Article

The International Journal of Robotics Research
2014, Vol. 33(4) 581–599
© The Author(s) 2014
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0278364913514283
ijr.sagepub.com

Learning hierarchical sparse features for RGB-(D) object recognition

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

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 significantly 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. As a multi-layer sparse coding network, HMP builds feature hierarchies layer by layer with an increasing receptive field size to capture abstract representations from raw RGB-D data. HMP uses sparse coding to learn codebooks at each layer in an unsupervised way and builds hierarchical feature representations from the learned codebooks in conjunction with orthogonal matching pursuit, spatial pooling and contrast normalization. Extensive experiments on various datasets indicate that the features learned with our approach enable superior object recognition results using linear support vector machines.

Keywords
Object recognition, feature learning, sparse coding, hierarchical segmentation, RGB-D cameras

1. Introduction

Recognizing object instances and categories is a crucial capability for an autonomous robot to understand and interact with the physical world. Only by recognizing objects can a robot reason about their affordances, manipulate them in a purposeful way, and understand commands given in natural language, such as "Put the dirty cereal bowl into the dish washer". Because of its importance, object recognition has recently received substantial attention in the robotics community. Humans are able to recognize objects despite large variation in their appearance due to changing viewpoints, scales, lighting conditions and deformations. 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 manually designed representations (features), such as SIFT (Lowe, 2004), HOG (Dalal and Triggs, 2005), Spin Images (Johnson and Hebert, 1999), SURF (Bay et al., 2008), Fast Point Feature Histogram (Morisset et al., 2009), LINE-MOD (Hinterstoisser et al., 2011), or feature combinations (Lai et al., 2011a). 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 horizontal 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 machine learning algorithms to automatically extract robust representations from raw sensor data.

Instead of manually designing features, feature learning aims to transform raw data to expressive features with which tasks such as classification can be solved more accurately. This approach is attractive as it exploits the availability of a large amount of unlabeled data and avoids the need of feature engineering, and recently it has become increasingly popular and effective for object recognition. In the last few years, a variety of feature learning techniques have been developed (Hinton et al., 2006; Lee et al., 2009; Wang et al., 2010; Bo et al., 2011c). Algorithms such as deep belief nets, convolutional deep belief networks, convolutional neural networks, deep autoencoders, recursive neural networks, k-means based feature learning, and sparse coding have been proposed to this end. Such approaches build

[1]ISTC-Pervasive Computing Intel Labs, Seattle, USA
[2]University of Washington, Seattle, USA

Corresponding author:
Liefeng Bo, Box 352350, 101 Paul G. Allen Center for CSE446 Seattle, Washington 98105, USA.
Email: lfb@cs.washington.edu

feature hierarchies from scratch, and have exhibited very impressive performance on many types of recognition tasks including handwritten digit recognition, face recognition, tiny image recognition, object recognition, and scene recognition. Though very successful, the current vision applications are somewhat limited to 2D images, typically in grayscale.

Recently introduced low-cost RGB-D cameras are capable of providing high quality synchronized videos of both color and depth. The registered color and depth have potential to boost a wide range of robotics applications. For instance, the Microsoft Kinect sensor, developed by PrimeSense, provides 640×480 color and depth images at 30 frames per second and has been successfully used for computer gaming (Shotton et al., 2011) and 3D reconstruction (Henry et al., 2010; Newcombe et al., 2011). With its advanced sensing capabilities, this technology represents an opportunity to dramatically increase the capabilities of object recognition and manipulation as well. Over the past few years, the robotics community has made substantial progress in recognizing pre-segmented objects (Bo et al., 2011b; Lai et al., 2011a; Tang et al., 2012), detecting objects from cluttered scenes (Hinterstoisser et al., 2011; Lai et al., 2011a), and labeling objects in 3D point-clouds from RGB-D video sequences (Anand et al., 2012; Lai et al., 2012a; Jiang et al., 2012). Though very promising, these approaches are still built on designed feature sets that could bound the overall performance of the whole system. This raises the problem of developing expressive features for the color and depth channels of RGB-D sensors.

In this paper, we propose hierarchical matching pursuit (HMP) to learn features from scratch for color and depth images captured by RGB-D cameras. HMP is a multi-layer sparse coding network that builds feature hierarchies from raw RGB-D data layer by layer with an increasing receptive field size. In particular, HMP learns codebooks at each layer via K-SVD (Aharon et al., 2006) in order to represent patches or pooled sparse codes as a sparse, linear combination of codebook entries (codewords). With the learned dictionary, feature hierarchies are built, layer by layer, using three components: orthogonal matching pursuit, spatial pooling, and contrast normalization. We discuss the architecture of HMP, and show that all three components and hierarchical structure are critical for learning good features for object recognition. We further present batch tree orthogonal matching pursuit to speed up the computation of sparse codes by pre-computing quantities shared by a large number of image patches.

Extensive evaluations on several publicly available benchmark datasets enable us to gain various experimental insights: features learned 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 (Tang et al., 2012); HMP can take full advantage of the additional information contained in color and depth channels and significantly boosts the performance of RGB-D based object recognition. We also present results showing that HMP can be combined with a state-of-the-art segmentation technique to achieve very good performance on object detection, which is closer to realistic robotics applications.

This paper is organized as follows. Section 2 reviews related work. Our feature learning approach, HMP, is presented in Section 3, followed by an extensive experimental evaluation in Section 4. We conclude in Section 5.

2. Related work

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

2.1. *RGB-D object recognition and detection*

Low-cost RGB-D cameras have dramatically increased the capabilities of object recognition and detection. Kinect makes the human body a controller by enabling accurate real-time human pose estimation (Shotton et al., 2011). Lai et al. (2011a) collected an RGB-D Object Dataset using a Kinect style camera, and investigated classical bag-of-words based object recognition and sliding window based object detection using designed shape and color features extracted from a single RGB-D frame. Tang et al. (2012) combine SIFT feature matching and geometric verification to recognize highly textured object instances, and won the ICRA 2011 Solutions in Perception instance recognition challenge. Hinterstoisser et al. (2011) proposed a template based approach for real-time object detection by quantizing gradient orientations computed on a grayscale image and surface normals computed on a depth image into integer values. Object recognition and detection can be further improved by leveraging the geometric information of the reconstructed 3D scene, i.e. 3D point clouds. Anand et al. (2012) detect commonly found objects in the 3D point cloud of indoor scenes obtained from RGB-D cameras. The approach exploits a graphical model to capture various features and contextual relations, including local visual appearance and shape cues, object co-occurence relationships and geometric relationships, and achieves good accuracy in labeling office and home scenes. Lai et al. (2012a) utilize sliding window detectors trained from RGB-D frames to assign probabilities to voxels in a reconstructed 3D scene. A Markov Random Field is then applied to label voxels by combining cues from view-based detection and 3D shape. Jiang et al. (2012) proposed a supervised learning approach for detecting good placements from the given point clouds of an object and the placing area. The approach learns to combine the features that capture support, stability, and preferred placements using a shared sparsity structure in the parameters, and enables a robot to stably place several new objects in several new placing areas.

2.2. Deep networks

Since a 2006 breakthrough by Hinton et al. (2006), deep learning has evolved as a popular machine learning tool for image recognition. Deep belief nets (Hinton et al., 2006) build a hierarchy of features layer by layer using the unsupervised restricted Boltzmann machine. The procedure is called pre-training and very helpful for avoiding shallow local minima. The found weights are then fed to multi-layer feed-forward neural networks and further adjusted to the current task using supervised information. To make deep belief nets applicable to full-size images, Lee et al. (2009) 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 by adapting the idea of convolutional neural networks (LeCun and Haffner, 1998). Deconvolutional networks (Zeiler et al., 2011) convolutionally decompose images in an unsupervised way under a sparsity constraint. By building a hierarchy of such decompositions, robust representations can be built from raw images for image recognition. Tiled convolutional neural networks (Le et al., 2011) are a type of sparse deep architecture with three important ingredients: local receptive fields, pooling, and local contrast normalization that enjoy the benefit of significantly reducing the number of learnable parameters while giving the algorithm flexibility to learn invariant features. Recent work (Le et al., 2012) has shown that such deep networks can build high-level, class-specific feature detectors from a large collection of unlabeled images, for instance human and cat face detectors. Very recently, deep convolutional neural networks have achieved impressive results on the latest ImageNet challenge and demonstrated its potential for recognizing images (Krizhevsky et al., 2012). In addition to image recognition, recursive neural networks (Socher et al., 2011) have shown promising results for semantical scene labeling by leveraging recursive structure in natural scene images. The key idea is to compute a score for merging neighboring regions into a larger region, a new semantic feature representation for this larger region, and its class label.

2.3. Sparse coding

For many years, sparse coding (Olshausen and Field, 1996) has been a popular tool for modeling images. Sparse coding on top of raw image patches has been applied not only for image processing including image denoising (Elad and Aharon, 2006a), image compression (Donoho, 2006), and super-resolution (Yang et al., 2008), but also for image recognition including face recognition (Wright et al., 2009), digit recognition (Mairal et al., 2008b), and texture segmentation (Mairal et al., 2008a). More recently, it has been shown that combining sparse coding with successful image features dramatically boosts image recognition. For instance, sparse coding on top of SIFT features achieves state-of-the-art performance on challenging object recognition benchmarks (Yang et al., 2009). Yang et al. (2009) first proposed sparse coding over SIFT features rather than raw image patches for image recognition, and demonstrated that it significantly outperforms popular bag-of-visual-words models. Wang et al. (2010) presented a fast local coordinate coding scheme for recognizing images, and the approach 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. (2010) compared many types of object recognition algorithms, and found that SIFT based sparse coding followed by spatial pyramid max pooling are the best-performing approaches, and macrofeatures can increase recognition accuracy even further. Coates and Ng (2011b) evaluated many sparse coding 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. Sparse coding requires a computationally expensive optimization stage for computing sparse codes, which makes it infeasible for real-time applications. Invariant predictive sparse decomposition (Jarrett et al., 2009; Kavukcuoglu et al., 2010) approximates sparse codes using multi-layer feed-forward neural networks and avoids solving expensive optimizations at runtime. Bo et al. (2011c) proposed a batch tree orthogonal matching pursuit to speed up the computation of sparse codes. The approach can handle full-size images efficiently, and is particularly suitable for encoding a large number of observations that share the same large dictionary. Inspired by deep networks, multi-layer sparse coding networks including hierarchical sparse coding (Yu et al., 2011) and HMP (Bo et al., 2011c, 2013) have been proposed for building hierarchical features from raw sensor data. Such networks learn codebooks at each layer in an unsupervised way so that image patches or pooled sparse codes can be represented by a sparse, linear combination of codebook entries.

