Unsupervised Feature Learning for RGB-D Based Object Recognition

Liefeng Bo[1]*, Xiaofeng Ren[2], and Dieter Fox[1]

[1] University of Washington, Seattle, USA
{lfb,fox}@cs.washington.edu
[2] ISTC-Pervasive Computing Intel Labs, Seattle, USA
xiaofeng.ren@intel.com

Abstract. Recently introduced RGB-D cameras are capable of providing high quality synchronized videos of both color and depth. With its advanced sensing capabilities, this technology represents an opportunity to dramatically increase the capabilities of object recognition. It also raises the problem of developing expressive features for the color and depth channels of these sensors. In this paper we introduce hierarchical matching pursuit (HMP) for RGB-D data. HMP uses sparse coding to learn hierarchical feature representations from raw RGB-D data in an unsupervised way. Extensive experiments on various datasets indicate that the features learned with our approach enable superior object recognition results using linear support vector machines.

1 Introduction

Recognizing object instances and categories is a crucial capability for an autonomous robot to understand and interact with the physical world. Humans are able to recognize objects despite large variation in their appearance due to changing viewpoints, deformations, scales and lighting conditions. This ability fundamentally relies on robust visual representations of the physical world. However, most state-of-the-art object recognition systems are still based on hand-designed representations (features), such as SIFT [26], Spin Images [18], SURF [3], Fast Point Feature Histogram [30], LINE-MOD [15], or feature combinations [20, 7]. Such approaches suffer from at least two key limitations. Firstly, these features usually only capture a small set of recognition cues from raw data; other useful cues are ignored during feature design. For instance, the well-known SIFT features capture edge information from RGB images using a pair of horizonal and vertical gradient filters while completely ignoring color information. Secondly, the features have to be re-designed for new data types, or even new tasks, making object recognition systems heavily dependent on expert experience. It is desirable to develop efficient and effective learning algorithms to automatically learn robust representations from raw data.

Recently, several research groups have developed techniques for unsupervised feature learning from raw vision data [16, 40, 38, 24, 12, 8]. Algorithms such as deep belief nets [16], denoising autoencoders [40], deep Boltzmann machines [38], convolu-

*Bo is now with the ISTC-Pervasive Computing Intel Labs

tional deep belief networks [24], K-Means based feature learning [12, 4], hierarchical sparse coding [43], and hierarchical matching pursuit [8] have been proposed to this end. Such approaches build feature hierarchies from scratch, and have exhibited very impressive performance on many types of recognition tasks such as handwritten digit recognition [16, 40, 38], face recognition [24], tiny image recognition [12], object recognition [24, 12, 43, 8], event recognition [8] and scene recognition [8]. However, the current applications are somewhat limited to 2D images, typically in grayscale.

Recently introduced RGB-D cameras are capable of providing high quality synchronized videos of both color and depth. With its advanced sensing capabilities, this technology represents an opportunity to dramatically increase the capabilities of object recognition. It also raises the problem of developing expressive features for the color and depth channels of these sensors. Inspired by the success of our previous work, hierarchical matching pursuit (HMP) for image classification, we propose *unsupervised feature learning for RGB-D based object recognition* by making hierarchical matching pursuit suitable for color and depth images captured by RGB-D cameras. HMP learns dictionaries over image and depth patches via K-SVD [2] in order to represent observations as sparse combinations of codewords. With the learned dictionary, feature hierarchies are built from scratch, layer by layer, using orthogonal matching pursuit and spatial pyramid pooling [8]. Two major innovations are introduced in this work: (1) Unsupervised feature learning on both color and depth channels; (2) spatial pyramid pooling over sparse codes from both layers of the HMP hierarchy. Extensive evaluations on several publicly available benchmark datasets [20, 10, 39] allowed us to gain various experimental insights: unsupervised feature learning from raw data can yield recognition accuracy that is superior to state-of-the-art object recognition algorithms, even to ones specifically designed and tuned for *textured* objects; the innovations introduced in this work significantly boost the performance of HMP applied to RGB-D data; and our approach can take full advantage of the additional information contained in color and depth channels.

2 Related Work

This research focuses on hierarchical feature learning and its application to RGB-D object recognition. In the past few years, a growing amount of research on object recognition has focused on learning rich features using unsupervised learning, hierarchical architectures, and their combination.

Deep Networks: Deep belief nets [16] learn a hierarchy of features by training multiple layers recursively using the unsupervised restricted Boltzmann machine. This pretraining phase has been shown to avoid shallow local minima. The learned weights are then further adjusted to the current task using supervised information. To make deep belief nets applicable to full-size images, Lee et al. [24] proposed convolutional deep belief nets that use a small receptive field and share the weights between the hidden and visible layers among all locations in an image. Invariant predictive sparse decomposition [17, 19] approximates sparse codes from sparse coding approaches using multilayer feed-forward neural networks and avoid solving computationally expensive optimizations at runtime. Stacked denoising autoencoders [40] build deep networks, based

on stacking layers of denoising autoencoders that train one-layer neural networks to reconstruct input data from partial random corruption. Deconvolutional networks [44] reconstruct images using a group of latent feature maps in a convolutional way under a sparsity constraint. These approaches have been shown to yield competitive performance with the SIFT based bag-of-visual-words model on object recognition benchmarks such as Caltech-101.

