aligned bounding cube. We compute a 1000-dimensional EMK features 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. We also include as shape features the width, depth and height of the bounding cube. For SVM and random forest classifiers, we concatenate these features to yield a 2703-dimensional shape descriptor. For distance learning approaches, we compute euclidean feature distances for each of the 27 spin image cells, as well as each dimension of the bounding box separately, meaning we learn weighted distances over 30 shape features.

*Visual features* are extracted from the RGB image to capture the appearance of a view. We extract SIFT descriptors [30] densely on an  $8 \times 8$  grid. To generate image-level features we use EMK on a two-level spatial pyramid: First we compute a 1000-dimensional EMK feature from the entire image. Then we divide the image into  $2 \times 2$  blocks and extract EMK features in each block. We perform PCA on each block and take the first 300 components, giving a 1500-dimensional EMK SIFT descriptor. We also extract texton histogram [28] features. We used 100 textons learned from images on LabelMe and computed histograms five image regions, as was done in [31]. We also include a color histogram (11 bins three color channels) and the mean and standard deviation color. For SVM and random forest classifiers, we concatenate all the features to yield a 2039-dimensional visual descriptor. For distance learning approaches, we compute euclidean feature distances for each of five spatial pyramid blocks of SIFT features, five image regions of texton histograms, the color histogram, the mean color, and the color standard deviation separately. Hence, we learn weighted distances over 13 visual features.

### 9.5.2.3 Evaluation

We compared instance distance learning with the exemplar-based distance learning approach of [31]. This approach learns a distance function independently for each view of an object, i.e. an exemplar. Classification of a test view is performed by computing distances to every view in the training set and taking the label of the nearest neighbor. We also compared with three standard classifiers including linear and gaussian kernel support vector machine (LinSVM and kSVM) [6, 10], and random forests(RF) [5, 14]. We compared the performance of the proposed instance distance learning approach and alternative classification methods on two recognition tasks: Table 9.2 shows the classification accuracy on category recognition, and Table 9.3 for shows the accuracy on instance recognition with alternating contiguous frames (refer to Sect. 9.5.1 for a description of the different experimental setups). We report classification accuracies when using only shape features, only visual features,

**Table 9.2** Category recognition on leave-out objects. Performance of various classification methods on the RGB-D Object Dataset using shape features, visual features, and with all features

Method	Category recognition		
	Shape	Vision	All
LinSVM	53.1 $\pm$ 1.7	74.3 $\pm$ 3.3	81.9 $\pm$ 2.8
kSVM	64.7 $\pm$ 2.2	74.5 $\pm$ 3.1	83.8 $\pm$ 3.5
RF	66.8 $\pm$ 2.5	74.7 $\pm$ 3.6	79.6 $\pm$ 4.0
EBLocal [31]	58.9 $\pm$ 2.1	70.1 $\pm$ 3.4	78.4 $\pm$ 2.8
IDL [25]	70.2 $\pm$ 2.0	78.6 $\pm$ 3.1	85.4 $\pm$ 3.2

**Table 9.3** Instance recognition on alternating contiguous frames. Performance of various classifiers on the RGB-D Object Dataset using shape features, visual features, and with all features

Method	Instance recognition (alternating contiguous frames)		
	Shape	Vision	All
LinSVM	32.4 $\pm$ 0.5	90.9 $\pm$ 0.5	90.2 $\pm$ 0.6
kSVM	51.2 $\pm$ 0.8	91.0 $\pm$ 0.5	90.6 $\pm$ 0.6
RF	52.7 $\pm$ 1.0	90.1 $\pm$ 0.8	90.5 $\pm$ 0.4
EBLocal [31]	41.2 $\pm$ 0.6	81.2 $\pm$ 0.6	84.5 $\pm$ 0.5
IDL [25]	54.8 $\pm$ 0.6	89.8 $\pm$ 0.2	91.3 $\pm$ 0.3

**Table 9.4** Instance recognition on left-out sequence. Performance of various classifiers on the RGB-D Object Dataset using shape features, visual features, and with all features

Method	Instance recognition (leave-sequence-out)		
	Shape	Vision	All
LinSVM	32.3	59.3	73.9
kSVM	46.2	60.7	74.8
RF	45.5	59.9	73.1

and using both shape and visual features. From the results, we see that the proposed instance distance learning outperforms per-exemplar distance learning and the three standard classifiers on both category and instance recognition.

Aside from comparing the instance distance learning with existing classification methods, we also investigated the usefulness of shape and visual features for category and instance recognition. For this we also report in Table 9.4 the performance of the three standard classifiers on instance recognition on left-out video sequences. We find that regardless of the classification technique used, the chosen visual features are more useful than shape features for both category and instance recognition (see Tables 9.2, 9.3, and 9.4). However, shape features are relatively more useful in category recognition, while visual features are relatively more effective in in-

stance recognition. This is because 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 performance regardless of classification technique. The features complement 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.

