Sparse Dictionary Learning for Identifying Grasp Locations

Ludovic Trottier Philippe Giguère Brahim Chaib-draa
Laval University, QC, CA
ludovic.trottier.1@ulaval.ca
{philippe.giguere, brahim.chaib-draa}@ift.ulaval.ca

Abstract

The ability to grasp ordinary and potentially never-seen objects is an important task in both domestic and industrial robotics. For a system to accomplish this, it must autonomously identify grasping locations by using information from various sensors, such as Microsoft Kinect 3D camera. Despite numerous progress, significant work still remains to be done for this task. To this effect, we propose a dictionary learning and sparse representation (DLSR) framework for representing RGBD images from 3D sensors in the context of identifying grasping locations. In contrast to previously proposed approaches that relied on sophisticated regularization or very large datasets, our derived perception system has a fast training phase and can work with small datasets. It is also theoretically founded for dealing with masked-out entries, which are common with 3D sensors. We contribute by presenting a comparative study of several DLSR approach combinations for recognizing and detecting grasp candidates on the standard Cornell dataset. Experimental results show a performance improvement of 1.69% in detection and 3.16% in recognition over current state-of-the-art convolutional neural network (CNN). Even though nowadays most popular vision-based approach is CNN, this suggests that DLSR is also a viable alternative with interesting advantages that CNN has not.

1. Introduction

In robotics, automating the grasping of ordinary objects is an important open problem, for which the grasp localization task in an essential element [10, 16, 19, 26, 27]. This task aims to determine the most convenient grasping locations, and is usually viewed as a vision-based detection problem [29]. One key principle of this detection is relying on a visual perception system that interprets light and depth information, known as RGBD images, for performing scene understanding. This involves elaborate visual tasks, such as scene segmentation, object categorization and visual disambiguation. Due to the complexity of these tasks, learning the proper feature representation of the surrounding environment from the RGBD images is essential. This is an important problem because the performance of the grasp localization system heavily depends on the quality of the extracted features.

Extracting the most convenient features from RGBD images is a challenging problem because RGBD images are complex in nature. One particular avenue that has been explored in previous works is employing Dictionary Learning and Sparse Representations (DLSR) [3, 17, 33]. DLSR is a *representation learning* framework that has received a lot of attention from the vision, signal and image processing communities in the last decade [37]. DLSR approaches aim at learning sparse feature representations where observations are expressed as sparse linear combinations of few atoms from a dictionary. Sparsity-based models are biologically-inspired and theoretically well-founded [31]. The assumption that observations admit a sparse representation has been verified in several previous vision-related works, such as object recognition [1, 34], face recognition [35], scene analysis [18] and image restoration [9, 21].

The sparsity assumption also holds in the context of RGBD images. For instance, [3] proposed a DLSR approach for RGBD object recognition, [17] performed 3D scene labeling and [33] tackled the problem of 3D contour detection. One key principle that has been observed is that DLSR is well-suited to handle the Microsoft Kinect depth information noise [36]. The mask noise model of missing 3D data can be particularly cumbersome. Objects with shiny surfaces often cause structured-light 3D cameras to fail which results in absence of information. DLSR allows decomposing the essential parts of the observation in a sparse way, making the input noise representation dense and easily modeled [1]. This makes DLSR well-adapted for managing complex noisy depth information in Kinect.

To the best of our knowledge, applying the DLSR paradigm for identifying grasping locations is currently lacking in grasping literature. Our contributions with this paper include the following:

- First, we propose a DLSR-based framework for learn-

ing and extracting useful information from RGBD images. We show that it has a fast training phase, can work with a small dataset and is theoretically well-founded for dealing with masked-out entries.
- Second, we demonstrate the applicability of our proposed framework for identifying grasp locations on the standard Cornell task [15], and compare it with other approaches in the literature.
- Finally, we present an empirical evaluations of several dictionary learning and feature coding approach combinations in order to understand the relationship between the two. We analyze which combinations are best suited for the task at hand by comparing performance, speed of training and the ability to be parallelized (either on CPU or GPU).