2.4. RGB-D features learning

Motivated by the success of feature learning, researchers in the robotics community started to investigate their use for RGB-D object recognition. Kernel descriptors (Bo et al., 2010) learn patch level feature descriptors based on kernels comparing manually designed pixel descriptors such as gradients, local binary patterns or colors. By adapting the kernel view to depth maps and 3D points, Bo et al. proposed RGB-D kernel descriptors (2011a; 2011b), and showed that they achieve much higher recognition accuracy than designed feature sets on the RGB-D Object Dataset (Lai et al., 2011a). Blum et al. (2012) adapted k-means based feature learning proposed by Coates and Ng (2011b) to the RGB-D setting, and demonstrated that the learned RGB-D descriptors from raw data are competitive with RGB-D kernel descriptors on the RGB-D Object Dataset. These approaches are built on either designed filters (Bo et al., 2011b) or single-layer architectures (Coates and Ng, 2011b), and thus have difficulty learning highly abstract representations.

3. Hierarchical matching pursuit

State-of-the-art object recognition and detection systems (Yang et al., 2009; Felzenszwalb et al., 2010; Hinterstoisser et al., 2011) are built on designed features, such as SIFT (Lowe, 2004), HOG (Dalal and Triggs, 2005), and LINE-MOD (Hinterstoisser et al., 2011). Though very successful, such features heavily depend on the knowledge of the domain experts and might throw away useful information. A typical feature design pipeline is to first define a set of filters, quantize their responses on input data and then apply local spatial pooling to the quantized vectors. The procedure has at least two limitations. First, it is time-consuming for the domain experts to find good filters for object recognition. Second, the designed filter set might fail to capture important recognition cues. For instance, both SIFT and HOG use a pair of horizontal and vertical gradient filters to generate features from RGB images. Gradient filters are strong for capturing edge-type cues while totally ignore color-type cues: color blocks, color textures, color gradients and so on that are very important for recognizing objects. We aim to alleviate these limitations by proposing a hierarchical sparse feature learning framework. Our approach learns large filter sets from raw RGB-D images, and provides an efficient and effective way to capture rich recognition cues.

This section provides an overview of our feature learning algorithms. We discuss the key ideas behind sparse coding, and propose our dual-layer architecture for unsupervised feature learning and fast orthogonal matching pursuit for computing sparse codes. Our approach learns codebooks and encodes features using full RGB-D data: gray, RGB, depth, and surface normal channels. The extensive experiments show that these features encode rich information and improve both category and instance recognition.

3.1. *Orthogonal matching pursuit*

Sparse coding has become a popular tool in many fields including signal processing and image recognition (Olshausen and Field, 1996; Donoho, 2006; Wright et al., 2009; Yang et al., 2009). It models data as sparse linear combinations of codewords selected from a codebook. The key idea is to learn a codebook, which is a set of vectors, or codes, such that the data can be represented by a sparse, linear combination of codebook entries. In our case, the data are patches of pixel values in RGB-D frames. For instance, a codebook 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(1), RGB(3), depth(1), and surface normal values(3). Grayscale intensity values are computed from the associate RGB values and surface normal values are computed from the associated depth values and their coordinates.

Algorithm 1: Orthogonal matching pursuit (OMP)

```
1. Input: Codebook $D$, observation $y$, and the desired sparsity
       level $K$
2. Output: Sparse code $x$ such that $y \approx Dx$
3. Initialization: $I = \emptyset$, $r = y$, and $x = 0$
4. For $k = 1 : K$
5.     Selecting the new codeword: $\overline{m} = \text{argmax}_m |d_m^\top r|$
6.     $I = I \cup \overline{m}$
7.     Computing the sparse code: $x_I = (D_I^\top D_I)^{-1} D_I^\top y$
8.     Computing the residual: $r = y - D_I x_I$
9. End
```

For now, assume a known codebook, $D = [d_1, \ldots, d_m, \ldots, d_M] \in R^{H \times M}$, containing M codewords of size H each. The size of the codewords is identical to the size of the image patches to be encoded. Given an image patch, y, the encoding problem in sparse coding is to find the sparse code x of y

$$\min_x \|y - Dx\|^2 \qquad s.t. \ \ \|x\|_0 \leq K \tag{1}$$

where the zero-norm $\| \cdot \|_0$ counts the non-zero entries in the sparse codes x, and K is the sparsity level controlling the number of the non-zero entries. This problem is also known as compressed sensing in the signal processing community (Candès et al., 2006; Donoho, 2006). Computing the optimal solution involves searching over all the $\binom{M}{K}$ possible combinations and thus is NP-hard. Approximation algorithms are often considered. One such approach is convex relaxation: basis pursuit (Chen et al., 1998) or LASSO (Tibshirani, 1996), which replaces the zero-norm $\| \cdot \|_0$ by the one-norm $\| \cdot \|_1$.

In our work, we use orthogonal matching pursuit (OMP) (Pati et al., 1993) to compute the sparse code x due to its efficiency and effectiveness, as shown in Algorithm 1. As a greedy-style algorithm, OMP selects the codeword best correlated with the current residual at each iteration, which is the reconstruction error remaining after the codewords chosen thus far. At the first iteration, this residual is exactly the observation y. 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. Figure 1 visualizes how the OMP algorithm works for the sparsity level $K = 4$. As can been seen, OMP quickly finds the sparse code x and the associated codewords that approximately reconstructs the image patch y. More codewords significantly improve

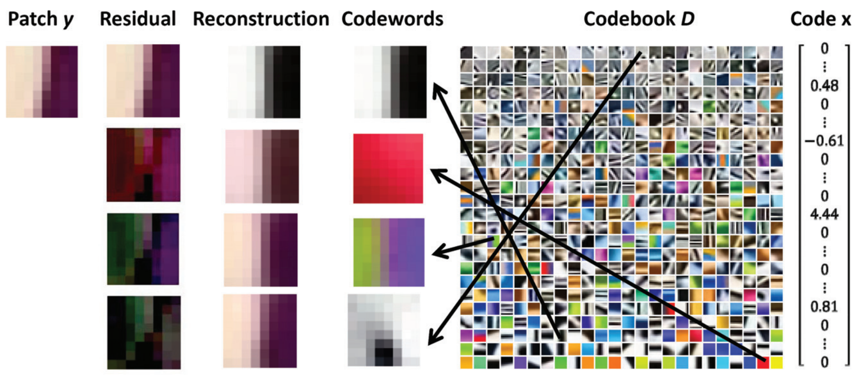


Fig. 1. Visualization of orthogonal matching pursuit. From the *leftmost* column to the *rightmost* column: an image patch, residual from zero, one, two and three codewords (from top to bottom), reconstructions from one, two, three and four codewords, the selected codewords at the first, second, third and fourth iterations, the learned codebook on the RGB channel and the computed sparse code.

the quality of reconstructions. When four codewords are selected, the reconstructed image patch is almost identical to the original one.

Figure 2 shows an example of an RGB-D image pair along with reconstructions achieved for different levels of sparsity. The shown results are achieved by non-overlapping 5×5 reconstructed patches based on the learned codebooks visualized in Figure 3, below. As can been seen, a sparsity level of $K = 5$ achieves results that are almost indistinguishable from the input data, indicating that this technique could also be used for RGB-D compression (Ruhnke et al., 2013), contrary to Ruhnke et al. (2010). For object recognition, the sparse codes become the features representing images, or patches.

The major advantages of OMP are its speed and its ease of implementation. Despite its simplicity, empirical evidence suggests that OMP works well in terms of approximation performance (Tropp, 2004). In the past few years, there has been significant interest in theoretical analysis of the OMP algorithm (Tropp and Gilbert, 2007; Jain et al., 2011; Zhang, 2011). Tropp and Gilbert (2007) demonstrated theoretically and empirically that OMP can recover a signal with K nonzero entries in dimension M with large probability, given $O(K \ln M)$ random linear measurements of that signal. Zhang (2011) commented that basis pursuit (Chen et al., 1998), i.e. a one-norm based convex optimization approach, is slightly more favorable in terms of its dependency on the lower strong convexity constant, while OMP is more favorable in terms of its dependency on the upper strong convexity constant. Since the OMP algorithm is faster and easier to implement, it is an attractive alternative to convex relaxation approaches for image recovery problems.

Recently, more and more attention has been focused on studying OMP under the restricted isometry property (Davenport and Wakin, 2010; Livshitz, 2010; Jain et al., 2011; Mo and Shen, 2012). To see, let δ_{K+1} be the restricted isometry constant of the measurement matrix (here codebook). Davenport and Wakin (2010) proved that $\delta_{K+1} < \frac{1}{3\sqrt{K}}$ is sufficient for OMP to uniformly recover any K-sparse signal in K iterations. Mo and Shen (2012) further improved the above sufficient condition and showed that $\delta_{K+1} < \frac{1}{\sqrt{K+1}}$ is sufficient for OMP to recover any K-sparse vector in K iterations. Both above results require $O(K^2)$ measurements for the recovery of a K-sparse vector. Livshitz (2010) proved that K-sparse vectors can be recovered via OMP by essentially a lesser number of measurements $O(K^{1.6} \ln M)$. The results are strong from the theoretical viewpoint and provide an explanation why OMP works well in practice. In our case, K is much smaller than the number of measurements H and OMP is expected to find a good solution.

In our HMP, OMP is used to efficiently compute the sparse codes for image patches at the first layer and pooled sparse codes at the second layer. In our setting, a large number of image patches share the same codebook and the total computational cost of OMP can be reduced by a batch version that keeps some quantities in memory (Davis et al., 1997; Rubinstein et al., 2008; Bo et al., 2011c). We analyze the batch versions of OMP in details in Section 3.4. We first turn out attention to learning codebooks from image data.