### ***9.5.3 Kernel Descriptors for RGB-D Object Recognition***

The core of building a robust object recognition system is to extract underlying representations (features) from high-dimensional sensor data such as images, depths and 3D point clouds. Given the wide availability of RGB-D cameras, it is an open question what is the best way to extract features over RGB-D images. The standard approach to object recognition is to compute pixel attributes in small windows around (a subset of) pixels. For example, in SIFT [29], gradient orientation and magnitude attributes are computed from  $5 \times 5$  image windows. Another example is Spin Images [20] over local 3D point clouds. A key question for object recognition is then how to measure the similarity of local patches based on the attributes of pixels within them, because this similarity measure is used in classifiers such as linear support vector machines (SVM). Techniques based on histogram features, such as SIFT and Spin Images, discretize individual pixel attribute values into bins and then compute a histogram over the discrete attribute values within a patch. The similarity between two patches can then be computed based on their histograms. Unfortunately, the binning restricts the similarity measure and introduces quantization errors, which limit the accuracy of recognition.

#### **9.5.3.1 Kernel Descriptors**

Kernel descriptors, which we proposed in [1–3], aim to discover underlying representations of RGB-D sensor data using machine learning methodology. We highlight the kernel view of SIFT and Spin Images, and show that histogram features are a special, rather restricted case of efficient match kernels. This novel insight allows us to design a family of kernel descriptors. Kernel descriptors avoid the need for pixel attribute discretization and are able to turn any pixel attribute into compact patch-level features. Here, the similarity between two patches is based on a kernel function, called the match kernel, that averages over continuous similarities between all pairs of pixel attributes in the two patches. Match kernels are extremely

flexible and it is easy to incorporate domain knowledge, since the similarity measure between pixel attributes can be any positive definite kernel, such as the popular Gaussian kernel function. While match kernels provide a natural similarity measure for image patches, evaluating these kernels can be computationally expensive, in particular for large image patches. To compute kernel descriptors, one has to move to the feature space forming the kernel function. Unfortunately, the dimensionality of these feature vectors is high, or even infinite, if for instance a Gaussian kernel is used. Thus, for computational efficiency and for representational convenience, we reduce the dimensionality by projecting the high/infinite dimensional feature vector to a set of finite basis vectors using kernel principal component analysis. This procedure can approximate the original match kernels very well, as shown in [1–3].

As an example, we briefly describe the gradient kernel descriptors over depth patches. We treat depth images as grayscale images and compute gradients at pixels. The gradient kernel descriptors  $F_{\text{grad}}$  is constructed from the pixel gradient similarity function  $k_o$

$$F_{\text{grad}}^t(Z) = \sum_{i=1}^{d_o} \sum_{j=1}^{d_s} \alpha_{ij}^t \left\{ \sum_{z \in Z} \tilde{m}_z k_o(\tilde{\theta}_z, p_i) k_s(z, q_j) \right\} \quad (9.2)$$

where  $Z$  is a depth patch, and  $z \in Z$  are the 2D relative position of a pixel in a depth patch (normalized to  $[0, 1]$ ).  $\tilde{\theta}_z$  and  $\tilde{m}_z$  are the normalized orientation and magnitude of the depth gradient at a pixel  $z$ . The orientation kernel  $k_o(\tilde{\theta}_z, \tilde{\theta}_x) = \exp(-\gamma_o \|\tilde{\theta}_z - \tilde{\theta}_x\|^2)$  computes the similarity of gradient orientations. The position Gaussian kernel  $k_s(z, x) = \exp(-\gamma_s \|z - x\|^2)$  measures how close two pixels are spatially.  $\{p_i\}_{i=1}^{d_o}$  and  $\{q_j\}_{j=1}^{d_s}$  are uniformly sampled from their support region,  $d_o$  and  $d_s$  are the numbers of sampled basis vectors for the orientation and position kernels.  $\alpha_{ij}^t$  are projection coefficients computed using kernel principal component analysis. Other kernel descriptors are constructed in a similar fashion from pixel-level similarity functions (see [2] and [3] for details).

To summarize, extracting kernel descriptors involves the following steps: (1) define pixel attributes; (2) design match kernels to measure the similarities of image patches based on these pixel attributes; (3) determine approximate, low dimensional match kernels. While the third step is done automatically by learning low dimensional representations and the defined kernels, while the first two steps allow the user to tune the approach for specific scenarios and application. Thus, kernel descriptors provides a unified and principled framework for extracting rich features from sensor data. We have developed eight types of kernel descriptor [1–3] for RGB-D images; a relatively complete feature set to capture rich cues for robust object recognition. Kernel descriptors outperform state-of-the-art recognition algorithms on many benchmarks, including USPS, extended Yaleface, Scene-15, Caltech-101, CIFAR-10, CIFAR-10-ImageNet, and the RGB-D Object Dataset. More importantly, the features have exhibited very robust performance in several real-world recognition systems, including the autonomous chess playing manipulator robot [32] and the object-aware situated interactive sys-

tem (OASIS) [24]. The source code for RGB-D kernel descriptors is available at <http://www.cs.washington.edu/rgb-d-dataset/software.html>.