Single Layer Sparse Coding: Sparse coding [31] on top of raw images/patches has been developed for face recognition [1], digit recognition [28] and texture segmentation [27]. More recently. researchers have shown that single layer sparse coding on top of SIFT features achieves state-of-the art performance on more challenging object recognition benchmarks [23, 42, 41, 9, 12, 43]. Yang et al. [42] learn sparse codes over SIFT features instead of raw image patches using sparse coding approaches. Their comparisons suggested that such an approach significantly outperforms the standard bag-of-visual-words model. Wang et al. [41] presented a fast implementation of local coordinate coding that computes sparse codes of SIFT features by performing local linear embedding on several nearest codewords in a codebook learned by K-Means. Boureau et al. [9] compared many types of recognition algorithms, and found that the SIFT based sparse coding approaches followed by spatial pyramid max pooling work very well, and macrofeatures can boost recognition performance further. Coates and Ng [12] evaluated many feature learning approaches by decomposing them into training and encoding phases, and suggested that the choice of architecture and encoder is the key to a feature learning system. Yu et al. [43] showed that hierarchical sparse coding at pixel level achieves similar performance with SIFT based sparse coding.

Feature Learning for RGB-D: Kernel descriptors [6] learn patch level feature descriptors based on kernels comparing manually designed pixel descriptors such as gradients, local binary patterns or colors. Adapting this view to depth maps and 3D points, RGB-D kernel descriptors are proposed in [5, 7], and the experiments showed that they obtain higher recognition accuracy than hand-designed feature sets on the RGB-D object dataset [20]. By adapting K-Means based feature learning proposed by Coates and Ng [12] to the RGB-D setting, Blum and colleagues showed that it is possible to learn RGB-D descriptors from raw data that are competitive with RGB-D kernel descriptors on the RGB-D object dataset [4].

3 Unsupervised Feature Learning

This section provides an overview of our feature learning architecture. We review the key ideas behind dictionary learning and discuss our two-layer architecture to generate features over complete RGB-D images.

Building on our previous work on feature learning for object recognition [8], we propose two innovations to make the approach suitable for RGB-D based object recognition. Firstly, we perform feature learning on both color and depth images. The orignal HMP work [8] uses grayscale images only, insufficient in many cases: color is distinctively useful for object instance recognition where appearance details matter, and the depth channel in RGB-D can greatly improve object category recognition and its robustness. We learn dictionaries and encode features using full RGB-D data: gray, RGB,

depth and surface normal channels. Secondly, as described in Section 3.2, we extract features not only from the top of the feature hierarchy, but also from lower layers.

3.1 Dictionary Learning via K-SVD

The key idea of sparse coding is to learn a dictionary, which is a set of vectors, or codes, such that the data can be represented by a sparse, linear combination of dictionary entries. In our case, the data are patches of pixel values in RGB-D frames. For instance, a dictionary for 5×5 RGB-D patches would contain vectors of size $5 \times 5 \times 8$, where the last component is due to grayscale intensity, RGB, depth and surface normal values. Grayscale intensity values are computed from the associate RGB values and normal values are computed from the associated depth values and their coordinates.

K-SVD [2] is a popular dictionary learning approach that generalizes K-Means. It learns dictionaries $D = [d_1, \cdots, d_m, \cdots, d_M]$ and the associated sparse codes $X = [x_1, \cdots, x_n, \cdots, x_N]$ from a matrix Y of observed data by minimizing the reconstruction error

$$\min_{D,X} \|Y - DX\|_F^2 \quad s.t. \ \forall m, \ \|d_m\|_2 = 1 \text{ and } \forall n, \ \|x_n\|_0 \leq K \tag{1}$$

Here, the notation $\|A\|_F$ denotes the Frobenius norm, the zero-norm $\|\cdot\|_0$ counts the non-zero entries in the sparse codes x_n, and K is the sparsity level controlling the number of the non-zero entries. When the sparsity level is set to be 1 and the sparse code matrix is forced to be a binary(0/1) matrix, K-SVD exactly reproduces the K-Means algorithm.

K-SVD solves the optimization problem (1) in an alternating manner. During each iteration, the current dictionary D is used to encode the data Y by computing the sparse code matrix X. Then, the codewords of the dictionary are updated one at a time, resulting in a new dictionary. This new dictionary is then used in the next iteration to recompute the sparse code matrix followed by another round ot dictionary update. We now briefly outline these steps, see [2, 8] for details.

Computing the sparse code matrix via orthogonal matching pursuit: Given the current dictionary D, optimizing the sparse code matrix X can be decoupled into N subproblems; one for each data item y_n. The sparse code x_n for each item y_n is computed efficiently using orthogonal matching pursuit (OMP) [34], a greedy algorithm. In each iteration, OMP selects the codeword d_m that best matches the current residual, which is the reconstruction error remaining after the codewords chosen thus far. In the first iteration, this residual is exactly the observation y_n. Once the new codeword is selected, the observation is orthogonally projected onto the span of all the previously selected codewords and the residual is recomputed. The procedure is repeated until the desired K codewords are selected. In our unsupervised feature learning setting, a large number of image patches share the same dictionary and the total cost of OMP can be reduced by its batch version that keeps some quantities in memory to save computational cost [13, 36, 8].