3.2. *Codebook learning via K-SVD*

K-SVD (Aharon et al., 2006) is a popular codebook learning approach that learns an overcomplete codebook from a set of observations with sparsity constraints. The approach has lead to the state-of-the-art results in various applications such as image denoising (Elad and Aharon, 2006b), face image compression (Bryt and Elad, 2008), texture segmentation (Mairal et al., 2008a), and contour detection (Ren and Bo, 2012). K-SVD learns codebooks $D = [d_1, \ldots, d_m, \ldots, d_M] \in R^{H\times M}$ and the associated sparse codes $X = [x_1, \ldots, x_n, \ldots, x_N] \in R^{M\times N}$ from a matrix $Y = [y_1, \ldots, y_n, \ldots, y_N] \in R^{H\times N}$ of observed data by minimizing the overall reconstruction error

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

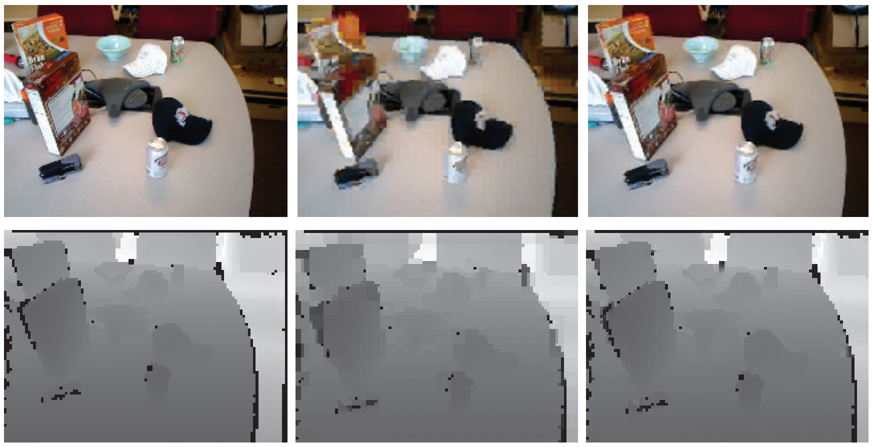


Fig. 2. Reconstructed images using the learned codebooks. *Left:* Original RGB and depth images. *Middle*: Reconstructed RGB and depth images using only two codewords per 5 × 5 patch. *Right:* Reconstructions using five codewords.

Algorithm 2: Codebook learning via K-SVD

```
1. Input: Data matrix $Y$, and the desired sparsity level $K$
2. Output: Codebook $D$
3. Initialize the codebook $D$ with Discrete Cosine Dictionary
4. Alternate between
5.     Computing sparse codes using OMP and the current
       codebook $D$
6.     Updating codebook $D$ via SVD based on computed
       sparse codes
7. End
```

Here, N is the size of the training data, 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 non-zero entries. When the sparsity level is set to 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 (2) in an alternating manner as outlined in Algorithm 2. During each iteration, the current codebook D is used to encode the data Y by computing the sparse code matrix X. Then, the codewords of the codebook are updated one at a time, resulting in a new codebook. This new codebook is then used in the next iteration to recompute the sparse code matrix followed by another round of codebook update.

Computing the sparse code matrix via orthogonal matching pursuit: Given the current codebook D, optimizing the sparse code matrix X is decoupled into N encoding problems, i.e. equation (1); one for each data item y. Orthogonal matching pursuit is used to compute the sparse code x of y, as discussed in Section 3.1.

Updating the codebook via singular value decomposition: Given the sparse code matrix X, the codebook D is optimized sequentially via singular value decomposition (SVD). In the mth step, the mth codeword and its sparse codes can be computed by performing SVD of the residual matrix corresponding to that codeword

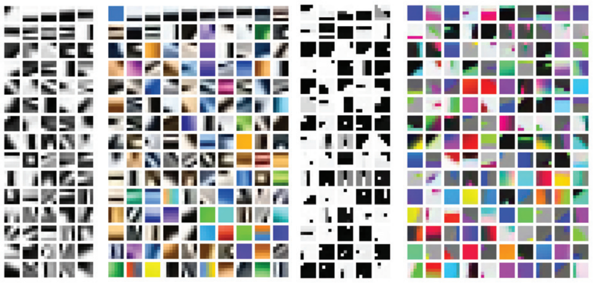


Fig. 3. Codebooks learned for different channels. From *left* to *right*: Grayscale intensity, RGB, depth, 3D surface normal (three normal dimensions color coded as RGB). The codeword sizes are 5 × 5 × 1 for grayscale intensity and depth values, and 5 × 5 × 3 for RGB and surface normal values. Codebook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

$$\|Y - DX\|_F^2 = \|Y - \sum_{i\neq m} d_i x_i^\top - d_m x_m^\top\|_F^2 = \|R_m - d_m x_m^\top\|_F^2 \quad (3)$$

where $x_i^\top$ are the rows of X, and $R_m = Y - \sum_{i\neq m} d_i x_i^\top$ is the residual matrix for the mth codeword. This matrix 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 mth codeword. It can be shown that each iteration of codebook learning followed by codebook updating decreases the reconstruction error (2). In practice, K-SVD converges to good codebooks for a wide range of initializations (Aharon et al., 2006).

In our HMP, K-SVD is used to learn codebooks 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). Figure 3 visualizes the learned codebooks in the first layer for four channels: grayscale and RGB for RGB images, and depth and surface normal for depth images. As can been seen, the learned codebooks have very rich appearances and include separated red, green, and 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.

It is helpful to consider both RGB and grayscale channels since our codebooks are learned in a hierarchical manner. In the first layer, the codebook learned on the RGB channel contains the most recognition cues captured by the codebook learned on the grayscale channel. However, in the second layer, each codeword learned on the RGB channel includes both color and non-color recognition cues while the one learned on the grayscale channel only includes non-color recognition cues. This leads to complementary features and boosts the recognition accuracy significantly, particularly for category recognition, which will be further demonstrated in the experiments.

We treat the sparsity level K, the codebook size N, and the codeword dimensionality H (uniquely determined by the patch size given the codebook size) as hyperparameters and search for them by ten-fold cross validation on training data. In the experiments, we show how category recognition changes with the sparsity level and the patch size in Figure 9, below. We found that the recognition accuracy is quite robust to these parameters and a set of fixed values seem to work well on different benchmarks.

3.3. *Architecture of hierarchical matching pursuit*

In object recognition, images are frequently modeled as unordered collections of local patches, i.e. bag of patches. Such models are flexible, and the image itself can be considered as a special case (bag with one large patch). Traditional bag-of-patches models introduce invariance while completely ignoring spatial information of patches that is generally useful for visual recognition. Spatial pyramid bag-of-patches models (Lazebnik et al., 2006) overcome the above problem by organizing patches into spatial cells at multiple levels (pyramid) and then concatenating features from spatial cells into one feature vector. Such models effectively balance the importance of invariance and discriminative power, leading to much better performance than simple bags. Spatial pyramid modeling is a compelling strategy for unsupervised feature learning, because of a number of advantages: (1) bag-of-patches virtually generate a large number of training samples (patches) for learning algorithms and decrease the chance of overfitting; (2) local invariance and stability of the learned features are increased by pooling features in spatial cells.

Hierarchical matching pursuit builds feature hierarchies by applying the orthogonal matching pursuit encoder recursively (Figure 4). The encoder consists of three modules: orthogonal matching pursuit, spatial pyramid max pooling, and contrast normalization. To balance invariance and spatial information, we aggregate sparse codes at each layer by spatial pyramid max pooling followed by contrast normalization. Our approach can be viewed as a dual-layer spatial pyramid bag-of-patches model where patches are represented as sparse codes. In the following, we discuss a dual-layer HMP network in detail.

First layer: The goal of the first layer is to generate pooled sparse codes (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 from a small neighborhood around this pixel (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:

$$F(C) = \left[\max_{j \in C} |x_{j1}|, \ldots, \max_{j \in C} |x_{jm}|, \ldots, \max_{j \in C} |x_{jM}| \right] \quad (4)$$

Here, j ranges over all entries in the cell, and x_{jm} is the mth component of the sparse code vector x_j of entry j. Note that $F(C)$ has the same dimensionality as the original sparse codes but may be less sparse due to the max pooling operation. The feature F_P describing a 16×16 image patch P are the concatenation of aggregated sparse codes in each spatial cell

$$F_P = \left[F(C_1^P), \ldots, F(C_s^P), \ldots, F(C_S^P) \right] \quad (5)$$

where $C_s^P \subseteq P$ is a spatial cell generated by spatial pyramid partitions over the image patches, and S is the total number of spatial cells. As an example, we visualize spatial cells generated by a three-level spatial pyramid pooling on a 16×16 image patch in Figure 5: 4×4, 2×2, and 1×1 cell sizes. In this example, the dimensionality of the pooled feature vector F_P is $(16 + 4 + 1)M$, where M is the size of the codebook (see also Figure 4). The main idea behind spatial pyramid pooling is to allow the features F_P to encode different levels of invariance to local deformations (Lee et al., 2009; Yang et al., 2009; Bo et al., 2011c), thereby increasing the discrimination of the features. An alternative way to capture fine-grained cues is to use a multipath architecture, as recently investigated in Bo et al. (2013).

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 to make sure the denominator is larger than zero. Since the magnitude of sparse codes varies over a wide range due to

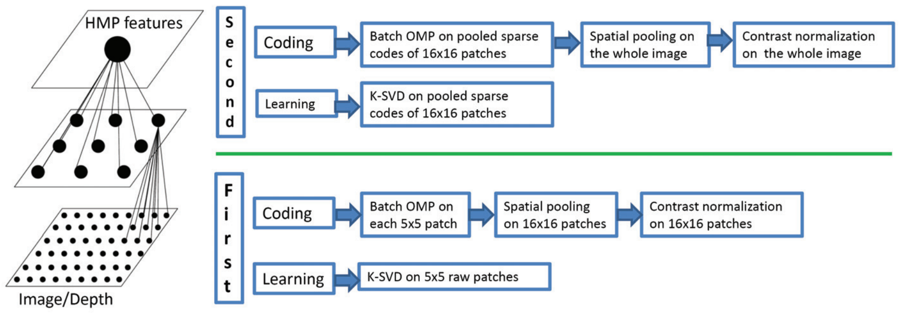

HMP features
Second
Coding
Batch OMP on pooled sparse codes of 16x16 patches
Spatial pooling on the whole image
Contrast normalization on the whole image
Learning
K-SVD on pooled sparse codes of 16x16 patches
First
Coding
Batch OMP on each 5x5 patch
Spatial pooling on 16x16 patches
Contrast normalization on 16x16 patches
Learning
K-SVD on 5x5 raw patches
Image/Depth

Fig. 4. 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 and contrast normalization. The second layer encodes these feature vectors using orthogonal matching pursuit and codebooks learned from sampled feature vectors. Whole image features are then generated from the resulting sparse codes, again by spatial pyramid max pooling and contrast normalization.