### 9.5.3.2 Evaluation

We evaluated the proposed kernel descriptors on the RGB-D Object Dataset for both category and instance recognition (leave-sequence-out). For each RGB-D image, we compute seven kernel descriptors on dense regular grids: gradient kernel descriptors (GradKDES) over image and depth patches, local binary pattern kernel descriptors (LBPKDES) over image and depth patches, normalized RGB kernel descriptors (NRGBKDES) over image patches, spin kernel descriptors (SpinKDES) and size kernel descriptors (SizeKDES) over 3D point clouds. NRGBKDES is a variant of RGB kernel descriptors [2] that normalizes RGB values by subtracting the mean and dividing by the standard deviation in order to be robust to lighting condition changes. We extract gradient, local binary pattern, and normalized RGB kernel descriptors on  $16 \times 16$  depth or image patches with spacing of eight pixels. For size kernel descriptors, we consider the whole point cloud and subsample the number of 3D points to be no more than 200 for each interest point. For spin kernel descriptors, we set the radius of the local region around interest points to be 4 cm and again subsample the number of neighboring points to be no larger than 200. We consider  $1 \times 1$ ,  $2 \times 2$  and  $4 \times 4$  pyramid sub-regions and form object-level features using EMK [4] with 1000 basis vectors learned by K-means on about 500,000 kernel descriptors sampled from training data. The dimensionality per kernel descriptor is  $(1 + 4 + 16) \times 1000 = 21000$ . The total feature extraction time per kernel descriptor is around 0.2 seconds using unoptimized MATLAB code. We train linear SVMs for recognition, which our experiments suggest are sufficient for good accuracy when using kernel descriptors.

We report the results of RGB-D kernel descriptors in Table 9.5. Results from using the existing shape and visual feature set (FeaSet, see Sect. 9.5.2) is also repeated here for comparison. First of all, we observe that combining all kernel descriptors performs much better than the best single kernel descriptors for both category and instance recognition. There are two reasons at least. Firstly, RGB-D kernel descriptors capture different recognition cues of objects including shape, color and size, which are strong in their own right and complement each other. The weights learned by the linear SVM using label information can automatically balance the contribution of each kernel descriptors for a specific task. The results in Table 9.5 show that RGB-D kernel descriptors significantly outperform the set of existing shape and visual features used in Sect. 9.5.2.

For category recognition, we observe in Table 9.5 that the best single kernel descriptor is gradient over RGB images, achieving 77.7 % accuracy. Combining depth and image kernel descriptors achieves 86.3 %, much higher than that obtained by gradient kernel descriptors only. We observe that the performance of depth kernel descriptors is comparable with image kernel descriptors, indicating that depth information is as important as visual information for category recognition. For instance

**Table 9.5** Category recognition (leave-objects-out) and instance recognition (leave-sequence-out). Performance of each kernel descriptor and their combination is reported. Results using the feature set in Sect. 9.5.2 is repeated here for comparison. The classifier is linear SVM in all cases

Features	Category	Instance
FeaSet (RGB)	$74.3 \pm 3.3$	59.3
FeaSet (Cloud)	$53.1 \pm 1.7$	32.3
FeaSet Combination	$81.9 \pm 2.8$	73.9
GradKDES (RGB)	$77.7 \pm 2.7$	82.2
LBPKDES (RGB)	$77.5 \pm 2.5$	80.9
NRGBKDES (RGB)	$64.7 \pm 3.3$	83.4
GradKDES (Depth)	$72.8 \pm 2.4$	40.1
LBPKDES (Depth)	$72.1 \pm 2.1$	33.5
SpinKDES (Cloud)	$60.2 \pm 2.5$	33.1
SizeKDES (Cloud)	$56.3 \pm 3.5$	25.2
Combination of all KDES	$86.5 \pm 1.0$	91.2

recognition, we observe in Table 9.5 that the best single feature is the normalized RGB kernel descriptor (83.4 %). Combining depth and image kernel descriptors achieves 91.2 %, substantially better than that obtained by normalized RGB kernel descriptors. We also notice that depth features are much worse than image features in the context of instance recognition. This is not very surprising since the different instances in the same category could share very similar shape.