The rest of this paper is divided as follows. We make an overview of related works in section 2. We review DLSR in section 3 and present our proposed DLSR framework in section 4. We show our experiments in section 5 and discuss our results in section 6. We finally conclude in section 7.

2. Related Work

Several previous approaches relied on 3D simulations to learn good grasp locations [6, 12, 23, 25]. They are powerful but necessitate building the 3D physical model of each object in order to perform grasp localization. Requiring *a priori* the physical characteristics of the objects reduces their applicability for general purpose robots. These characteristics are rarely known in advance. With our approach, we seek to perform grasp localization without building complex physical models prior to the execution.

Recently, works in the field of Deep Learning have also been proposed. For recognizing potential grasp regions, [19] proposed a cascade of multi-layered perceptrons, while [32] proposed a deep convolutional neural network (CNN). One limitation of these approaches is that they rely on hand-designed regularization terms. [19] defined a mask-based term that reduces the network susceptibility to masked-out entries, and [32] used a structured penalty term to improve the multi-modal connections. With our proposed DLSR method, we intend to create a theoretically founded grasp localization model without resorting to a custom regularization. Other works also proposed using a CNN for predicting candidate grasp locations [14, 27]. One inconvenient of CNN is that they require several days for learning their millions of parameters. In an industrial context where datasets are small and new objects are regularly added, a fast and robust training phase is essential. Increasing the dataset size by repetitively adding more images from different viewpoints is an efficient strategy to make objects easier to grasp. This is only viable with a fast training phase, which our proposed DLSR method offers.

Some efforts have been dedicated into building grasp localization system from a Big Data perspective [13, 16, 20]. These works make use of hundreds of thousands of observations to train their approaches. While these initiatives are essential to reach the long-term goal of general purpose robotics, industrial contexts are often constrained in the amount of available data. A grasp localization approach that is able to train with only a handful of observations can be useful in early development stages. As we show in our experiments, our proposed approach achieved state-of-the-art performances by only training with the small Cornell dataset.

3. Dictionary Learning and Sparse Representation Framework

A standard DLSR method is divided into a *dictionary learning* phase, where a dictionary is trained to capture the latent structure of the data, and a *feature coding* phase, where the dictionary is used to transform raw observations into features. In this section, we review the basics of Dictionary Learning and Sparse Representation (DLSR) and introduce our framework for grasp localization.

3.1. Dictionary Learning

Given a set of n-dimensional observations $x^{(i)} \in \mathbb{R}^n$, our goal is to learn a dictionary $D \in \mathbb{R}^{n \times d}$ of d atoms ($D^{(j)}$ is the jth column) capturing the essential features of the given observations. Dictionary learning aims to learn D such that the sparsest linear combination $w^{(i)}$ of the dictionary atoms faithfully approximate each observation $x^{(i)}$. This problem can be formulated as follows:

$$D = \arg\min_{\bar{D}} \min_{\bar{w}^{(i)}} \sum_i \|\bar{D}\bar{w}^{(i)} - x^{(i)}\|_2^2 + \lambda \mathrm{Sp}(\bar{w}^{(i)}) \quad (1)$$

$$\text{subject to } \|\bar{D}^{(j)}\|_2 = 1, \forall j \,,$$

where Sp is a vector sparsity measure and $\|.\|_2^2$ is the squared euclidean norm. The quantity $\|\bar{D}\bar{w}^{(i)} - x^{(i)}\|_2^2$ to be minimized is called the *residual*, and is penalized by the sparsity constraint $\lambda \mathrm{Sp}(\bar{w}^{(i)})$. Hyper-parameter $\lambda > 0$ governs the *sparseness* of the weights and will be chosen by cross-validation.

One of the key principle of dictionary learning is that atoms $D^{(j)}$ are learned from the observations. Previously to DLSR, the standard practice was to use analytically predefined dictionaries such as Discrete Cosine Transform (DCT) or Wavelet Transform (WT). These fixed dictionaries were rarely flexible enough to well represent any data distribution, and only worked in specific scenarios [37]. Training a dictionary by minimizing eq. (1) significantly improves the sparsity of the weight vector solution $w^{(i)}$, as it is explicitly asked to fit the input data.