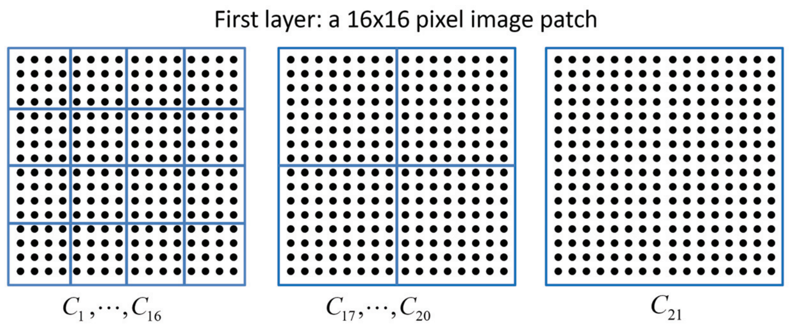


Fig. 5. Spatial pyramid partitions over a 16 × 16 image patch. Each black dot denotes sparse codes of a pixel that are computed on a 5 × 5 small patch around this pixel using orthogonal matching pursuit. *Left:* Level 2. The 16 × 16 image patch is partitioned into 4 × 4 = 16 spatial cells. Each cell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

local variations in illumination and occlusion, this operation makes the appearance features robust to such variations, as commonly done in the designed SIFT features. Contrast normalization often improves the recognition accuracy by about 2%. Our experiments suggest that a small but non-negligible $\varepsilon = 0.1$ works best for the first layer and leads to about 1% improvement over a negligible $\varepsilon = 0.0001$.

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 codebook 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 orthogonal matching pursuit, followed by spatial pyramid max pooling and contrast normalization (Figure 6). In this layer, however, we perform max pooling over both the sparse

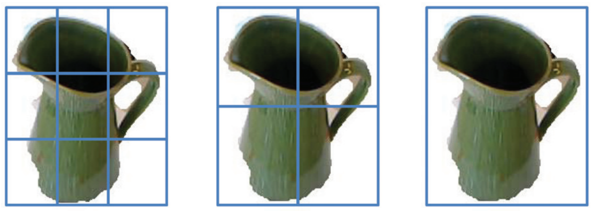


Fig. 6. Spatial pyramid partitions over the whole image. *Left:* Level 2. The whole image is partitioned into 3 × 3 = 9 spatial cells. Each cell is represented by the component-wise maximum of the sparse codes of 16 × 16 patches within the cell, $\overline{F}(C)$. *Middle:* Level 1. The whole image is partitioned into 2 × 2 = 4 spatial cells. *Right:* Level 0. The whole image is one spatial cell. The feature vector for the whole image is the concatenation of pooled sparse codes over C_1 through C_{14}.

codes and the patch level features computed in the first layer:

$$\overline{F}(C) = \left[\max_{j\in C} |z_{j1}|, \ldots, \max_{j\in C} |z_{jU}|, \max_{j\in C} |F_{j1}|, \ldots, \max_{j\in C} |F_{jV}| \right] \quad (6)$$

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 codebook, and V is given by the size of the patch level feature computed via (5). 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 the features.

The features of the whole image/object are the concatenation of the aggregated sparse codes within each spatial cell

$$\overline{F}_I = \left[\overline{F}(C_1^P), \ldots, \overline{F}(C_s^P), \ldots, \overline{F}(C_S^P)\right] \quad (7)$$

where $C_s^P \subseteq P$ is a spatial cell generated by spatial pyramid partitions over the whole image, and S is the total number of spatial cells. The image feature vector $\overline{F}_I$ is then normalized by dividing with its L_2 norm $\sqrt{\|\overline{F}_I\|^2 + 0.0001}$. In the higher layers, we use a negligible $\varepsilon = 0.0001$: a larger value decreases the accuracy but a smaller one works as well as $\varepsilon = 0.0001$.

Though we only consider two layers in this paper, it is straightforward to append more layers in a similar way to generate deep representations. The features from different layers encode different amounts of spatial invariance (mainly due to spatial pooling and contrast normalization), so additional layers could boost the accuracy further, as shown in Bo et al. (2013). It should be noted that HMP is a fully unsupervised feature learning approach: no supervision (e.g. object class) is required for codebook learning and feature coding. The feature vectors $\overline{F}_I$ of images/objects and their corresponding labels are then fed to classifiers to learn recognition models.

3.4. Fast orthogonal matching pursuit

In our recognition tasks, a large number of image patches share the same codebook. This allows us to dramatically reduce the total cost of orthogonal matching pursuit by pre-computing some quantities (Davis et al., 1997; Rubinstein et al., 2008; Bo et al., 2011c). The resulting algorithm is called batch orthogonal matching pursuit (BOMP), as shown in Algorithm 3. Note that the inversion of the matrix $(G_{II})^{-1}$ is updated by Cholesky factorization in an incremental way, with the cost $O(k^2)$ at iteration k, where G_{II} is the sub-matrix of G containing the rows indexed by I and the columns indexed by I.

The running time of the OMP algorithm is dominated by the codeword selection, whose total cost is $O(HMK)$. Our key finding is that the codeword selection doesn't require the residual r explicitly. Let $\alpha = D^\top r$, we have

$$\alpha = D^\top r = D^\top(y - D_I(D_I^\top D_I)^{-1} D_I^\top y) = \alpha^0 - G_I G_{II}^{-1}\alpha_I^0 \quad (8)$$

where we have set $\alpha^0 = D^\top y$ and $G = D^\top D$, and D_I denotes the sub-matrix of D containing the columns indexed

Algorithm 3: Batch orthogonal matching pursuit (BOMP)

```
1. Input: Codebook $D$, observation $y$, and the desired sparsity
      level $K$
2. Output: Sparse code $x$ such that $y \approx Dx$
3. Initialization: $I = \emptyset$, $\alpha = \alpha^0 = D^\top y$, $G = D^\top D$, and $x = 0$
4. For $k = 1 : K$
5.     Selecting the new codeword: $\overline{m} = \text{argmax}_m |\alpha_m|$
6.     $I = I \cup \overline{m}$
7.     Updating the sparse code: $x_I = G_{II}^{-1}\alpha_I^0$
8.     Updating $\alpha$: $\alpha = \alpha^0 - G_I x_I$
9. End
```

by I. Equation (8) indicates that if α^0 and G are pre-computed, the cost of updating α is $O(MK)$, instead of $O(MH)$ of orthogonal matching pursuit. Here, we have $K \leq H$ since the H codewords allow us to exactly reconstruct the image patches. The cost of searching sparse codes quickly dominates that of the pre-computation as the number of image patches increases. When using an overcomplete codebook, K is usually much less than H. In our experiments, K is less than 10 and H is several hundreds in the second layer of HMP, and we have observed significant speedup over the original orthogonal matching pursuit. Note that BOMP and OMP lead to the same solution and enjoy the same theoretical guarantees.

Pre-computing G takes $O(M^2H)$ time and $O(M^2)$ memory, which becomes infeasible for a very large codebook. To overcome this problem, we further propose batch tree orthogonal matching pursuit (BTOMP), as shown in Algorithm 4. BTOMP organizes the codebook using a dual-layer tree structure. K-means is used to group the codebook into T sub-codebooks $\{g_1, \ldots, g_T\}$. Each sub-codebook g_t is indexed by a k-means center $z_t \in Z = [z_1, \ldots, z_T]$. The codeword selection consists of two steps: (a) select the k-means center best correlated with the current residual and (b) select the codeword best correlated with the current residual from the sub-codebook indexed by the selected center. The algorithm reduces the cost of selecting codewords to $O(TK + \frac{MH}{T})$ and the memory of pre-computing G to $O(TM)$. If $T = M$, BTOMP exactly recovers the batch orthogonal matching pursuit. We have found that BTOMP works well in practice.

K-SVD training can be done in our case in the range of 30 min to a few hours. The cost is acceptable for most applications as the training procedure is usually done offline. In fact, the algorithms with the shorter running times are preferred for many real world applications. Figure 7 compares the computational cost of OMP, BOMP and BTOMP for a typical 150 × 150 object image. We consider the computational cost of the second layer of an HMP network for the depth channel since the second layer dominates the running time of HMP. All experiments are run on a single 3.30 GHz

Algorithm 4: Batch tree orthogonal matching pursuit (BTOMP)

```
1. Input: Codebook $D$, Centers $Z$, observation $y$, and sparsity
      level $K$
2. Output: Sparse code $x$ such that $y \approx Dx$
3. Initialization: $I = \emptyset, r = y, \alpha = \alpha^0 = Z^\top y, B = Z^\top D$,
      and $x = 0$
4. For $k = 1 : K$
5.     Choosing the sub-codebook $g_{\bar{t}}$: $\bar{t} = \text{argmax}_t |\alpha_t|$
6.     Selecting the new codeword: $\overline{m} = \text{argmax}_{m \in g_{\bar{t}}} |d_m^\top r|$
7.     $I = I \cup \overline{m}$
8.     Updating the sparse code: $x_I = (D_I^\top D_I)^{-1} D_I^\top y$
9.     Updating $\alpha$: $\alpha = \alpha^0 - B_I x_I$
10.    Computing the residual: $r = y - D_I x_I$
11. End
```

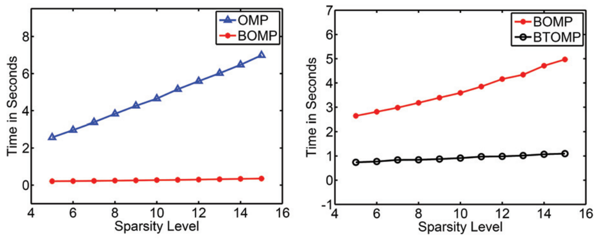


Fig. 7. Computational cost of OMP, BOMP and BTOMP. *Left:* 1000 codewords are used for comparisons. *Right:* 10,000 codewords are used for comparisons.