### 9.5.4 Joint Object Category, Instance, and Pose Recognition

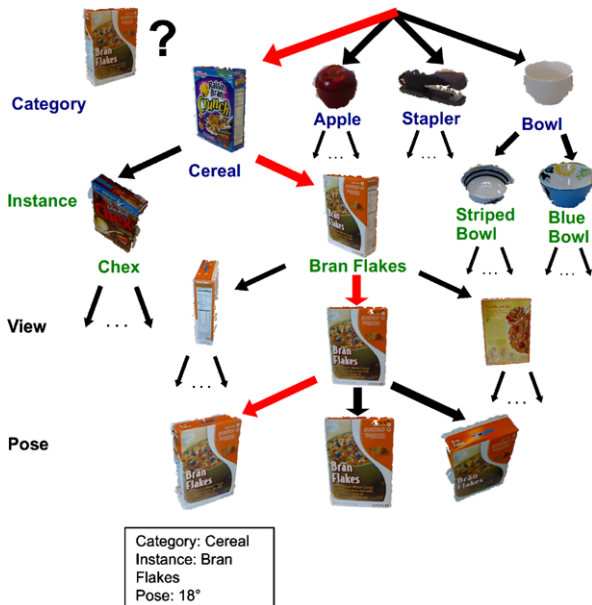
Object perception has multiple levels of semantics. When an autonomous robot encounters an object, we may want it to answer any or all of the following questions: *Is this a coffee mug or a plate?* (**category** recognition); *Is this Alice’s coffee mug or Bob’s coffee mug?* (**instance** recognition); *Am I looking at the mug with the handle facing left or right?* (**pose** recognition or approximate pose estimation). Although it is clear that category, instance, and pose recognition are closely connected and multiple facets of a single object perception problem, they have traditionally been studied in different contexts and solved using different techniques.

#### 9.5.4.1 Object-Pose Tree

We investigated a technique, called the Object-Pose Tree, for simultaneously addressing three object recognition tasks: category recognition, instance recognition, and pose estimation [24]. These three object recognition tasks form a tree as naturally defined by the semantic structures: a category covers multiple object instances,



**Fig. 9.6** Recognition of a box of Bran Flakes cereal using the Object-Pose Tree. The system labels the test image by starting with the category level *at the top* and traversing down the tree to the instance, view, and finally pose level *at the bottom*. The system finds the most similar (but not identical) pose in the training set



an instance covers multiple (discrete) “views”, and each view is a collection of (continuous) object poses (see Fig. 9.6). Each node in the tree is a linear decision function. Given a test image containing a cropped and segmented view of an object, the system first evaluates the score that each node at the first (category) layer assigns to the image using the corresponding linear functions. The test image proceeds down the node with the highest score, and scores are computed for nodes in the second (instance) layer under that subtree. This process is repeated until the image reaches a leaf node, which represents one example in the training set (a view of an object in a particular pose). The system then assigns the category, instance, and pose based on the path traced by the test image.

Parameter learning of the entire Object-Pose Tree is formulated as structured SVM learning, where the path traced by the test image is the structured output of the system. A single convex objective function is defined that takes into account a margin-based loss on all three recognition tasks. This objective function is optimized using stochastic gradient descent. The learning procedure is detailed in [24].

### 9.5.4.2 Evaluation

We evaluated the Object-Pose Tree on the RGB-D Object Dataset, which annotates the pose of every view of every object as the angle about the vertical axis. Each object category has a canonical pose that is labeled as  $0^\circ$ , and every image in the dataset is labeled with a pose in  $[0, 360^\circ]$ . As features, we use gradient and shape (local binary pattern) kernel descriptors [2] extracted over both RGB and depth images. We use the leave-sequence-out procedure for train/test data split: the tree is

**Table 9.6** Category, instance, and pose recognition results using the SVM Tree and the Object-Pose Tree

Approach	Category accuracy	Instance accuracy	Pose error
SVM Tree	92.0	77.4	44.6°
Object-Pose Tree	94.3	78.4	30.2°

**Fig. 9.7** Recognition results from the Object-Pose Tree for two objects: *Red Mug* (top left), and *Ultrabrite Toothpaste* (top right). (Bottom) From left to right, the top five objects with the highest classifier response at the instance level and at the pose level for *Red Mug* and *Ultrabrite toothpaste*



trained on views of all 300 objects at 30° and 60° with the horizon (not to be confused with the pose angle about the vertical axis that the system is to estimate), and evaluated on views taken at 45° with the horizon.

Table 9.6 shows results from using the Object-Pose Tree and from using a tree of SVM classifiers. The SVM Tree has the same structure as the Object-Pose Tree, but the parameters are learned differently. For the SVM Tree, we first train a multi-class linear SVM for the category layer. Then we train SVMs for distinguishing instances within each category separately. We repeat this procedure down to the view and pose layers. We report category and instance recognition accuracies, as well as the median pose error for cases where the object instance is correctly identified. The results show that the Object-Pose Tree outperforms the SVM Tree, demonstrating that learning the parameters of the entire tree jointly through a single objective function is better than learning for each task independently. Figure 9.7 shows recognition results on two images using the Object-Pose Tree, a red mug and an Ultrabrite toothpaste. For each image, the top five matching instances and poses are shown.