Updating the dictionary via singular value decomposition: Given the sparse code matrix X, the dictionary D is optimized sequentially via Singular Value Decomposition (SVD). In the m-th step, the m-th codeword and its sparse codes can be computed by performing SVD of the residual matrix corresponding to that codeword. This matrix

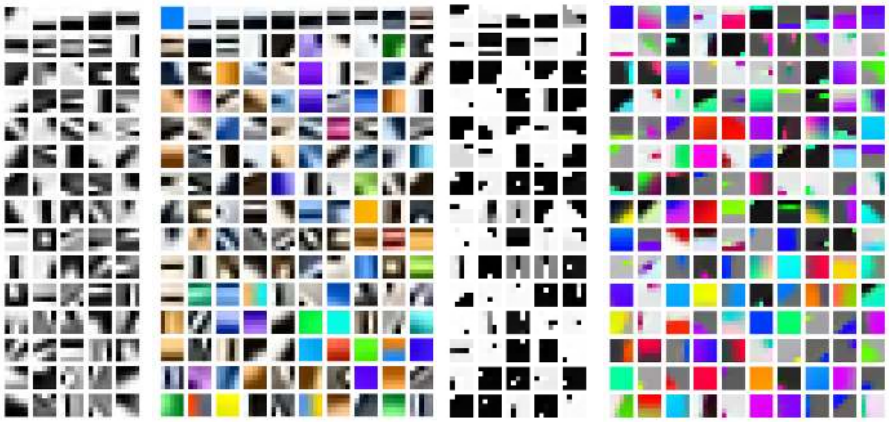

Fig. 1. Dictionaries learned for different channels. From left to right: Grayscale intensity, RGB, depth, 3D surface normal (3 normal dimensions color coded as RGB). The codeword sizes are 5x5x1 for grayscale intensity and depth values, and 5x5x3 for RGB and surface normal values. Dictionary sizes are 75 for grayscale intensity and depth values, and 150 for RGB and surface normal values.

contains the differences between the observations and their approximations using all other codewords and their sparse codes. To avoid introducing new non-zero entries in the sparse code matrix X, the update process only considers observations that use the m-th codeword. It can be shown that each iteration of sparse coding followed by dictionary updating decreases the reconstruction error (1). In practice, K-SVD converges to good dictionaries for a wide range of initializations [2].

In our hierarchical matching pursuit, K-SVD is used to learn dictionaries at two layers, where the data matrix Y in the first layer consists of patches sampled from RGB-D images, and Y in the second layer are sparse codes pooled from the first layer (details below). Fig. 1 visualizes the learned dictionaries in the first layer for four channels: grayscale and RGB for RGB images, and depth and surface normal for depth images. As can be seen, the learned dictionaries have very rich appearances and include separated red, green, blue colors, transition filters between different colors, gray and color edges, gray and color texture filters, depth and normal edges, depth center-surround (dot) filters, flat normals, and so on, suggesting many recognition cues of raw data are well captured.

Once dictionaries are learned via K-SVD, sparse codes can be computed for new images using orthogonal matching pursuit or the more efficient batch tree orthogonal matching pursuit [8]. Fig. 2 shows an example of an RGB / depth image pair along with reconstructions achieved for different levels of sparsity. The shown results are achieved by non-overlapping 5x5 reconstructed patches. As can be seen, a sparsity level of $K = 5$ achieves results that are virtually indistinguishable from the input data, indicating that this technique could also be used for RGB-D *compression*, alternative to [37]. For object recognition, the sparse codes become the features representing the image or segment, as we describe next.

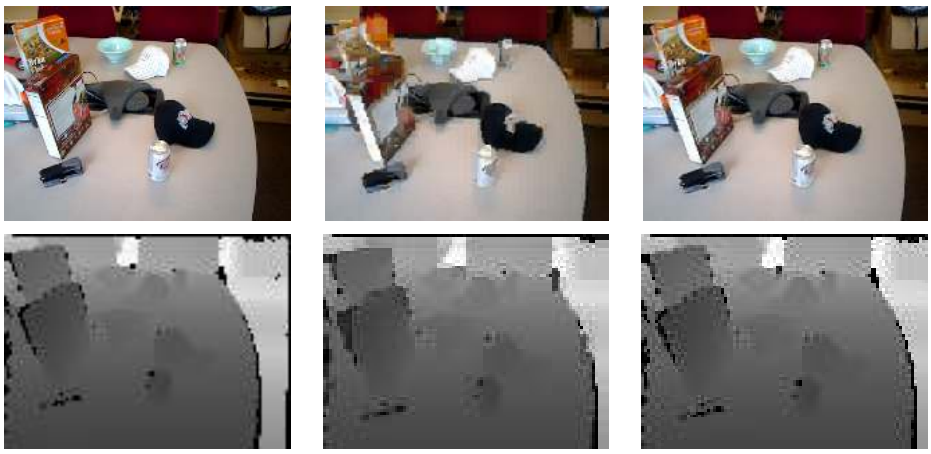

Fig. 2. Reconstructed images using the learned dictionaries. **Left:** Original RGB and depth images. **Middle**: Reconstructed RGB and depth images using only two codewords per 5x5 patch. **Right:** Reconstructions using five codewords.

3.2 Hierarchical Matching Pursuit

With the learned dictionaries D, hierarchical matching pursuit builds a feature hierarchy by applying the orthogonal matching pursuit encoder recursively (Fig. 3). This encoder consists of three modules: batch orthogonal matching pursuit, pyramid max pooling, and contrast normalization (see also [8]).