Minimizing (1) for both the dictionary D and the weights $w^{(i)}$ is however computationally expensive. This is due to

the interleaving dependency between D and all $w^{(i)}$, which is shown in (1) by the nested minimization. More viable alternatives rather employ an *iterative-alternative* scheme for finding a suitable approximative solution. The technique works as follows. First, initialize the dictionary D randomly. Second, minimize (1) with respect to the weights $w^{(i)}$ considering D fixed. Third, minimize (1), this time with respect to D considering the $w^{(i)}$ fixed. Finally, alternate step 2 and 3 until convergence. This optimization procedure has been widely used in previous works and has shown promising results [1, 21].

Dictionary learning can thus be decomposed into two part: 1. *dictionary optimization* with fixed weights, and 2. *weight optimization* with a fixed dictionary. Different methods have been proposed in the last decade for performing these two optimization problems. One of the most popular and generally adopted approach is Online Dictionary Learning (ODL) [21]. ODL is a *dictionary optimization* approach that can solve (1) given an appropriate *weight optimization* method. ODL can be used with either a $\text{Sp} = \ell_0$ or a $\text{Sp} = \ell_1$ sparsity measure. In the former case, the sparseness of the solution is defined as the number of non-zero entries, and in the later case, is defined as the sum of absolute values. We refer to ODL with a ℓ_0 sparsity measure as ODL-0, and ODL with a ℓ_1 sparsity measure as ODL-1.

Previous dictionary learning approaches required processing the whole data matrix X (containing in its rows the observations $x^{(i)}$) at each step of the training phase. This limited the applicability of the approaches to small dataset that could be fitted into memory. ODL rather learns the dictionary D in an online fashion, making use of only a small portion of the data at each step. It starts by drawing at random a batch of observations $x^{(i)}$, then applies the *weight optimization* method to extract the weights $w^{(i)}$, and finally updates the dictionary using block-coordinate gradient descent. In addition to having several local convergence guaranties [21], this method significantly reduces the amount of needed memory for processing the data.

Other alternatives exist to ODL. As part of our contributions with our paper, we also use other dictionary learning approaches and compare them to ODL. One is Gain Shape Vector Quantization (GSVQ) [11]. GSVQ as a similar formulation to ODL-0, but is different in the way it represents the weight vector $w^{(i)}$. It explicitly separates its norm (the gain) from its orientation (the shape), and split the *weight optimization* into a gain and a shape optimization. This parametrization has been shown to improve convergence in some cases [11].

We also tried learning a dictionary D without specifying a sparsity measures. One approach is to run a KMeans clustering algorithm on whitened observations and use as dictionary D the normalized centroids [5]. By combining the powerful ZCA-whitening decorrelation technique with KMeans ability to find relevant clusters in the data distribution, we can learn representative centroids that capture non-linear dependencies within the observations. We refer to this method as Normalized KMeans (NKM).

Another one is to randomly sample d whitened observations and use their normalized version as dictionary. This can be seen as a zero-cost effective approximation of KMeans centroids. We refer to this approach as Random Patches (RP). Finally, the last one is to sample d times the uniform distribution $U\left([-1,1]^n\right)$ and use the normalized vectors as dictionary atoms. Random dictionaries has been advertised in past works, but we show in our experiments that they always give the worst performances [28]. We refer to this method as Random (R).