## 9.6 Object Detection in Scenes Using RGB-D Cameras

In Sect. 9.5 we demonstrated how to perform RGB-D object recognition, where images are already cropped and segmented so that they contain only one object. In

some applications such segmentations may not be easy to obtain. In this section, we demonstrate how to use the RGB-D Object Dataset to perform object detection in real-world scenes that can contain multiple objects. 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. Existing work generally localizes objects to one of two levels of granularity: (1) localize the object to a rectangular subregion of the image (bounding box), and (2) assign an object label to every pixel or 3D point, leading to a pixel/point-level segmentation of the scene.

We present approaches for detecting objects at both levels of granularity. Experiments are performed where object detectors are trained using views of objects in the RGB-D Object Dataset, and evaluated on scenes in the RGB-D Scenes Dataset. During training, videos in the RGB-D Scenes Dataset that contain the objects are not used. A set of videos of taken in similar office and kitchen environments but without the presence of objects in the RGB-D Object Dataset are used for sampling negative training examples. These videos are available as “background” scenes in the RGB-D Scenes Dataset.

### 9.6.1 RGB-D Object Detection

Our RGB-D object detection system uses sliding window detectors [8, 12, 18], 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

$$s_w(p) = w^\top \phi(H, p) + b \quad (9.3)$$

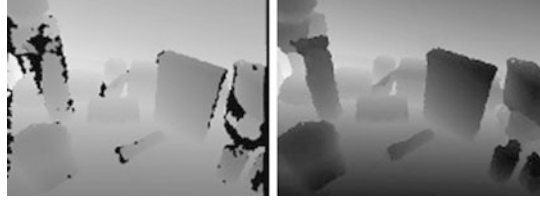
where  $w$  is the filter (weights),  $b$  the bias term, and  $\phi(H, p)$  the feature vector at position  $p$ . We train the filter  $w$  using a linear support vector machine (SVM):

$$L(w) = \frac{w^\top w}{2} + C \sum_{i=1}^N \max(0, 1 - y_i(w^\top x_i + b)) \quad (9.4)$$

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

**Fig. 9.8** 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



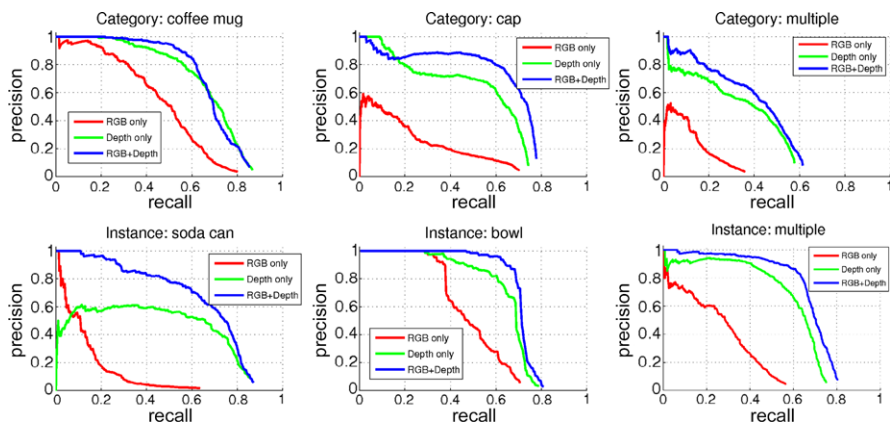
used to generate  $10^5$  negative examples for a sliding window classifier. Therefore, we follow a bootstrapping hard negative mining procedure. The positive examples are object windows we are interested in. The initial negative examples are randomly chosen from background images and object images from 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 five times to obtain the final classifier.

As features we use a variant of histogram of oriented gradients (HOG) proposed in [12]. The gradient orientations in each cell ( $8 \times 8$  pixel grid) are encoded using two different quantization levels into 18 ( $0^\circ$ – $360^\circ$ ) and nine orientation bins ( $0^\circ$ – $180^\circ$ ), 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 [12].

Aside from HOG over RGB image, we also compute HOG over depth image where each pixel value is the object-to-camera distance. Before extracting HOG features, we need to fill up missing values in the depth image. Since missing values tend to be grouped together, we use a recursive median filter. Instead of considering all neighboring pixel values, we take the median of the non-missing values in a  $5 \times 5$  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. 9.8.

Finally, we also compute a feature capturing the scale (true size) of the object. Observe 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. For sliding window detection 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  $8 \times 8$  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 0.15 m. Helmer et al. [18] also used depth information, but they used it as a prior in their probabilistic model while we construct a scale histogram feature from normalized depth values.