First Layer: The goal of the first layer is to generate features for image patches whose size is typically 16×16 pixels or larger. Each pixel in such a patch is represented by the sparse codes computed for the pixel and a small neighborhood (for instance, 5×5 pixel region). Spatial pyramid max pooling is then applied to these sparse codes to generate patch level features. Spatial pyramid max pooling partitions an image patch P into multiple level spatial cells. The features of each spatial cell C are the max pooled sparse codes, which are simply the component-wise maxima over all sparse codes within a cell:

$$F(C) = \left[\max_{j \in C} |x_{j1}|, \cdots, \max_{j \in C} |x_{jm}|, \cdots, \max_{j \in C} |x_{jM}| \right] \tag{2}$$

Here, j ranges over all entries in the cell, and x_{jm} is the m-th component of the sparse code vector x_j of entry j. Note that $F(C)$ has the same dimensionality as the original sparse codes. The feature F_P describing a 16x16 image patch P are the concatenation of aggregated sparse codes in each spatial cell

$$F_P = \left[F(C_1^P), \cdots, F(C_s^P), \cdots, F(C_S^P) \right] \tag{3}$$

where $C_s^P \subseteq P$ is a spatial cell generated by spatial pyramid partitions, and S is the total number of spatial cells. As an example, we visualize spatial cells generated by a 3 level spatial pyramid pooling on a 16×16 image patch in Fig. 4. In this example, the dimensionality of the pooled feature vector F_P is $(16+4+1)M$, where M is the size

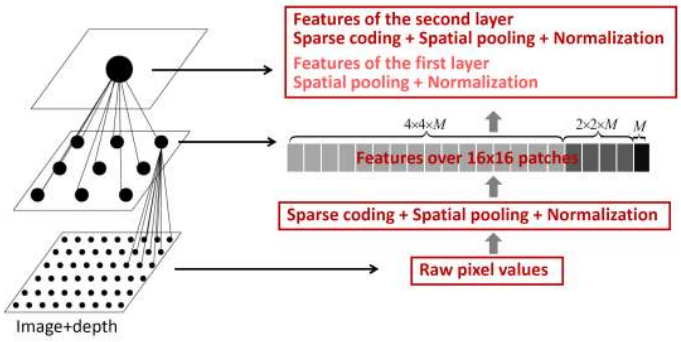

Fig. 3. Hierarchical matching pursuit for RGB-D object recognition. In the first layer, sparse codes are computed on small patches around each pixel. These sparse codes are pooled into feature vectors representing 16×16 patches, by spatial pyramid max pooling. The second layer encodes these feature vectors using a dictionary learned from sampled patch level feature vectors. Whole image features are generated from sparse codes of both first and second layers.

of the dictionary (see also Fig. 3). The main idea behind spatial pyramid pooling is to allow the features F_P to encode different levels of invariance to local deformations [24, 42, 8], thereby increasing the discrimination of the features.

We additionally normalize the feature vectors F_P by their L_2 norm $\sqrt{\|F_P\|^2 + \varepsilon}$, where ε is a small positive number. Since the magnitude of sparse codes varies over a wide range due to local variations in illumination and occlusion, this operation makes the appearance features robust to such variations, as commonly done in the hand-designed SIFT features. We found that $\varepsilon = 0.1$ works well for the recognition problems we considered.

Second Layer: The goal of the second layer in HMP is to generate features for a whole image/object. To do so, HMP applies the sparse coding and max pooling steps to image patch features F_P generated in the first layer. The dictionary for this level is learned by sampling patch features F_P over RGB-D images. To extract the feature describing a whole image, HMP first computes patch features via the first layer (usually, these patches cover 16×16 pixels and are sampled with a step size of 4×4 pixels). Then, just as in the first layer, sparse codes of each image patch are computed using batch orthogonal matching pursuit, followed by spatial max pooling (3×3, 2×2, and 1×1 cell sizes). However, in this layer, we perform max pooling over the sparse codes *and* the patch level features computed in the first layer:

$$G(C) = \left[\max_{j \in C} |z_{j1}|, \cdots, \max_{j \in C} |z_{jU}|, \max_{j \in C} |F_{j1}|, \cdots, \max_{j \in C} |F_{jV}| \right] \tag{4}$$

Here, C is a cell and F_j and z_j are the patch features and their sparse codes, respectively. U and V are the dimensionality of z_j and F_j, where U is given by the size of the layer two dictionary, and V is given by the size of the patch level feature (3). Jointly pooling z_j and F_j integrates both fine-grained cues captured by the codewords in the first layer and coarse-grained cues by those in the second layer, increasing the discrimination of

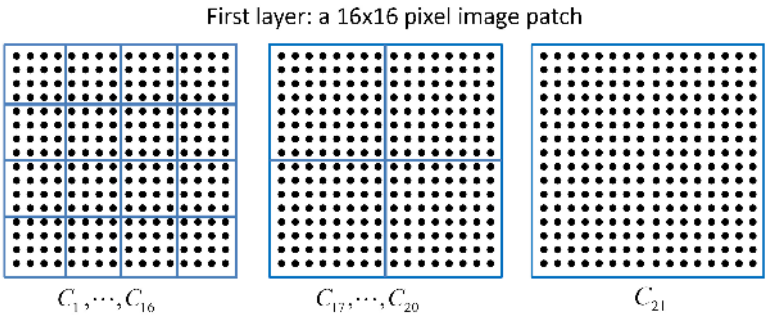

Fig. 4. Spatial pyramid partitions. Each black dot denotes sparse codes of a pixel that are computed on a 5×5 small patch around this pixel using batch orthogonal matching pursuit. **Left:** Level 2. The 16×16 image patch is partitioned into $4 \times 4 = 16$ spatial cells. Each cell is represented by the component-wise maximum of the sparse codes of pixels within the cell. **Middle:** Level 1. The 16×16 image patch is partitioned into $2 \times 2 = 4$ spatial cells. Here, maxima are computed over the level 2 cells. **Right:** Level 0. The whole 16×16 image patch is one spatial cell. The concatenation of C_1 through C_{21} gives the max pooled feature for the whole patch.

the features (joint pooling improves results over those reported in our original HMP work [8]).