3.2. Feature Coding

Executing any dictionary learning approaches presented in the previous section gives a dictionary D representing the latent structure of the data. The goal now is extracting relevant features from the observations using dictionary D. This step is called *feature coding*. One straightforward choice of a *feature coding* method is performing *weight optimization* and use the weight solution as features. This was the second part of *dictionary learning* as introduced in section 3.1. The task can be formulated as follows:

$$w^{(i)} = \arg\min_{\bar{w}^{(i)}} \|D\bar{w}^{(i)} - x^{(i)}\|_2^2 + \lambda \text{Sp}(\bar{w}^{(i)}),\ \forall i\,. \quad (2)$$

The sparsest weight solution $w^{(i)}$ that minimizes the residual $\|D\bar{w}^{(i)} - x^{(i)}\|_2^2$ is an accurate feature representation of observation $x^{(i)}$. The weight values reflect the importance of each atom $D^{(j)}$ in the linear combination. Large weight magnitudes indicate the presence of certain characteristics from which $x^{(i)}$ is composed. Observations that share these characteristics will have comparable weight vectors and will be neighbors in the feature space. This allows us learning classifiers on top of these features that more easily capture the relations between the observations.

Different approaches exist for solving (2) based on the sparsity measure Sp. In the case of a $\text{Sp} = \ell_0$ function, the standard method is Orthogonal Matching Pursuit (OMP) [24]. OMP is a greedy algorithm that iteratively updates the weight solution $w^{(i)}$ one position at a time by maximizing the correlation with the residual. It has the important property that once an atom has been given a weight value, it can no longer be changed. This guaranties that OMP will find a solution $w^{(i)}$ with the required number of non-zero entries $\ell_0(w^{(i)}) = \gamma$ in exactly γ steps.

In the case of a $\text{Sp} = \ell_1$ function, one of the standard methods is Least-Angle Regression (LARS) [8]. LARS works similarly to OMP as it updates the weight solution $w^{(i)}$ in an iterative fashion. It however differs in its way of updating the solution. Instead of computing the weight

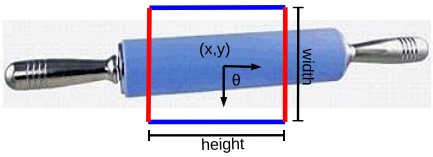

Figure 1. The grasp rectangle is a five-dimensional grasp representation. The 2D rectangle is fully determined by its center coordinates (x, y), width, height and its angle θ from the x-axis. The blue edges indicate the gripper plate location and the red edges show the gripper opening, prior to grasping.

value one atom at a time, it rather performs a step along a direction equiangular to each atom's correlation with the residual. LARS has the same computational complexity as OMP and can deal with continuous sparsity measure.

Equation (2) can be extended to take into account a mask vector m of masked-out entries in observation $x^{(i)}$. This removes the participation of these entries and constrains the residual to only those that are available. In that case, LARS and OMP can be extended to process the mask m, and we refer to these approaches as mLARS and mOMP. Previous work suggested that explicitly modeling the mask-noise may improve the sparseness of the weight solution [21]. However, as we show in our experiments, we do not observe any improvements over the standard ones. This supports the previous observation that DLSR is already well-adapted to deal with mask noise [3].

Other alternatives exist to LARS and OMP. As part of our contributions, we also use other feature coding approaches. One is called Soft-Thresholding (ST) [7]. ST gives as weight solution $w^{(i)}$ a vector containing correlation magnitudes between observation $x^{(i)}$ and each atom $D^{(j)}$. It is formulated as follows:

$$w_j^{(i)} = \text{sign}(D^{(j)} \cdot x^{(i)}) \cdot \max\left\{0, |D^{(j)} \cdot x^{(i)}| - \tau\right\} \quad (3)$$

where τ is the shrinkage hyper-parameter that thresholds insignificant correlations with low magnitudes. One advantage of ST over LARS and OMP is that it is not an optimization method. Its computational complexity is solely dependent on the matrix-vector multiplication, as seen in (3). This can be particularly advantageous when speed is important.

We also define Natural (N) feature coding approaches. These are defined as follows. For ODL, we use as feature coding whichever approach was used for weight optimization. For instance, the natural feature coding of ODL-1 is LARS with the same sparsity parameter λ that was used during ODL-1. Similarly, the natural feature coding of ODL-0 is OMP, this time with the same γ used during ODL-0. For GSVQ, we used OMP with a fixed $\gamma = 1$. For R and RP, we used ST with $\tau = 0$, which corresponds to a random linear projection.