We evaluated RGB-D object detection on eight natural scene video sequences described in Sect. 9.4. Since consecutive frames are very similar, we subsample the video data and run our detection algorithm on every 5th frame. We constructed four category detectors (bowl, cap, coffee mug, and soda can) and 20 instance detectors from the same categories. We follow the PASCAL Visual Object Challenge (VOC)

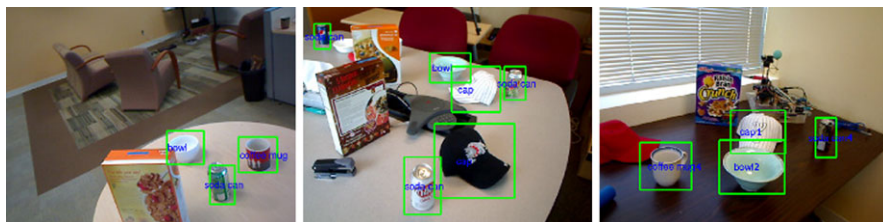


**Fig. 9.9** Precision-recall curves comparing performance with image features only (*red*), depth features only (*green*), and both (*blue*). The *top row* shows category-level results. *From left to right, the first two plots* show precision-recall curves for two binary category detectors, while the *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

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. 9.9 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. 9.8), 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.

Figure 9.10 shows multi-object detection results in three scenes. The leftmost scene contains three objects observed from a viewpoint significantly different from what 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 detections in a very cluttered scene with many distracting objects. The system is able to correctly detect all objects except the par-



**Fig. 9.10** 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*

tially 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 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 4 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 described approach can perform multi-object detection in real-time.

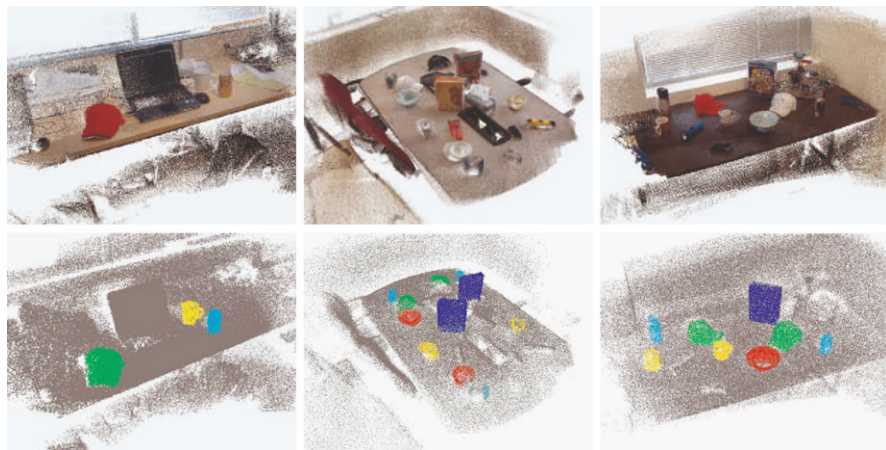
## 9.6.2 Scene Labeling

For robotics applications such as object grasping, localizing objects with a bounding box is not enough. Instead, a pixel-level classification is needed, which provides both recognition and segmentation of objects in the scene. In [26], we presented a technique for pixel-level object labeling in 3D scenes reconstructed from RGB-D videos. To do this, we used sliding window detectors to assign a class probability to every pixel. Evidence is aggregated over multiple video frames by transforming the pixels into points in a 3D scene. The transformation is computed based on camera poses estimated using the RGB-D Mapping algorithm [19]. The scene is voxelized and a Markov Random Field (MRF) over the voxels that combines cues from view-based detection and 3D geometry is used to obtain the final object labeling.

We evaluated our object labeling approach on labeling five object categories in the RGB-D Scenes Dataset, bowls, caps, cereal boxes, coffee mugs, and soda cans. We achieve an overall F-score of 89.8 % when evaluated on all eight scenes in the dataset, where equal weight is assigned to each of the five object categories and to the background class. Table 9.7 shows the per-category and overall precisions and recalls of the proposed approach (Det3DMRF), as well as picking the label of each point uniformly at random (Random). The precisions for Random is 16.7 % for each of the six classes as expected, while the recalls show that the vast majority (87.5 %) of points in our scenes is background. The proposed approach performs consistently well for six classes, achieving close to overall 90 % precision and recall.

**Table 9.7** Per-category and overall (macro-averaged across categories) precisions and recalls for the proposed detection-based 3D scene labeling approach and for random labeling. Our approach works very well for all object categories in the RGB-D Scene Dataset