The features of the whole image/object are the concatenation of the aggregated sparse codes within each spatial cell. The image feature vector G_I is then normalized by dividing with its L_2 norm $\sqrt{\|G_I\|^2 + 0.0001}$.

It should be noted that hierarchical matching pursuit is a fully unsupervised feature learning approach: no supervision (e.g. object class) is required for dictionary learning and feature coding. The feature vectors G_I of images/objects and their corresponding labels are then fed to classifiers to learn recognition models.

4 Experiments

We evaluate our RGB-D hierarchical matching pursuit framework on three publicly available RGB-D object recognition datasets and two RGB object recognition datasets. We compare HMP to results achieved by state-of-the-art algorithms published with these datasets. For all five datasets, we follow the same training and test procedures used by the corresponding authors on their respective data.

In the first layer, we learn the dictionaries of size 75 with sparsity level 5 on 1,000,000 sampled 5×5 raw patches for grayscale and depth channels, and dictionaries of size 150 on 1,000,000 sampled $5 \times 5 \times 3$ raw patches for RGB and normal channels using K-SVD (see their visualizations in Fig. 1). We remove the zero frequency component from raw patches by subtracting their means. With these learned dictionaries, we compute sparse codes of each pixel (5×5 patch around it) using batch OMP with sparsity level 5, and generate patch level features by max pooling over 16×16 image patches with 4×4, 2×2, and 1×1 partitions. Note that this architecture leads to fast computation of patch level features.

In the second layer, we learn the dictionaries of size 1,000 with sparsity level 10 on 1,000,000 sampled 16×16 patch level features for all four channels using K-SVD. With these learned dictionaries, we compute sparse codes of image patches that are densely sampled from the whole image with a step size of 4×4 pixels. We then pool both patch level features and their sparse codes on the whole image with 3×3, 2×2, and 1×1 partitions to generate the image level features. The final image feature vectors are the concatenation of those from all four channels, resulting in a feature size of 188,300 dimensions.

The above hyperparameters are optimized on a subset of the RGB-D object recognition dataset we collected. We empirically found that they work well on different datasets. In the following experiments, we will keep these values fixed, even though the performance might improve via tuning these parameters for each dataset using cross validation on the associated training data. With the learned HMP features, linear support vector machines (SVMs) are trained for recognition. Linear SVMs are able to match the performance of nonlinear SVMs with the popular histogram intersection kernel [29] while being scalable to large datasets [8].

4.1 RGB-D Object Dataset

The first dataset, called `RGBD`, contains 41,877 RGB-D images of 300 physically distinct everyday objects taken from different viewpoints [20]. The objects are organized into 51 categories arranged using WordNet hypernym-hyponym relationships. The objects in the dataset are segmented from the background by combining color and depth segmentation cues. The `RGBD` dataset is challenging since it not only contains textured objects such as food bags, soda cans, and cereal boxes, but also texture-less objects such as bowls, coffee mugs, fruits, or vegetables. In addition, the data frames in `RGBD` additionally exhibit large changes in lighting conditions.

Object Recognition. We distinguish between two types of object recognition tasks: instance recognition and category recognition. Instance recognition is to recognize known object instances. Category recognition is to determine the category name of a previously unseen object. Each object category consists of a number of different object instances. Following the experimental setting in [20], we randomly leave one object instance out from each category for testing, and train models on the remaining 300 - 51 = 249 objects at each trial for category recognition. We report the accuracy averaged over 10 random train/test splits. For instance recognition, we train models on images captured from 30° and 60° elevation angles, and test them on the images of the 45° angle (leave-sequence-out).

We compare HMP with the baseline [20], kernel descriptors [7], convolutional k-means descriptors (CKM Desc) [4], and the original HMP [8] (features from the second layer only; grayscale and depth, but no color and normal) in Table 1. The recognition systems developed in [20, 7, 22] use a rich set of manually designed features. As can be seen, HMP outperforms all previous approaches for both category and instance recognition. For instance recognition, features learned on color images substantially improve the performance relative to those on grayscale images.

RGBD	Category			Instance		
Methods	RGB	Depth	RGB-D	RGB	Depth	RGB-D
ICRA11 [20]	74.3 ± 3.3	53.1 ± 1.7	81.9 ± 2.8	59.3	32.3	73.9
Kernel descriptors [7, 22]	80.7 ± 2.1	80.3 ± 2.9	86.5 ± 2.1	90.8	54.7	91.2
CKM Desc [4]	N/A	N/A	86.4 ± 2.3	82.9	N/A	90.4
HMP [8]	74.7 ± 2.5	70.3 ± 2.2	82.1 ± 3.3	75.8	39.8	78.9
This work	82.4 ± 3.1	81.2 ± 2.3	87.5 ± 2.9	92.1	51.7	92.8

Table 1. Comparisons with the baseline [20], kernel descriptors [7], convolutional k-means descriptor [4] and the original HMP [8].