Intel Xeon CPU with a single thread. As can been seen from the left of Figure 7, BOMP takes about 0.3 s to compute sparse codes and is much faster than the flat OMP. For comparisons, we also run the efficient feature-sign search algorithm (Lee et al., 2006) to compute sparse codes from the L_1 based convex optimization. The procedure takes about 15 s under the comparable sparsity constraint with $K = 10$, significantly slower than BOMP. As can been seen from the right of Figure 7, BTOMP is more than three times faster than BOMP for larger codebooks. Our experiments suggest that BTOMP provides virtually the same performance as BOMP.

4. Experiments

We evaluate our RGB-D HMP 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. All images are resized to be no larger than 300×300 pixels with preserved ratio.

In the first layer, we learn dictionaries of size 75 with sparsity level 5 on 1,000,000 sampled 5×5 raw patches for grayscale and depth channels, and codebooks of size 150 on 1,000,000 sampled $5 \times 5 \times 3$ raw patches for RGB and normal channels using K-SVD. We remove the zero frequency component from raw patches by subtracting their mean and initialize K-SVD with overcomplete dictionaries generated by the Discrete Cosine Transform (Bo et al., 2011c). As can be seen from Figure 3, the learned codebooks over four channels capture a large variety of distinguishable structures from raw pixels. With the learned codebooks, 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. The resulting patch level features are $21 \times 75 = 1575$-dimensional vectors or $21 \times 150 = 3150$-dimensional vectors. Note that this architecture leads to fast computation of patch level features.

In the second layer, we learn codebooks of size 1000 with sparsity level 10 on pooled sparse codes sampled over 1,000,000 16×16 patches for each of four channels using K-SVD. We initialize K-SVD with randomly sampled pooled sparse codes (Bo et al., 2011c). With the learned codebooks, 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 of parameters fixed, even though it might further improve the accuracy via tuning these parameters per dataset using cross validation on the associated training data. With the learned HMP features, linear support vector machines (SVMs) are trained for recognition. For comparisons, we have experimented with linear logistic regression classifiers, and found that it is consistently worse than linear SVMs by about 2%. Linear SVMs are able to match the performance of nonlinear SVMs with the popular histogram intersection kernel (Maji et al., 2008) while being scalable to large datasets (Bo et al., 2011c).

4.1. *RGB-D object dataset*

The first dataset, called `RGBD`, contains 41,877 RGB-D images of 300 everyday objects taken from different viewpoints (Lai et al., 2011a). 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

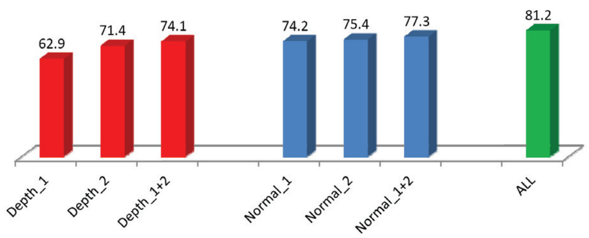


Fig. 8. Category recognition accuracy with different channels and different layers using depth images only. Depth_1 and Depth_2 denote single-layer and dual-layer HMP networks for the depth channel, respectively. Normal_1 and Normal_2 denote single-layer and dual-layer HMP networks for the surface normal channel, respectively. Depth_1+2 and Normal_1+2 denote the combination of the single-layer and the dual-layer HMP networks for the depth and the surface normal channels, respectively. "All" denotes the combination of all HMP networks.

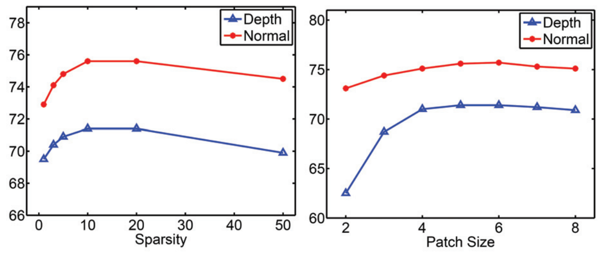


Fig. 9. Category recognition accuracy with different sparsity levels and different patch sizes. *Left:* Category recognition accuracy as a function of sparsity level in the second layer of HMP for the depth and surface normal channels, respectively. *Right:* Category recognition accuracy as a function of patch size in the second layer of HMP.

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 exhibit large changes in lighting conditions.

We distinguish between two types of object recognition tasks: category recognition and instance recognition. Category recognition is to determine the category name of a previously unseen object. Each object category consists of a number of different object instances. Instance recognition is to recognize known object instances. Following the experimental setting in Lai et al. (2011a), 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).

4.1.1. Object recognition using depth images. To provide a better understanding of our approach, we performed a detailed analysis of architecture and hyperparameters of HMP on depth images. We chose the depth channel because less attention has been paid to it in the deep learning community. In fact, similar conclusions hold for the color channel.

Though the deep learning community emphasized that multi-layer architectures are necessary for maximizing the accuracy, Coates and Ng (2011b) provide counterintuitive evidence by showing that single-layer networks work as well as multi-layer networks for object recognition on datasets they used. However, more recent work (Hinton et al., 2012; Le et al., 2012) shows that deep architectures trained on large-scale datasets significantly outperform single-layer networks and lead to large improvements over the state-of-the-art. This indicates that a large amount of training data might be the key to the success of deep networks.

To demonstrate that multi-layer HMP networks help RGB-D object recognition, we perform the experiments for category recognition with both single-layer and dual-layer HMP networks on the depth and surface normal channels. The results are reported in Figure 8. First of all, we observe that dual-layer HMP networks are always better than single-layer HMP networks: about 9% higher on the depth channel and about 1% higher on the surface normal channel. The combination of the single-layer and the dual-layer HMP networks outperform either one and further boosts the accuracy. The combination of all HMP networks achieves the highest recognition accuracy, significantly better than the best single HMP network.

These results are very encouraging and provide strong evidence for the importance of architectures of multiple channels and multiple layers. We believe that features from different layers provide complementary information that is the key for their success. Intuitively, features from the first layer capture basic structures of patches and are sensitive to spatial displacement. Features from the second layer are highly abstract, and robust to local deformations due to the introduction of spatial max pooling and contrast normalization in the first layer. These features are strong in their own right and their combination turns out to be much better than the best single one.

The sparsity level K and the patch size should be selected carefully to maximize the performance of HMP. A small K might lead to poor approximation and discard too much useful information, while an overly large K increases the computational cost and the risk of overfitting to unnecessary details. On the other hand, it is difficult to capture sufficient spatial context with a small patch size while an overly large patch size could incur severe over-smoothing on important structures.

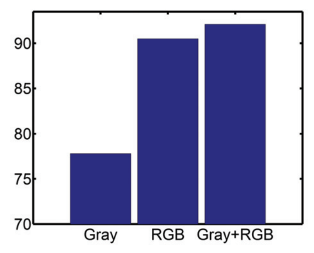
To understand the influence of these parameters, we vary the sparsity level and the patch size while fixing all other parameters to default values. We report the results in Figure 9. As can been seen, both small and large K values decrease accuracy, and the best test accuracy is achieved when K is around 10. The test accuracy becomes saturated when the patch size is larger than a threshold (here 5). The patch size of 5 gives the highest test accuracy among all patch sizes.

Spatial pyramid max pooling enables different levels of spatial context, and outperforms flat spatial max pooling (3×3 partitions) by about 2% in our experiments. We performed HMP with and without contrast normalization and found that contrast normalization improves recognition accuracy by about 3%, which indicates the importance of this component for learning robust features.

4.1.2. Object recognition using both RGB and depth images. We compare HMP with the baseline (Lai et al., 2011a), fast point feature histograms (Aldoma et al., 2012), kernel descriptors (Bo et al., 2011b), and convolutional k-means descriptors (CKM Desc) (Blum et al., 2012) in Table 1. Fast point feature histograms capture the relative orientation of normals, as well as distances, between point pairs. Note that these descriptors are standard features for 3D point clouds in the 3D point cloud library (Aldoma et al., 2012). Convolutional k-means descriptors adapt the k-means based feature learning approach proposed by Coates and Ng (2011b) to the RGB-D setting. The recognition systems developed in Lai et al. (2011a), Bo et al. (2011b), and Lai et al. (2012b) use a rich set of manually designed features. As can been seen, HMP outperforms all previous approaches for both category and instance recognition.

We performed additional experiments to shed light on different aspects of our approach for instance recognition. In Figure 10 (*left*), we show instance recognition accuracy using features on grayscale images and features on color images, and both. As can been 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 Figure 10. 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.

4.1.3. 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 $[0, 360°]$. Similar to instance recognition, we use the 30° and 60° viewing angle sequences as training data 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 (Lai et al., 2011b). 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.



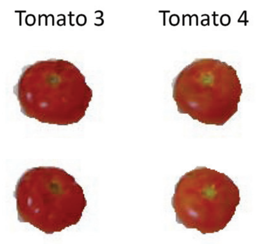


Fig. 10. *Left:* Instance recognition accuracy by using features on grayscale images (Gray), features on color images (RGB), and both (Gray + RGB). *Right:* Two tomato instances confused by our approach.

Table 2 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 (Lai et al., 2011b). 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 (Lai et al., 2011b). 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 Figure 11. 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 the same appearances.

4.1.4. Hierarchical segmentation based object detection. So far, our evaluation focused on object recognition, in which the object of interest occupies major portions of an image, or image segment. This setting applies, for instance, to robot manipulation tasks in which objects can easily be segmented from a scene, such as objects placed sufficiently far from each other on a table top. However, in many robotics tasks the location, or even presence, of an object is not known in advance and the object might be partially occluded by other objects. This is the more challenging object detection problem.

Table 1. Comparisons with the baseline (Lai et al., 2011a), fast point feature histograms (Aldoma et al., 2012), kernel descriptors (Bo et al., 2011b), and convolutional k-means descriptor (Blum et al., 2012).