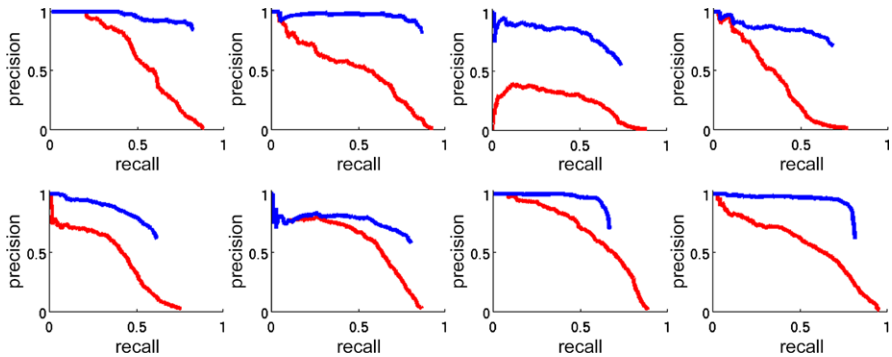
Technique	Precision/recall						
	Bowl	Cap	Cereal box	Coffee mug	Soda can	Background	Overall
Random	16.7/2.9	16.7/2.2	16.7/6.5	16.7/1.0	16.7/0.9	16.7/87.5	16.7/16.7
Det3DMRF	91.5/85.1	90.5/91.4	93.6/94.9	90.0/75.1	81.5/87.4	99.0/99.1	91.0/88.8



**Fig. 9.11** 3D scene labeling results for three complex scenes in the RGB-D Scenes Dataset. 3D reconstruction (*top*), our detection-based scene labeling (*bottom*). Objects colored by their category: bowl is red, cap is green, cereal box is blue, coffee mug is yellow, and soda can is cyan

In Fig. 9.11 we show three complex scenes that were labeled by our 3D scene labeling technique (Det3DMRF). The top row shows the reconstructed 3D scene and the bottom row shows results obtained by Det3DMRF. Objects are colored by their category label, where bowl is red, cap is green, cereal box is blue, coffee mug is yellow, and soda can is cyan. More detailed comparisons with alternative approaches are presented in [26].

If the desired output is bounding boxes in the original RGB-D video frames, it is possible to use the labeled 3D scene to validate object detections. We do this by running object detectors with a low threshold and pruning out bounding box candidates whose labels do not agree with the majority label of points in a central sub-rectangle of the bounding box. Figure 9.12 shows precision-recall curves obtained from both the individual frame-by-frame object detections (red) and detections validated by 3D scene labeling (blue). Each point along the curve is generated by ranking detections from all five category detectors together and thresholding on the detection score. It is clear that 3D scene labeling can significantly reduce false positives by aggregating evidence across the entire video sequence. While the precision of frame-by-frame



**Fig. 9.12** Precision–recall curves comparing the performance of labeling images with bounding boxes of detected objects. Each plot shows results on one of the eight video sequences in the RGB-D Scenes Dataset, aggregated over all five category detectors. Frame-by-frame object detection is drawn in *red*, while 3D scene labeling (our approach) is drawn in *blue*

detection rapidly decreases beyond 60 % recall for all eight scenes, using 3D scene labeling it is possible to obtain 80 % recall and 80 % precision in a majority of them.

## 9.7 Discussion

We presented a large-scale, hierarchical multi-view object dataset collected using an RGB-D camera. We demonstrated methods for doing segmentation by combining depth and visual background subtraction and video ground truth annotation via 3D reconstruction. We also presented and evaluated state-of-the-art features and classification techniques for doing object recognition, object detection, and 3D scene labeling in RGB-D data. The RGB-D Object Dataset, the RGB-D Scenes Dataset, and accompanying software tools are publicly available at <http://www.cs.washington.edu/rgbd-dataset>.

**Acknowledgements** This work was funded in part by the Intel Science and Technology Center for Pervasive Computing and by ONR MURI grant N00014-07-1-0749.

## References

1. Bo, L., Lai, K., Ren, X., Fox, D.: Object recognition with hierarchical kernel descriptors. In: IEEE Conference on Computer Vision and Pattern Recognition (2011)
2. Bo, L., Ren, X., Fox, D.: Kernel descriptors for visual recognition. In: Advances in Neural Information Processing Systems (2010)
3. Bo, L., Ren, X., Fox, D.: Depth kernel descriptors for object recognition. In: Intelligent Robots and Systems (2011)
4. Bo, L., Sminchisescu, C.: Efficient match kernel between sets of features for visual recognition. In: Advances in Neural Information Processing Systems (2009)