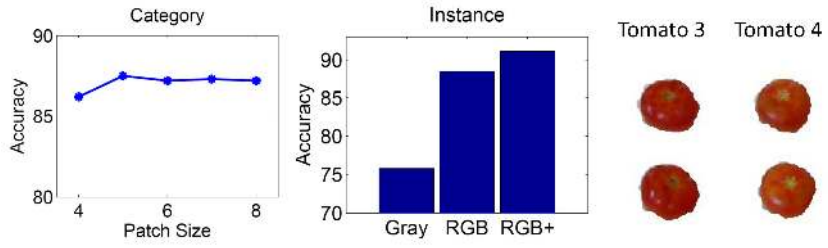

Fig. 5. Left: category recognition accuracy as a function of the filter size. **Middle:** instance recognition accuracy by using features on grayscale images (Gray), features on color images (RGB) from the second layer only, features on color images from both layers (RGB+). **Right:** two tomato instances confused by our approach.

We performed additional experiments to shed light on different aspects of our approach. Fig. 5 (*left*) shows category recognition accuracy as a function of the patch size in the first layer. A larger patch size helps to improve the accuracy, but becomes saturated around 5×5. In Fig. 5 (*middle*), we show instance recognition accuracy using features on grayscale images and features on color images from the second layer only, and features on color images from both layers. As can be seen, features on color images work much better than those on grayscale images. This is expected since color information plays an important role for instance recognition. Object instances that are distinctive in the color space may have very similar appearance in grayscale space. We investigated the object instances misclassified by HMP, and found that most of the mistakes are from fruits and vegetables. Two misclassified tomatoes are shown in Fig. 5. As one can see, these two tomato instances are so similar that even humans struggle to tell them apart. If such objects are excluded from the dataset, our approach has more than 95% accuracy for instance recognition on the RGBD dataset.

We also investigate recognition accuracies using features from the first layer only, from the second layer only, and from both layers. We observe that integrating features from both layers improves performance by about 2 percents. Features from the second layer are better than those from the first layer for category recognition while features from the first layer are better than the second layer for instance recognition. This makes sense intuitively, since coarse-grained information (second layer) is more important for category recognition whereas fine-grained information (first layer) is more important for instance recognition.

Technique	MedPose	MedPose(C)	MedPose(I)	AvePose	AvePose(C)	AvePose(I)	Test Time(s)
NN	144.0°	105.1°	33.5°	109.6°	98.8°	62.6°	54.8
Indep Tree	73.3°	62.1°	44.6°	89.3°	81.4°	63.0°	0.31
OPTree	62.6°	51.5°	30.2°	83.7°	77.7°	57.1°	0.33
This work	20.0°	18.7°	18.0°	53.6°	47.5°	44.8°	0.51

Table 2. Pose estimation error (in degrees) and running time (in seconds) comparison of several approaches. Indep Tree is a tree of classifiers where each level is trained as independent linear SVMs, NN is nearest neighbor regressor, and OPTree is the Object-Pose Tree proposed in [21]. Median pose accuracies for MedPose, MedPose(C) and MedPose(I) are 88.9%, 89.6% and 90.0%, respectively. Mean pose accuracies for AvePose, AvePose(C) and AvePose(I) are 70.2%, 73.6% and 75.1%, respectively.

Pose Estimation. We further evaluated the HMP features for pose estimation, where the pose of every view of every object is annotated as the angle about the vertical axis. Each object category has a canonical pose that is labeled as 0°, and every image in the dataset is labeled with a pose in $[0, 360^\circ]$. Similar to instance recognition, we use the 30° and 60° viewing angle sequences as training set and the 45° sequence as test set. For efficiency, we follow an independent tree approach to estimate pose, where each level is trained as an independent classier [21]: Firstly, one-versus-all category classifiers are trained in the category level; secondly, one-versus-all instance classifiers are trained in the instance level within each category; and finally one-versus-all pose classifiers in the pose level are trained within each instance. At test time, category, instance and pose classifiers are run in turn to estimate the pose of a query object.

Table 6 shows pose estimation errors under three different scenarios. We report both median pose (MedPose) and mean pose (AvePose) errors because the distribution across objects is skewed [21]. For MedPose and AvePose, pose errors are computed on the entire test set, where test images that were assigned an incorrect category or instance label have a pose error of 180.0°. MedPose(C) and AvePose(C) are computed only on test images that were assigned the correct category by the system, and, MedPose(I) and AvePose(I) are computed only on test images that were assigned the correct instance by the system. We compare HMP to our previous results [21]. As can been seen from Table 2, with our new HMP features, pose estimation errors are significantly reduced under all scenarios, resulting in only 20° median error even when classification errors are measured as 180.0° offset. We visualize test images and the best matched images in Fig. 6. The results are very intuitive: estimations are quite accurate for non-symmetric objects and sometimes inaccurate for symmetric objects for which different poses could share very similar or exactly same appearances.

4.2 Willow and 2D3D Datasets

We evaluated HMP on two other publicly available RGB-D recognition datasets. The first dataset, `2D3D`, consists of 156 object instances organized into 14 categories [10]. The authors of this dataset also use a large set of 2D and 3D manually designed shape and color features. SVMs are trained for each feature and object class, followed by multilayer perceptron learning to combine the different features. The second dataset,

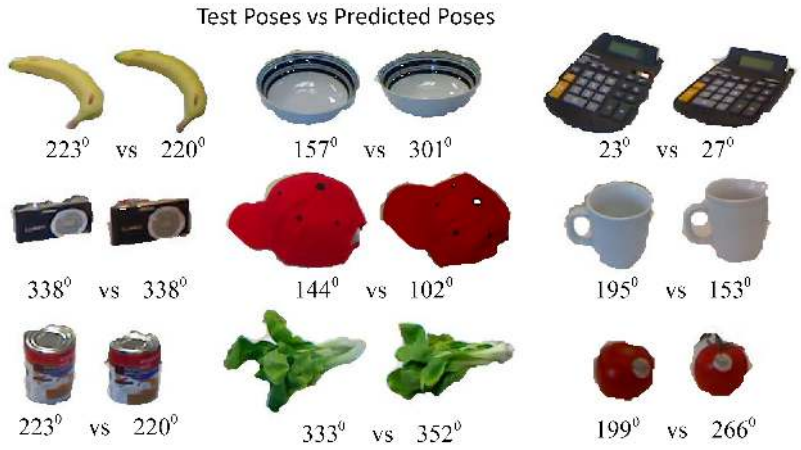

Fig. 6. Test images and the best matched images using HMP features.