RGBD	Category			Instance		
Methods	RGB	Depth	RGB-D	RGB	Depth	RGB-D
ICRA11 (Lai et al., 2011a)	74.3 ± 3.3	53.1 ± 1.7	81.9 ± 2.8	59.3	32.3	73.9
FPFH (Aldoma et al., 2012)	/	56.0 ± 1.5	/	/	/	/
CKM Desc (Blum et al., 2012)	/	/	86.4 ± 2.3	82.9	/	90.4
Kernel descriptors (Bo et al., 2011b; Lai et al., 2012b)	80.7 ± 2.1	80.3 ± 2.9	86.5 ± 2.1	90.8	54.7	91.2
HMP	82.4 ± 3.1	81.2 ± 2.3	87.5 ± 2.9	92.1	51.7	92.8

Table 2. Pose estimation error (in degrees) and running time (in seconds) comparison against several approaches using kernel descriptors as features. 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 Lai et al. (2011b). 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.

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
HMP	20.0	18.7	18.0	53.6	47.5	44.8	0.51

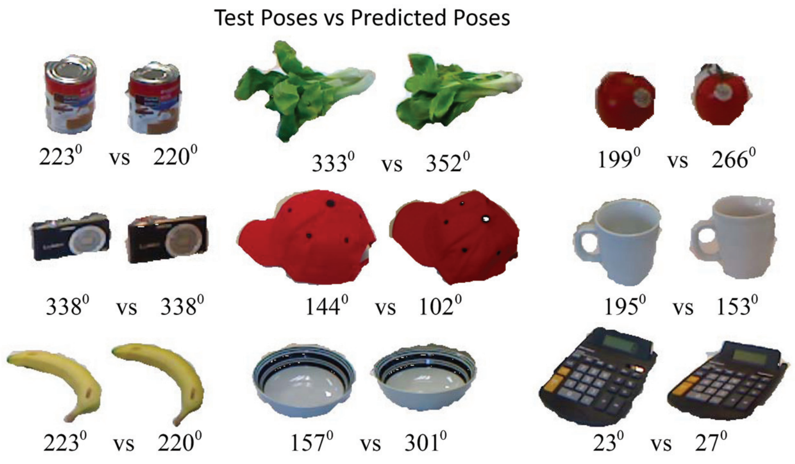


Fig. 11. Test images and the best matched angles in the training data using HMP features.

In this section we demonstrate the applicability of our hierarchical feature learning approach to object detection. The classical object detection pipeline is based on brute-force, for instance, sliding window approaches. By extracting fixed-size image rectangles at multiple locations and scales, sliding window approaches reduce object detection to a binary classification problem, i.e. “is the image rectangle an object or background?” Here, we consider an alternative segmentation based detection pipeline in order to fully leverage the strengths of depth information. Recent research along this line has achieved impressive results in the PASCAL VOC detection and segmentation challenge (Van de Sande et al., 2011; Carreira et al., 2012) and RGB-D scene labeling (Ren et al., 2012). Instead of classifying image rectangles, segmentation based detection approaches classify the segments generated by a segmentation algorithm into object or background segments. The pipeline is particularly interesting for us since depth information is available and it makes segmentation much easier.

To generate segments from an RGB-D scene, we follow contour based segmentation approaches (Arbelaez et al., 2011): first, detect contours of the scene and then convert them into a hierarchical segment tree. We use recently introduced sparse code gradients (SCG) for contour detection due to its proven performance and ability to deal with RGB-D images (Ren and Bo, 2012). SCG applies K-SVD to learn dictionaries on RGB and depth channels and then

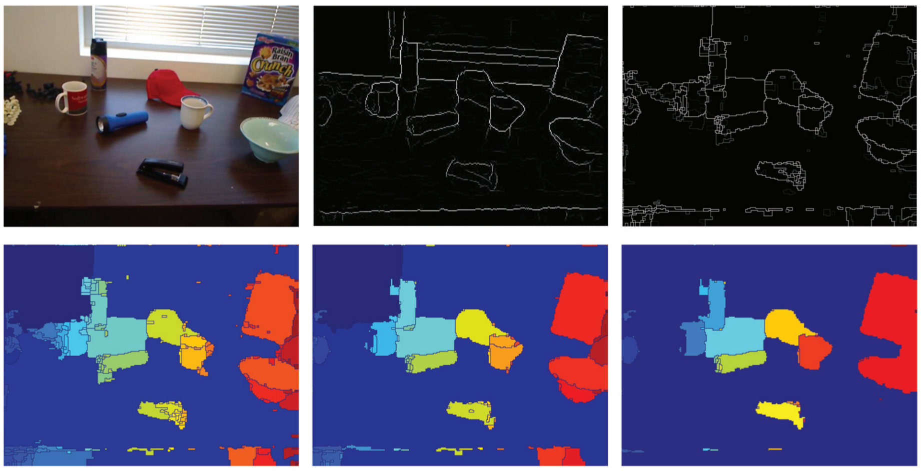


Fig. 12. Segmentation Results. *Top-left:* Original RGB image. *Top-middle*: SCG contours. Both RGB and depth channels are used. *Top-right:* Ultrametric contour map (UCM) generated from SCG contours. *Bottom* Segment map generated by thresholding UCM at three different levels. Different segments are shown in different colors.

uses OMP to encode rich contour cues by computing sparse codes on oriented local neighborhoods in multiple scales and pooling them together. SCG provides an elegant way to measure local contrasts and improves the previous best contour detection algorithm, global probability of boundary (gPb) (Arbelaez et al., 2011) by a large margin (Ren and Bo, 2012).

We extract hierarchical segments from an ultrametric contour map (UCM) (Arbelaez et al., 2011) converted from SCG contours. UCM defines a duality between closed, non-self intersecting weighted contours and a hierarchy of regions, and its values reflect the contrast between neighboring regions. The hierarchy is constructed by a greedy, graph-based region merging algorithm, and segments at different scales can be retrieved by thresholding the UCM at different levels. Such nested collection of segments allows object classifiers to select the object segments from multiple levels and dramatically increases the accuracy of object detection.

We visualize SCG contours, SCG UCM, and the segments at different levels on an RGB-D frame from the RGB-D Table Scene (Lai et al., 2011a) in Figure 12. As can been seen, SCG successfully detects most contours in the scene, which again demonstrates the effectiveness of sparse code features. Contour detection is quite challenging in this scene due to complex lighting conditions and object shadows. Depth information plays an important role in good performance. SCG UCM are a collection of closed weighted contours, as shown in the middle of Figure 12. The segments inferred from SCG UCM are shown in the bottom of Figure 12. We observe that different objects prefer different scale levels. For instance, the segmentation in the middle is best for the cereal box while the one in the right is best for coffee mugs, suggesting the benefits of hierarchical segmentation.

For object detection, we compute HMP features over bounding boxes of segments of size at least 40 $\times$ 40 pixels. Segments smaller than 24 $\times$ 24 pixels are not considered. All other segments are padded on each side of the segment by 8 pixels to keep edge information. We then run a linear SVM for each object instance and classify the segments into objects or background. In addition to object instance class, we also include a background class consisting of the segments generated from background scenes. Similarly to sliding window detection, we apply non-maximum suppression to eliminate redundancy when multiple object segments have large spatial overlaps.

We evaluate our segmentation based object detection on the RGB-D Table Scene, one of the most challenging scenes in Lai et al. (2011a). The whole video sequence consists of 125 RGB-D frames and contains seven object instances from five object categories: bowl, cap, cereal box, coffee mug and soda can. Note that flashlight, stapler and spray can are considered as background for this task. We report our object instance detection results in Figure 13. As can been seen, our approach gives not only the object bounding boxes (*left*) but also the object boundaries (*middle*). All target object instances are detected, except the highly occluded white cap (*middle right*). In the right of Figure 13, we show the precision-recall curves. Our average precision is about 10 absolute% higher than the previous sliding window detectors (Lai et al., 2011a). The missing recall is mostly because the majority of objects are occluded or not covered by a single segment.

This experiment demonstrates that HMP can be applied successfully to object detection, which is highly relevant to many robot manipulation and perception tasks.

4.2. *Willow and 2D3D datasets*

We additionally evaluated HMP on two other publicly available RGB-D recognition datasets, without further tuning any of its parameters. The first dataset, `2D3D`, consists of 156 object instances organized into 14

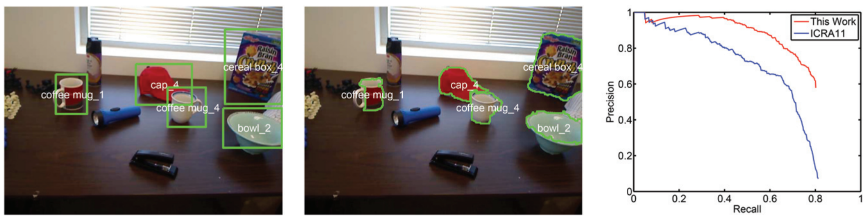


Fig. 13. Detection Results. *Left:* Detected object bounding boxes and their instance names. *Middle*: Detected object boundaries and their instance names. Note that flashlight, stapler and spray can are background for this task. *Right:* Precision-recall curves of our HMP detector and the sliding window detector (Lai et al., 2011a).

categories (Browatzki et al., 2011). The authors of this dataset 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, `Willow`, contains objects from the Willow and Challenge datasets for training and testing, respectively (Tang et al., 2012) (Figure 14). 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 (Tang et al., 2012). Note that `2D3D` and `Willow` only contain highly textured objects.

We report the results of HMP in Table 3. Following the experimental setting in Browatzki et al. (2011), HMP yields 91.0% accuracy for category recognition, much higher than the 82.8% reported in Browatzki et al. (2011). Learning models on training data from the Willow dataset and testing them on the training data from the Challenge dataset (Tang et al., 2012), HMP achieves higher precision/recall than the system proposed in (Tang et al., 2012), which won the 2011 Perception Challenge organized by Willow Garage. Note that this system is designed for textured objects and has not been evaluated on untextured objects such as those found in the `RGBD` dataset.

4.3. *Machine learning and vision datasets*

In this section, we evaluate the HMP features on 2D images only for object recognition and scene recognition tasks frequently considered in the machine learning and computer vision communities. With the proposed dual-layer architecture, HMP outperforms the state-of-the-art on standard learning and vision benchmarks.

4.3.1. Object recognition `STL-10`. introduced by Coates et al. (2011) is a standard dataset for evaluating feature learning approaches. We used the same architecture for these datasets as for the RGB-D datasets. The codebooks 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 Coates et al. (2011), we train linear SVMs on 1000 images and test on 8000 images using our HMP features and compute the averaged accuracy over 10 folds pre-defined by the authors.