Finally, we chose as natural feature coding for NKM the standard KMeans Triangular (KMT) [5]. KMT is a soft alternative to the usual *one-of-K* encoding that associates each observation to the closest centroid. It can be formulated as follows:

Figure 2. Left: a pair of scissors with a candidate grasp rectangle image. Center-Left: image taken from the rectangle rotated to match the global image orientation. Center-Right: rescaled image with preserved aspect ratio. The black regions indicate masked-out padding. Right: rescaled image without preserved aspect ratio. Preserving aspect ratio reduces distortion.

$$w_j^{(i)} = \max\{0, \mu(z^{(i)}) - z_j^{(i)}\}, \quad (4)$$

where $z_j^{(i)} = \|x^{(i)} - D^{(j)}\|_2$ is the euclidean distance between observation $x^{(i)}$ and centroid $D^{(j)}$, and $\mu(z^{(i)})$ is the mean of the elements of $z^{(i)}$. This feature coding technique gives sparse weight vector $w^{(i)}$ and is not computationally expensive.

After computing the weight solution $w^{(i)}$, we apply Polarity Splitting (PS) [4]. PS constructs a vector $f^{(i)}$ from $w^{(i)}$ by splitting the positive weights from the negative ones. This can be formulated as follows:

$$f_j^{(i)} = \max\{w_j^{(i)}, 0\} \qquad f_{j+d}^{(i)} = \max\{-w_j^{(i)}, 0\}$$

This technique improves the representative ability of linear classifiers as it forces them to model positive and negative weights differently.

Starting from this point, we are now referencing to feature vector $f^{(i)}$ when talking about the output vector of any feature coding approach.

4. DLSR-Based Grasp Localization

We have seen in previous section 3 how to learn a dictionary D and extract features $f^{(i)}$ from a set of observations $x^{(i)}$. In this section, we elaborate on the approach that we used to actually perform grasp localization.

4.1. Grasp Representation

A fundamental concept in grasping is the grasp representation. It has undergone significant evolution over the past decades. The 5-dimensional *grasp rectangle* representation is one of the most popular method for encapsulating the 7-dimensional two-plates parallel gripper configuration [15]. The 2D oriented rectangle, shown in Fig. 1, indicates the gripper's location, orientation and physical limitations:

$$R = \{x, y, \theta, w, h\}.$$

Using the grasp rectangle representation makes grasp recognition analogous to object recognition, and grasp detection

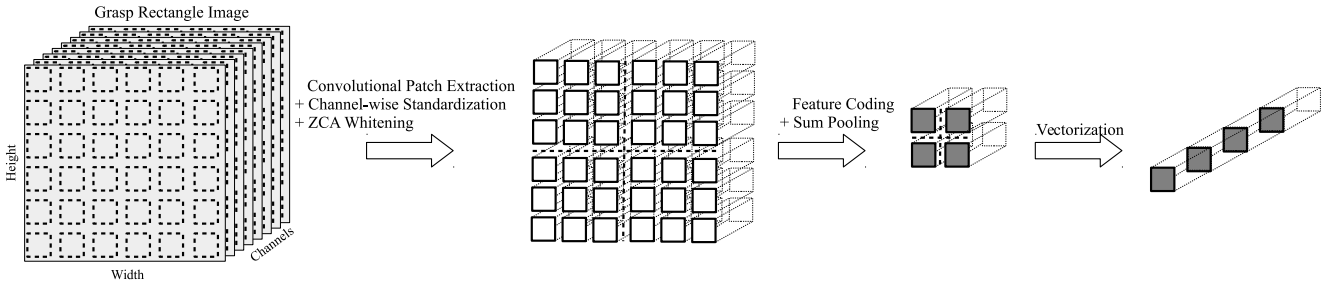

Figure 3. Feature extraction process. Left: all $6 \times 6 \times 8$ patches $x^{(i)}$ are extracted in a convolutional way from the image. Center-Left: each patch $x^{(i)}$ is mapped to its feature representation $f^{(i)}$ given a dictionary D and a feature coding method. Center-Right: a four quadrants sum pooling is applied on feature vectors $f^{(i)}$. Right: all quadrant pooled weights are concatenated into a single vector. The final vector on the right is used as input to the SVM.

analogous to object detection. For grasp recognition, the goal is to determine whether a grasp rectangle R is a good or bad candidate. For grasp detection, the goal is to predict the configuration of the best rectangle R^*.