Fig. 7. Ten of the thirty-five textured household objects from the Willow dataset

`Willow`, contains objects from the Willow and Challenge datasets for training and testing, respectively [39]. Both training and test data contain 35 rigid, textured, household objects captured from different views by Willow Garage. The authors present a processing pipeline that uses a combination of SIFT feature matching and geometric verification to perform recognition of highly textured object instances [39]. Note that `2D3D` and `Willow` only contain highly textured objects.

We report the results of HMP in Table 3. Following the experimental setting in [10], HMP yields 91.0% accuracy for category recognition, much higher than the 82.8% reported in [10]. Learning models on training data from the Willow dataset and testing them on the training data from the Challenge dataset [39], HMP achieves higher precision/recall than the system proposed in [39], which won the 2011 Perception Challenge organized by Willow Garage. Note that that system is specifically designed for textured objects and thus could not, in contrast to our learned features, be applied to untextured objects such as those found in the `RGBD` dataset.

4.3 Learning and Vision Datasets

We also tested our model on the feature learning dataset `STL-10` [11] and on the vision dataset `MITScenes-67` [35]. We used the same architecture for these datasets

2D3D	Category Recognition			Willow	Instance Recognition
Methods	RGB	Depth	RGB-D	Methods	Precision/Recall
ICCVWorkshop [10]	66.6	74.6	82.8	ICRA12 [39]	96.7/97.4
This work	86.3	87.6	91.0	This work	97.4/100.0

Table 3. Comparisons with the previous results on the two public datasets: Willow and 2D3D.

STL-10		MITScenes-67			
VQ [11]	54.9 ± 0.4	GIST-color [32]	29.7	OB [25]	37.6
SC [12]	59.0 ± 0.8	DPM [14]	30.4	RBoW [33]	37.9
Learned RF [12]	60.1 ± 1.0	SPM [32]	34.4	DPM+Gist-color+SPM [32]	43.1
This work	64.5 ± 1.0	SC [8]	36.9	This work	47.6

Table 4. Comparisons with the previous results on the STL-10 and MITScenes-67.

as for RGB-D datasets. The dictionaries are learned on both RGB and grayscale channels and the final features are the concatenation of HMP features from these two channels. Following the standard setting in [11], we train linear SVMs on 1000 images and test on 8000 images using our HMP features and report the averaged accuracy over 10 pre-defined folds by the authors. As can be seen in Table 4, HMP achieves much higher accuracy than the receptive field learning algorithm [12] that beat many types of deep feature learning approaches as well as single layer sparse coding on top of SIFT (SC) [12]. Training linear SVMs on 80 images and testing on 20 images per category on the pre-defined training/test split by the authors, HMP achieves higher accuracy than many state-of-the-art algorithms: spatial pyramid matching (SPM) [32], deformable parts models (DPM) [14], object bank (OB) [25], Reconfigurable Models (RBoW) [33], and even the combination of SPM, DPM, and color GIST [32].

5 Conclusions

We demonstrated that recent advances in unsupervised feature learning make it possible to learn very rich features from raw RGB-D data. Our approach, HMP, consistently outperforms state-of-the-art techniques on five benchmark datasets. Importantly, even though HMP is designed fsor very general object recognition, it even outperforms techniques specifically designed for highly textured objects, when applied to such data. These results are extremely encouraging, indicating that current recognition systems can be significantly improved without resorting to careful, manual feature design. We believe this work opens up many possibilities for learning rich, expressive features from raw RGB-D data. In the current implementation, we manually designed the architecture of HMP. Automatically learning such structure is interesting but also very challenging and left for future work. Our current experience is that learning dictionaries separately for each channel works better than learning them jointly. We plan to explore other possibilities of joint dictionary learning in the future.