5. Breiman, L.: Random forests. *Mach. Learn.* **45**(1), 5–32 (2001)
6. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines (2001)
7. Chen, Y., Gérard, M.: Object modelling by registration of multiple range images. *Image Vis. Comput.* **10**(3), 145–155 (1992)
8. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: *IEEE Conference on Computer Vision and Pattern Recognition* (2005)
9. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Fei-fei, L.: ImageNet: a large-scale hierarchical image database. In: *IEEE Conference on Computer Vision and Pattern Recognition* (2009)
10. Fan, R., Chang, K., Hsieh, C., Wang, X., Lin, C.: Liblinear: a library for large linear classification. *J. Mach. Learn. Res.* **9**, 1871–1874 (2008)
11. Fei-Fei, L., Fergus, R., Perona, P.: One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(4), 594–611 (2006)
12. Felzenszwalb, P., McAllester, D., Ramanan, D.: A discriminatively trained, multiscale, deformable part model. In: *IEEE Conference on Computer Vision and Pattern Recognition* (2008)
13. Fischler, M., Bolles, R.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **24**(6), 381–395 (1981)
14. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: *International Conference on Machine Learning*, pp. 148–156 (1996)
15. Frome, A., Singer, Y., Malik, J.: Image retrieval and classification using local distance functions. In: *Advances in Neural Information Processing Systems* (2006)
16. Frome, A., Singer, Y., Sha, F., Malik, J.: Learning globally-consistent local distance functions for shape-based image retrieval and classification. In: *International Conference on Computer Vision* (2007)
17. Grisetti, G., Grzonka, S., Stachniss, C., Pfaff, P., Burgard, W.: Estimation of accurate maximum likelihood maps in 3d. In: *Intelligent Robots and Systems* (2007)
18. Helmer, S., Lowe, D.G.: Using stereo for object recognition. In: *International Conference on Robotics and Automation*, pp. 3121–3127 (2010)
19. Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D.: RGB-D mapping: using depth cameras for dense 3d modeling of indoor environments. *Int. J. Robot. Res.* (2012). doi:[10.1177/0278364911434148](https://doi.org/10.1177/0278364911434148)
20. Johnson, A., Hebert, M.: Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Trans. Pattern Anal. Mach. Intell.* **21**(5) (1999). doi:[10.1109/34.765655](https://doi.org/10.1109/34.765655)
21. KaewTraKulPong, P., Bowden, R.: An improved adaptive background mixture model for real-time tracking with shadow detection. In: *European Workshop on Advanced Video Based Surveillance Systems* (2001)
22. Microsoft Kinect. <http://www.xbox.com/en-us/kinect>
23. Lai, K., Bo, L., Ren, X., Fox, D.: A large-scale hierarchical multi-view RGB-D object dataset. In: *International Conference on Robotics and Automation* (2011)
24. Lai, K., Bo, L., Ren, X., Fox, D.: A scalable tree-based approach for joint object and pose recognition. In: *Conference on Artificial Intelligence* (2011)
25. Lai, K., Bo, L., Ren, X., Fox, D.: Sparse distance learning for object recognition combining RGB and depth information. In: *International Conference on Robotics and Automation* (2011)
26. Lai, K., Bo, L., Ren, X., Fox, D.: Detection-based object labeling in 3d scenes. In: *International Conference on Robotics and Automation* (2012)
27. Lai, K., Fox, D.: Object Recognition in 3D point clouds using web data and domain adaptation. *Int. J. Robot. Res.* (2010). doi:[10.1177/0278364910369190](https://doi.org/10.1177/0278364910369190)
28. Leung, T., Malik, J.: Representing and recognizing the visual appearance of materials using three-dimensional textons. *Int. J. Comput. Vis.* **43**(1), 29–44 (2001)
29. Lowe, D.: Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **60**, 91–110 (2004)
30. Lowe, D.G.: Object recognition from local scale-invariant features. In: *International Conference on Computer Vision* (1999)

31. Malisiewicz, T., Efros, A.: Recognition by association via learning per-exemplar distances. In: IEEE Conference on Computer Vision and Pattern Recognition (2008)
32. Matuszek, C., Mayton, B., Aimi, R., Deisenroth, M., Bo, L., Chu, R., Kung, M., LeGrand, L., Smith, J., Fox, D.: Gambit: an autonomous chess-playing robotic system. In: International Conference on Robotics and Automation (2011)
33. Meier, L., Van De Geer, S., Bühlmann, P.: The group lasso for logistic regression. *J. R. Stat. Soc., Ser. B* **70**, 53–71 (2008)
34. Pfister, H., Zwicker, M., van Baar, J., Gross, M.: Surfels: surface elements as rendering primitives. *ACM Trans. Graph.* (2000). doi:10.1145/344779.344936
35. PrimeSense. <http://www.primesense.com/>
36. Russell, B., Torralba, K., Murphy, A., Freeman, W.: Labelme: a database and web-based tool for image annotation. *Int. J. Comput. Vis.* **77**(1–3) (2008). doi:10.1007/s11263-007-0090-8
37. Savarese, S., Fei-Fei, L.: 3d generic object categorization, localization and pose estimation. In: International Conference on Computer Vision (2007)
38. Schultz, M., Joachims, T.: Learning a distance metric from relative comparisons. *Adv. Neural Inf. Process. Syst.* (2003)
39. Vondrick, C., Ramanan, D., Patterson, D.: Efficiently scaling up video annotation with crowd-sourced marketplaces. In: European Conference on Computer Vision (2010)
40. Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. *J. Mach. Learn. Res.* **10**, 207–244 (2009)
41. Xing, E., Ng, A., Jordan, M., Russell, S.: Distance metric learning with application to clustering with side-information. *Adv. Neural Inform. Process. Syst.* (2002)