4.2. Feature Extraction Process

The grasp rectangle introduces another key notion that our proposed approach relies one, which is the *grasp rectangle image*. As shown in Fig. 2, it is the underlying image captured by the rectangle. Extracting relevant features from the grasp rectangle image is essential for grasp localization. We now elaborate on the feature extraction process that we chose for this task. The overall process is shown in Fig. 3.

From the RGBD image, we compute the gray channel (K) and the depth normal coordinates N_x, N_y and N_z. Each image now contains eight channels. We rotate the grasp rectangle image to match the global orientation and rescale it to $24 \times 24 \times 8$ with aspect-ratio preserved. As shown in Fig. 2, preserving aspect ratio is important to avoid distortion.

We then extract, in a convolutional way, all $6 \times 6 \times 8$ sub-images from the grasp rectangle image. The vectorized version of each sub-image is a 288-dimensional vector that is used as observation $x^{(i)}$. We perform channel-wise standardization and ZCA whitening, then extract feature vectors $f^{(i)}$ using a learned dictionary D and feature coding approach. After, we divide the image into four quadrants and perform sum pooling over the feature vectors $f^{(i)}$ of each quadrant respectively. The pooled feature vectors from all quadrants are concatenated into a single vector that is used as input to the classifier. This feature extraction pipeline is known as *spatial pyramid matching* which has shown promising results in past work [34].

4.3. Grasp Recognition and Detection

The feature vector obtained as output of the grasp rectangle image feature extraction process is then used a input for performing grasp localization. In the case of grasp recognition, we use as classification model a ℓ_2-linear SVM optimized with a standard L-BFGS solver from Schmidt's *minFunc* toolbox [30]. For grasp detection, we opted for a standard sliding window search in grasp rectangle space with a horizontal and vertical stride of 10 pixels. We varied the size of the rectangle from 10 pixels to 90 pixels with a stride of 10 pixels, and varied the orientation from 0 degree to 180 degrees with a stride of 15 degrees. For each of these grasp rectangle images, we performed feature extraction and inputed them to the SVM. The rectangle having the highest score was chosen as the candidate grasp.

Figure 4. The Cornell Grasping Dataset contains 885 RGBD images of 240 distinct objects. Each image has labeled positive and negative grasp rectangles. They are varied in term of shape and positions, but do not capture exhaustively all graspable regions.

5. Experiments

In this section, we perform several experiments of our proposed DLSR framework for grasp localization.

5.1. Cornell Dataset

The Cornell Grasping Dataset contains 885 RGBD images of 240 distinct objects [15]. Each image has multiple positively- and negatively-labeled grasp rectangles, specifically selected for parallel plate grippers. As shown in Fig. 4,

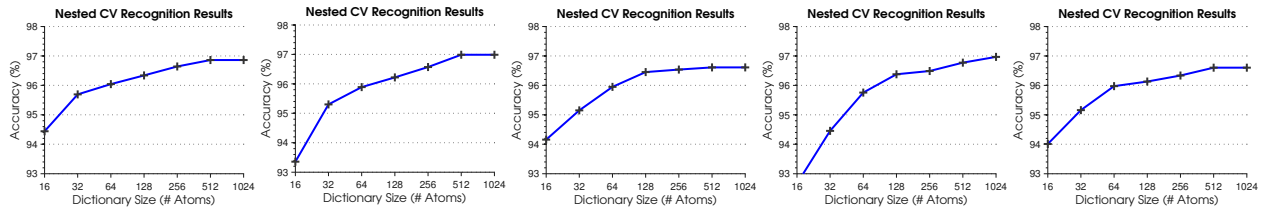

Figure 5. The effects of dictionary size on recognition accuracies. Increasing the dictionary size improves the results up to a limit (plateau) where no more boost is possible.