We report the results in Table 4. As can been seen, HMP achieves much higher accuracy than the receptive field learning algorithm (Coates and Ng, 2011a) that beats many types of deep feature learning approaches as well as single-layer sparse coding on top of SIFT (Coates and Ng, 2011b). In addition, HMP outperforms discriminative sum-product networks (SPNs), a recently introduced deep learning model (Gens and Domingos, 2012) that is trained in a supervised manner using an efficient backpropagation-style algorithm.

4.3.2. Scene recognition. Understanding scenes is an important problem in robotics and has been a fundamental research topic in computer vision for a while. We evaluate HMP on the popular `MITScenes-67` dataset (Quattoni and Torralba, 2009). This dataset contains 15,620 images from 67 indoor scene categories. All images have a minimum resolution of 200 pixels in each axis. Indoor scene recognition is very challenging as the intra-class variations are large and some scene categories are similar to one another. We used the same architecture for these datasets as for RGB-D datasets. The codebooks are learned on both RGB and grayscale channels and the final features are the concatenation of HMP features from these two channels.

Following standard experimental setting (Pandey and Lazebnik, 2011), we train models on 80 images and test on 20 images per category on the pre-defined training/test split by the authors. As can been seen in Table 5, HMP achieves higher accuracy than many state-of-the-art algorithms: GIST (Quattoni and Torralba, 2009), Spatial Pyramid Matching (SPM) (Pandey and Lazebnik, 2011), Deformable Parts Models (DPM) (Felzenszwalb et al., 2010), Object Bank (Li et al., 2010), Reconfigurable Models (RBoW) (Parizi et al., 2012), and even the combination of SPM, DPM, and color GIST (Pandey and Lazebnik, 2011). The HMP features on both RGB and grayscale channels achieve higher accuracy than those on either grayscale channel only (41.8%) or RGB channel only (43.9%).

5. Conclusions

We have proposed HMP for learning multiple level sparse features from raw RGB-D data. HMP learns codebooks

Fig. 14. Ten of the 35 textured household objects from the Willow dataset.

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

`2D3D`	Category recognition			`Willow`	Instance recognition
Methods	RGB	Depth	RGB-D	Methods	Precision/recall
ICCVWorkshop (Browatzki et al., 2011)	66.6	74.6	82.8	ICRA12 (Tang et al., 2012)	96.7/97.4
HMP	86.3	87.6	91.0	HMP	97.4/100.0

Table 4. Comparisons with the previous results on `STL-10`.

MCML (Kiros and Szepesvri, 2012)	54.2 ± 0.4	Sparse Coding (Coates and Ng, 2011b)	59.0 ± 0.8	SPNs (Gens andDomingos, 2012)	62.3
VQ (Coates et al., 2011)	54.9 ± 0.4	Learned RF (Coates and Ng, 2011a)	60.1 ± 1.0	HMP	64.5 ± 1.0

Table 5. Comparisons with the previous results on `MITScenes-67`.

GIST (Quattoni and Torralba, 2009)	22.0	SPM (Pandey and Lazebnik, 2011)	34.4	RBoW (Parizi et al., 2012)	37.9
GIST-color (Pandey and Lazebnik, 2011)	29.7	Sparse Coding (Bo et al., 2011c)	36.9	DPM + Gist-color + SPM (Pandey and Lazebnik, 2011)	43.1
DPM (Felzenszwalb et al., 2010)	30.4	Object Bank (Li et al., 2010)	37.6	HMP	47.6

at each layer via sparse coding on a large collection of image patches, or pooled sparse codes. With the learned codebooks, HMP builds feature hierarchies using orthogonal matching pursuit, spatial pyramid pooling and contrast normalization. We have proposed the batch versions of orthogonal matching pursuit to significantly speed up the computation of sparse codes at runtime. Our algorithms are scalable and can efficiently handle full-size RGB-D frames. HMP consistently outperforms state-of-the-art techniques on standard benchmark datasets. These results are extremely encouraging, indicating that current recognition systems can be significantly improved with the learned features.

We believe this work has the potential to significantly improve robotics capabilities in object recognition, targeted object manipulation, and semantic scene understanding. An extensive evaluation shows that HMP outperforms designed features by a large margin for recognizing pre-segmented objects. Our experiments also show that HMP can significantly improve view-based pose estimation of objects, which has direct applications in robot grasping tasks. The results on object detection demonstrate furthermore that HMP can be applied not only to pre-segmented objects, but also to more general settings in which objects may occlude each other and occupy only small fractions of a scene. HMP can furthermore increase the capabilities of robots to understand commands given in natural language, since people often refer to objects by their names and visual attributes, as recently investigated in Sun et al. (2013).

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 submission is an extension of our previous work presented at NIPS 2011 (Bo et al., 2011c) and ISER 2012 (Bo et al., 2012).

Funding

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

References

Aharon M, Elad M and Bruckstein A (2006) K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing* 54(11): 4311–4322.

Aldoma A, Marton Z, Tombari F, et al. (2012) Tutorial: Point cloud library: three-dimensional object recognition and 6 DOF pose estimation. *IEEE Robotics and Automation Magazine* 19: 80–91.

Anand A, Koppula H, Joachims T, et al. (2012) Contextually guided semantic labeling and search for 3D point clouds. *International Journal of Robotics Research* 32(1): 19–34.

Arbelaez P, Maire M, Fowlkes C, et al. (2011) Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33(5): 898–916.

Bay H, Ess A, Tuytelaars T, et al. (2008) Speeded-up robust features (SURF). *Computer Vision and Image Understanding* 110(3): 346–359.

Blum M, Springenberg J, Wülfing J and Riedmiller M (2012) A learned feature descriptor for object recognition in RGB-D data. In: *Proceedings of IEEE international conference on robotics and automation (ICRA)*, Saint Paul, MN, 14–18 May 2012, pp. 1298–1303.

Bo L, Lai K, Ren X, et al. (2011a) Object recognition with hierarchical kernel descriptors. In: *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*, Providence, RI, 20–25 June 2011, pp. 1729–1736.

Bo L, Ren X and Fox D (2010) Kernel descriptors for visual recognition. In: *Proceedings of annual conference on neural information processing systems (NIPS)*, Lake Tahoe, NV, 3–6 December 2012, pp. 1729–1736.

Bo L, Ren X and Fox D (2011b) Depth kernel descriptors for object recognition. In: *Proceedings of IEEE/RSJ international conference on intelligent robots and systems (IROS)*, San Francisco, CA, 25–30 September 2011, pp. 821–826.

Bo L, Ren X and Fox D (2011c) Hierarchical matching pursuit for image classification: Architecture and fast algorithms. In: *Proceedings of annual conference on neural information processing systems (NIPS)*, Granada, Spain, 12–14 December 2011, pp. 2115–2123.

Bo L, Ren X and Fox D (2012) Unsupervised feature learning for RGB-D based object recognition. In: *Proceedings of international symposium on experimental robotics (ISER)*, Québec, Canada, 17–21 June 2012.

Bo L, Ren X and Fox D (2013) Multipath sparse coding using hierarchical matching pursuit. In: *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*, Portland, OR, 23–28 June 2013, pp. 660–667.

Boureau Y, Bach F, LeCun Y, et al. (2010) Learning mid-level features for recognition. In: *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*, San Francisco, CA, 13–18 June 2010, pp. 2559–2566.

Browatzki B, Fischer J, Graf B, et al. (2011) Going into depth: Evaluating 2D and 3D cues for object classification on a new, large-scale object dataset. In: *2011 IEEE international conference on computer vision workshops (ICCV)*, Barcelona, Spain, 6–13 November 2011, 1189–1195.

Bryt O and Elad M (2008) Compression of facial images using the K-SVD algorithm. *Journal of Visual Communication and Image Representation* 19(4): 270–282.

Candès E, Romberg J and Tao T (2006) Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory* 52(2): 489–509.

Carreira J, Li F and Sminchisescu C (2012) Object recognition by sequential figure-ground ranking. *International Journal of Computer Vision* 98(3): 243–262.

Chen S, Donoho D and Saunders M (1998) Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing* 20(1): 33–61.

Coates A, Lee H and Ng A (2011) An analysis of single-layer networks in unsupervised feature learning. In: *Proceedings of international conference on AI and statistics*. Ft. Lauderdale, FL, USA, 11–13 April 2011, pp. 215–223.

Coates A and Ng A (2011a) Selecting receptive fields in deep networks. In: *Proceedings of annual conference on neural information processing systems (NIPS)*, Granada, Spain, 12–14 December 2011, pp. 2528–2536.

Coates A and Ng A (2011b) The importance of encoding versus training with sparse coding and vector quantization. In: *Proceedings of international conference on machine learning (ICML)*, Bellevue, WA, 28 June–2 July 2011, pp. 921–928.

Dalal N and Triggs B (2005) Histograms of oriented gradients for human detection. In: *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*, San Diego, CA, 25 June 2005, pp. 886–893.

Davenport M and Wakin M (2010) Analysis of orthogonal matching pursuit using the restricted isometry property. *IEEE Transactions on Information Theory* 56(9): 4395–4401.

Davis G, Mallat S and Avellaneda M (1997) Adaptive greedy approximations. *Constructive Approximation* 13(1): 57–98.

Donoho D (2006) Compressed Sensing. *IEEE Transactions on Information Theory* 52(4): 1289–1306.

Elad M and Aharon M (2006a) Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing* 15(12): 3736–3745.

Elad M and Aharon M (2006b) Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing* 15(12): 3736–3745.

Felzenszwalb P, Girshick R, McAllester D, et al. (2010) Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32: 1627–1645.

Gens R and Domingos P (2012) Discriminative learning of sum-product networks. In: *Proceedings of annual conference on neural information processing systems (NIPS)*, Lake Tahoe, NV, 3–6 December 2012, pp. 3248–3256.

Henry P, Krainin M, Herbst E, et al. (2010) RGB-D Mapping: Using depth cameras for dense 3D modeling of indoor environments. In: *Proceedings of the 12th international symposium on experimental robotics (ISER)*, 18–21 December 2010, Delhi, India, Vol. 20, pp. 22–25.

Hinterstoisser S, Holzer S, Cagniart C, et al. (2011) Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In: *Proceedings of IEEE international* *conference on computer vision (ICCV)*, Barcelona, Spain, 6–13 November, pp. 858–865.