Acknowledgments

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. Robust Face Recognition via Sparse Representation, author=Wright, J. and Yang, A. and Ganesh, A. and Sastry, S. and Ma, Y. *IEEE PAMI*, 31(2):210–227, 2009.
2. M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.
3. H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.
4. M. Blum, J. Springenberg, J. Wlfing, and M. Riedmiller. A Learned Feature Descriptor for Object Recognition in RGB-D Data. In *ICRA*, 2012.
5. L. Bo, K. Lai, X. Ren, and D. Fox. Object Recognition with Hierarchical Kernel Descriptors. In *CVPR*, 2011.
6. L. Bo, X. Ren, and D. Fox. Kernel Descriptors for Visual Recognition. In *NIPS*, 2010.
7. L. Bo, X. Ren, and D. Fox. Depth Kernel Descriptors for Object Recognition. In *IROS*, 2011.
8. L. Bo, X. Ren, and D. Fox. Hierarchical Matching Pursuit for Image Classification: Architecture and Fast Algorithms. In *NIPS*, 2011.
9. Y. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning Mid-level Features for Recognition. In *CVPR*, 2010.
10. B. Browatzki, J. Fischer, B. Graf, HH. Blthoff, and C. Wallraven. Going into Depth: Evaluating 2D and 3D Cues for Object Classification on a New, Large-scale Object Dataset. In *1st ICCV Workshop on Consumer Depth Cameras in Computer Vision*, 2011.
11. A. Coates, H. Lee, and A. Ng. An Analysis of Single-layer Networks in Unsupervised Feature Learning. In *International Conference on AI and Statistics*, 2011.
12. A. Coates and A. Ng. The Importance of Encoding versus Training with Sparse Coding and Vector Quantization. In *ICML*, 2011.
13. G. Davis, S. Mallat, and M. Avellaneda. Adaptive Greedy Approximations. *Constructive Approximation*, 13(1):57–98, 1997.
14. P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part-Based Models. *IEEE PAMI*, 32:1627–1645, 2010.
15. S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal Templates for Real-Time Detection of Texture-less Objects in Heavily Cluttered Scenes. In *ICCV*, 2011.
16. G. Hinton, S. Osindero, and Y. Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18(7):1527–1554, 2006.
17. K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the Best Multi-Stage Architecture for Object Recognition? In *ICCV*, 2009.
18. A. Johnson and M. Hebert. Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes. *IEEE PAMI*, 21(5), 1999.
19. K. Kavukcuoglu, P. Sermanet, Y. Boureau, K. Gregor, M. Mathieu, and Y. LeCun. Learning Convolutional Feature Hierarchies for Visual Recognition. In *NIPS*. 2010.
20. K. Lai, L. Bo, X. Ren, and D. Fox. A Large-Scale Hierarchical Multi-View RGB-D Object Dataset. In *ICRA*, 2011.

21. K. Lai, L. Bo, X. Ren, and D. Fox. A Scalable Tree-based Approach for Joint Object and Pose Recognition. In *AAAI*, 2011.
22. K. Lai, L. Bo, X. Ren, and D. Fox. RGB-D Object Recognition: Features, Algorithms, and a Large Scale Benchmark. In A. Fossati, J. Gall, H. Grabner, X. Ren, and K. Konolige, editors, *Consumer Depth Cameras for Computer Vision: Research Topics and Applications*. Springer, 2012.
23. H. Lee, A. Battle, R. Raina, and A. Ng. Efficient Sparse Coding Algorithms. In *NIPS*. 2007.
24. H. Lee, R. Grosse, R. Ranganath, and A. Ng. Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. In *ICML*, 2009.
25. L. Li, H. Su, E. Xing, and L. Fei-Fei. Object Bank: A High-Level Image Representation for Scene Classification and Semantic Feature Sparsification. In *NIPS*, 2010.
26. D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.
27. J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Discriminative learned dictionaries for local image analysis. In *cvpr*, pages 1–8, 2008.
28. J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised Dictionary Learning. In *NIPS*, pages 1033–1040, 2008.
29. S. Maji, A. Berg, and J. Malik. Classification Using Intersection Kernel Support Vector Machines is Efficient. In *CVPR*, 2008.
30. B. Morisset, R. Bogdan Rusu, A. Sundaresan, K. Hauser, M. Agrawal, J. Latombe, and M. Beetz. Leaving Flatland: Toward Real-Time 3D Navigation. In *ICRA*, 2009.
31. B. Olshausen and D. Field. Emergence of Simple-cell Receptive Field Properties by Learning a Sparse Code for Natural Images. *Nature*, 381:607–609, 1996.
32. M. Pandey and S. Lazebnik. Scene Recognition and Weakly Supervised Object Localization with Deformable Part-Based Models. In *ICCV*, 2011.
33. S. Naderi Parizi, J. Oberlin, and P. Felzenszwalb. Reconfigurable Models for Scene Recognition. In *CVPR*, 2012.
34. Y. Pati, R. Rezaiifar, and P. Krishnaprasad. Orthogonal Matching Pursuit: Recursive Function Approximation with Applications to Wavelet Decomposition. In *The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers*, pages 40–44, 1993.
35. A. Quattoni and A. Torralba. Recognizing Indoor Scenes. In *CVPR*, 2009.
36. R. Rubinstein, M. Zibulevsky, and M. Elad. Efficient Implementation of the K-SVD Algorithm using Batch Orthogonal Matching Pursuit. Technical report, 2008.
37. M. Ruhnke, B. Steder, G. Grisetti, and W Burgard. Unsupervised Learning of Compact 3D Models Based on the Detection of Recurrent Structures. In *IROS*, 2010.
38. R. Salakhutdinov and G. Hinton. Deep Boltzmann Machines. In *International Conference on AI and Statistics*, 2009.
39. J. Tang, S. Miller, A. Singh, and P. Abbeel. A Textured Object Recognition Pipeline for Color and Depth Image Data. In *ICRA*, 2012.
40. P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol. Extracting and Composing Robust Features with Denoising Autoencoders. In *ICML*, 2008.
41. J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Guo. Locality-constrained Linear Coding for Image Classification. In *CVPR*, 2010.
42. J. Yang, K. Yu, Y. Gong, and T. Huang. Linear Spatial Pyramid Matching using Sparse Coding for Image Classification. In *CVPR*, 2009.
43. K. Yu, Y. Lin, and J. Lafferty. Learning Image Representations from the Pixel Level via Hierarchical Sparse Coding. In *CVPR*, 2011.
44. M. Zeiler, D. Krishnan, G. Taylor, and R. Fergus. Deconvolutional Networks. In *CVPR*, 2010.