Table 1. Cross-validation results of all combinations of dictionary learning and feature coding, for Cornell dataset. Numbers are grasp recognition accuracies in percent (%).

Dictionary	Features					
	LARS	mLARS	OMP	mOMP	ST	N
ODL-0	96.68	96.71	96.60	96.69	96.50	96.58
ODL-1	96.63	96.74	96.61	96.61	96.73	96.65
GSVQ	96.72	96.50	96.66	96.70	96.50	95.65
NKM	**96.86**	96.64	96.58	96.64	96.42	96.52
RP	96.43	96.51	96.35	96.20	96.28	96.37
R	95.84	95.52	95.34	95.15	95.78	95.90

the labeled rectangles are varied in terms of size, orientation and position. This dataset is difficult because it is small and some images have unlabeled graspable regions.

5.2. Grasp Recognition Experiments

Previous works on the Cornell dataset reported their results using 5-fold cross-validation [15, 19]. They optimized for the hyper-parameters using a separate set of grasp examples (which we call the validation set). However, exactly comparing our results to theirs is impossible because they did not report which examples they selected for validation. We instead performed 5-5 folds nested cross-validation. This removes the need to specify the validation set, reduces the bias induced by choosing which examples to put in it, and makes our results still comparable to the previous ones.

The nested cross-validation accuracies (in %) for grasp recognition using a dictionary of $d = 300$ atoms are reported in Table 1. The best dictionary learning + feature coding combination is NKM-LARS with an accuracy of 96.86%. The worst combination is R-mOMP with an accuracy of 95.15%. As a comparison, previous approaches achieved 89.6% with a hand-designed score function [15], and 93.7% with a cascade of multi-layer perceptrons [19]. These recognition results suggest that DLSR is a viable approach for grasp recognition.

Interestingly, the accuracies vary by no more than 1%. Although it always performed worst, DL method R has competitive performances. Sparse dictionary learning and random dictionaries appears to be compatible in a similar manner than neural networks and random weights [28]. This suggests that other characteristics should be taken into consideration for deciding which combination is more desirable. One is to select the approach with the least amount of hyper-parameters. For dictionary learning, GSVQ, NKM and RP could be considered before ODL-0 and ODL-1 because they are hyper-parameter free. Similarly for feature coding, the natural features of NKM, GSVQ and RP may be taken in consideration before the others because they are also hyper-parameter free.

Another characteristic is computational complexity. For instance, LARS and mLARS were the most time consuming of all, due to the tedious optimization needed for weight optimization. Even though the ℓ_1 sparsity measure appears to extract the most robust features, these encoders would be too cumbersome to use in real-time scenarios. On the contrary, ST and the natural feature coding of NKM and RP are fast and give good performances. They are not optimization-based approaches and their computational complexities only depend on straightforward matrix-vector multiplications.

Explicitly modeling the mask noise in the observations may not be as important as it seems. As we see in Table 1, mLARS and mOMP are not significantly better alternatives then their standard version LARS and OMP. For instance, modeling the mask noise with LARS improved the performance of ODL-0 from 96.68% to 96.71%, but reduced the accuracy of GSVQ from 96.72% to 96.50%. Similarly, modeling the mask noise for OMP did not change the performance of ODL-1, as both alternatives got 96.61%. These results support the previous observation that standard feature coding approaches are already well-adapted to deal with the mask noise [3].

Given these considerations, we selected fives combinations and use them for the next experiments. We chose NKM-N, RP-N, GSVQ-ST, NKM-LARS and ODL-0-N. NKM-N and RP-N appear to be the most appealing combinations because they are both hyper-parameter free and have fast feature coding and dictionary learning algorithms. We also included GSVQ-ST, NKM-LARS and OMP-N

even though they need hyper-parameter tuning. NKM-LARS was the top performer, and GSVQ-ST and OMP-N have fast feature coding methods.