Hinton G, Osindero S and Teh Y (2006) A fast learning algorithm for deep belief nets. *Neural Computation* 18(7): 1527–1554.

Hinton G, Srivastava N, Krizhevsky A, et al. (2012) Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580.

Jain P, Tewari A and Dhillon I (2011) Orthogonal matching pursuit with replacement. *Proceedings of annual conference on neural information processing systems (NIPS)*, Granada, Spain, 12–14 December 2011, pp. 1672–1680.

Jarrett K, Kavukcuoglu K, Ranzato M, et al. (2009) What is the best multi-stage architecture for object recognition? In: *Proceedings of IEEE international conference on computer vision (ICCV)*, Kyoto, Japan, 29 September–2 October 2009, pp. 2146–2153.

Jiang Y, Lim M, Zheng C, et al. (2012) Learning to place new objects in a scene. *International Journal of Robotics Research* 31(9): 1021–1043.

Johnson A and Hebert M (1999) Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21(5): 433–449.

Kavukcuoglu K, Sermanet P, Boureau Y, et al. (2010) Learning convolutional feature hierarchies for visual recognition. In: *Proceedings of annual conference on neural information processing systems (NIPS)*, Vancouver, Canada, 6–8 December 2010, pp. 1090–1098.

Kiros R and Szepesvári C (2012) On linear embeddings and unsupervised feature learning. In: *ICML Representation Learning Workshop*, Edinburgh, UK, 26–30 June 2012.

Krizhevsky A, Sutskever I and Hinton G (2012) Imagenet classification with deep convolutional neural networks. In: *Proceedings of annual conference on neural information processing systems (NIPS)*, Lake Tahoe, NV, 3–6 December 2012, pp. 1106–1114.

Lai K, Bo L, Ren X, et al. (2011a) A large-scale hierarchical multi-view RGB-D object dataset. In: *Proceedings of IEEE international conference on robotics and automation (ICRA)*, Shanghai, China, 9–13 May 2013, pp. 1817–1824.

Lai K, Bo L, Ren X, et al. (2011b) A scalable tree-based approach for joint object and pose recognition. In: *Proceedings of AAAI conference on artificial intelligence (AAAI)*, San Francisco, CA, 7–11 August 2011.

Lai K, Bo L, Ren X, et al. (2012a) Detection-based object labeling in 3D scenes. In: *Proceedings of IEEE international conference on robotics and automation (ICRA)*, Saint Paul, MN, 14–18 May 2012, pp. 1330–1337.

Lai K, Bo L, Ren X, et al. (2012b) RGB-D object recognition: Features, algorithms, and a large scale benchmark. In: Fossati A, Gall J, Grabner H, et al. (eds) *Consumer Depth Cameras for Computer Vision: Research Topics and Applications*. London; New York: Springer, pp. 167–192.

Lazebnik S, Schmid C and Ponce J (2006) Beyond bags of features: Spatial Pyramid matching for recognizing natural scene categories. In: *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*, New York, 17–22 June 2006, pp. 2169–2178.

Le Q, Karpenko A, Ngiam J, et al. (2011) ICA with reconstruction cost for efficient overcomplete feature learning. In: *Proceedings of annual conference on neural information processing systems (NIPS)*, Granada, Spain, 12–14 December 2011, pp. 1017–1025.

Le Q, Ranzato M, Monga R, et al. (2012) Building high-level features using large scale unsupervised learning. In: *Proceedings of international conference on machine learning (ICML)*, Vancouver, Canada, 26–31 May 2013, pp. 8595–8598.

LeCun Y, Bottou YBL and Haffner P (1998) Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11): 2278–2324.

Lee H, Battle A, Raina R, et al. (2006) Efficient sparse coding algorithms. In: *Proceedings of annual conference on neural information processing systems (NIPS)*, Vancouver, Canada, 4–7 December 2006, pp. 801–808.

Lee H, Grosse R, Ranganath R, et al. (2009) Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: *Proceedings of international conference on machine learning (ICML)*, Montreal, Canada, 14–18 June 2009, pp. 609–616.

Li L, Su H, Xing E, et al. (2010) Object bank: A high-level image representation for scene classification and semantic feature sparsification. In: *Proceedings of annual conference on neural information processing systems (NIPS)*, Vancouver, Canada, 6–8 December 2010, pp. 1378–1386.

Livshitz E (2010) On efficiency of orthogonal matching pursuit. arXiv preprint arXiv:1004.3946.

Lowe D (2004) Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* 60: 91–110.

Mairal J, Bach F, Ponce J, et al. (2008a) Discriminative learned dictionaries for local image analysis. In: *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*, Anchorage, AK, 24–26 June 2008, pp. 1–8.

Mairal J, Bach F, Ponce J, et al. (2008b) Supervised dictionary learning. In: *Proceedings of annual conference on neural information processing systems (NIPS)*, Vancouver, Canada, 8–10 December 2008, pp. 1033–1040.

Maji S, Berg A and Malik J (2008) Classification using intersection kernel support vector machines is efficient. In: *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*, Anchorage, AK, 23–28 June 2008, pp. 1–8.

Mo Q and Shen Y (2012) A remark on the restricted isometry property in orthogonal matching pursuit. *IEEE Transactions on Information Theory* 58(6): 3654–3656.

Morisset B, Rusu RB, Sundaresan A, et al. (2009) Leaving flatland: Toward real-time 3D navigation. In: *Proceedings of IEEE international conference on robotics and automation (ICRA)*, Kobe, Japan, 12–17 May 2009, pp. 3786–3793.

Newcombe R, Davison A, Izadi S, et al. (2011) KinectFusion: Real-time dense surface mapping and tracking. In: *IEEE international symposium on mixed and augmented reality (ISMAR)*, Basel, Switzerland, 26–29 October 2011, pp. 127–136.

Olshausen B and Field D (1996) Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature* 381: 607–609.

Pandey M and Lazebnik S (2011) Scene recognition and weakly supervised object localization with deformable part-based models. In: *Proceedings of IEEE international conference on computer vision (ICCV)*, Barcelona, Spain, 6–13 November 2011, pp. 1307–1314.

Parizi SN, Oberlin J and Felzenszwalb P (2012) Reconfigurable models for scene recognition. In: *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*, Providence, RI, 16–21 June 2012, pp. 2775–2782.

Pati Y, Rezaiifar R and Krishnaprasad P (1993) Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In: *The twenty-seventh asilomar conference on signals, systems and computers*, Pacific Grove, CA, 1–3 November 1993, Vol. 1, pp. 40–44.

Quattoni A and Torralba A (2009) Recognizing indoor scenes. In: *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*, Miami, FL, 20–25 June 2009, pp. 413–420.

Ren X and Bo L (2012) Discriminatively trained sparse code gradients for contour detection. In: *Proceedings of annual conference on neural information processing systems (NIPS)*, Lake Tahoe, NV, 3–6 December 2012, pp. 593–601.

Ren X, Bo L and Fox D (2012) RGB-(D) scene labeling: Features and algorithms. In: *Proceedings of IEEE international conference on computer vision (ICCV)*, Providence, RI, 16–21 June 2012, pp. 2759–2766

Rubinstein R, Zibulevsky M and Elad M (2008) Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit. Technical report, CS Technion.

Ruhnke M, Bo L, Fox D, et al. (2013) Compact RGBD surface models based on sparse coding. In: *Proceedings of AAAI conference on artificial intelligence (AAAI)*, Bellevue, WD, 14–18 July 2013.

Ruhnke M, Steder B, Grisetti G, et al. (2010) Unsupervised learning of compact 3D models based on the detection of recurrent structures. In: *Proceedings of IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Taipei, Taiwan, 18–22 October 2010, pp. 2137–2142.

Shotton J, Fitzgibbon A, Cook M, et al. (2011) Real-time human pose recognition in parts from a single depth image. In: *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*, Providence, RI, 20–25 June 2011, pp. 1297–1304.

Socher R, Lin C, Ng A, et al. (2011) Parsing natural scenes and natural language with recursive neural networks. In: *Proceedings of international conference on machine learning (ICML)*, Bellevue, WA, 28 June–2 July 2011, pp. 129–136.

Sun Y, Bo L and Fox D (2013) Attribute based object identification. In: *Proceedings of IEEE international conference on robotics and automation (ICRA)*, Karlsruhe, Germany, 6–10 May 2013, pp. 2096–2103.

Tang J, Miller S, Singh A, et al. (2012) A textured object recognition pipeline for color and depth image data. In: *Proceedings of IEEE international conference on robotics and automation (ICRA)*, Saint Paul, MN, 14–18 May 2012, pp. 3467–3474.

Tibshirani R (1996) Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society—Series B (Methodological)* 58: 267–288.

Tropp J (2004) Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory* 50(10): 2231–2242.

Tropp J and Gilbert A (2007) Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory* 53(12): 4655–4666.

Van de Sande K, Uijlings J, et al. (2011) Segmentation as selective search for object recognition. In: *Proceedings of IEEE international conference on computer vision (ICCV)*, Barcelona, Spain, 6–13 November 2011, pp. 1879–1886.

Wang J, Yang J, Yu K, et al. (2010) Locality-constrained linear coding for image classification. In: *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*, San Francisco, CA, 13–18 June 2010, pp. 3360–3367.

Wright J, Yang A, Ganesh A, et al. (2009) Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31(2): 210–227.

Yang J, Wright J, Huang T, et al. (2008) Image super-resolution as sparse representation of raw image patches. In: *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*, Anchorage, AK, 23–28 June 2008, pp. 1–8.

Yang J, Yu K, Gong Y and Huang T (2009) Linear spatial pyramid matching using sparse coding for image classification. In: *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*, Miami, FL, 20–25 June 2009, pp. 1794–1801.

Yu K, Lin Y and Lafferty J (2011) Learning image representations from the pixel level via hierarchical sparse coding. In: *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*, Providence, RI, 20–25 June 2011, pp. 1713–1720.

Zeiler M, Taylor G and Fergus R (2011) Adaptive deconvolutional networks for mid and high level feature learning. In: *Proceedings of IEEE international conference on computer vision (ICCV)*, Barcelona, Spain, 6–13 November 2011, pp. 2018–2025.

Zhang T (2011) Sparse recovery with orthogonal matching pursuit under RIP. *IEEE Transactions on Information Theory* 57(9): 6215–6221.