The number of atoms d has a direct influence on the performance of the classifier. Small dictionaries are easy to learn, but may not capture faithfully the latent structure within the data. On the contrary, big dictionaries are more flexible, but can have convergence problem due to *dead atoms* [21]. To evaluate the effects of the dictionary size on the recognition accuracy, we varied the number of atoms d and looked at how performance changed. Figure 5 shows that increasing d improves the results up to a limit where no additional gain is possible. The plateau indicates that dictionary learning is unable to learn new useful features. This is the so-called dead atoms problem, where the learned dictionary D contains atoms that are never activated. It has reached a limit in its representative capability. Figure 5 suggests that using $d = 300$ atoms is a good trade-off between performance and computational complexity. We will therefore use $d = 300$ atoms in the following experiments.

5.3. Grasp Detection Experiments

We also evaluated the performance of the five selected dictionary learning + features coding of section 5.2 on the problem of grasp detection. We performed a standard 5 folds cross-validation using the same hyper-parameters as in grasp recognition. To evaluate the quality of a grasp candidate, we used the *rectangle metric* [15]. Specifically, if the rectangle metric of any of the ground truth rectangles with the candidate is positive, the regression is a success. In more detail, the metric is positive if: *1)* the candidate orientation is within 30° of the ground truth rectangle, and *2)* the Jaccard index between the candidate and the ground truth is greater than 25%, where the Jaccard index between two rectangles R_1 and R_2 is defined as:

$$J\left(R_1, R_2\right) = \frac{\text{area}(R_1 \cap R_2)}{\text{area}(R_1 \cup R_2)}\,. \tag{5}$$

For grasp detection, we have two learning scenarios. One is *image-wise splitting* where we split the images randomly, and the other is *object-wise splitting* where we split the objects randomly, gathering all the image of the object in the same fold. Image-wise splitting studies the ability to generalize to new positions and orientations of an object that has already been seen. This scenario is representative of a typical industrial context, because the set of objects is known beforehand. Object-wise splitting examine the capability to generalize to novel, unseen objects. This scenario is more realistic in the sense of general purpose robotics because training on all possible objects is usually impossible. Using both image-wise and object-wise splitting, we can more accurately ascertain the performance of the tested approaches.

Table 2. Cross validation detection results for the Cornell dataset. Numbers are grasp detection accuracies in percent (%).

Algorithm	Detection Accuracy (%)	
	Image-wise	Object-wise
Jiang *et al*. [15]	60.50	58.30
Lenz *et al*. [19]	73.90	75.60
Redmon *et al*. [27]	88.00	87.10
NKM-LARS	88.67	88.07
GSVQ-ST	88.72	**88.79**
OMP-N	89.34	88.56
NKM-N	**89.40**	88.17
RP-N	87.70	86.61

The cross-validation accuracies (in %) for grasp detection of the five selected approaches using a dictionary of $d = 300$ atoms are reported in Table 2. For image-wise and object-wise split respectively, the best accuracies are obtained by NKM-N with 89.40% and GSVQ-ST with 88.79%. As comparison, a cascade of multilayer perceptrons obtained 73.90% and 75.60% [19] while a convolutional neural network (CNN) achieved 88.00% and 87.10% [27].

Even though CNN is a powerful approach and is currently state-of-the-art in several vision-related problems, DLSR has one advantage over CNN. Due to the relative small amount of data in Cornell dataset, Redmon *et al*. pre-trained their CNN on ImageNet containing RGB images, replaced the blue with the depth channel to make it compatible with RGBD images (giving RGD images), and performed a final fine-tuning on Cornell images. Since blue channel-related low level features are unlikely extracting useful information from depth, such a pre-training approach is clearly sub-optimal. Directly training the CNN on Cornell dataset would require gathering a large quantity of additional images, at a substantial cost. In contrast, DLSR can be directly trained on Cornell RGBD images despite its small size. In an industrial context where datasets are small and new objects are regularly added, DLSR can be a viable approach in the early development stage. This can be seen as an advantage over CNN.

5.4. Dictionary Visualization

To provide an intuition on the representative ability of DLSR, we trained a dictionary with NKM and visualized the learned atoms. In Fig. 6, we show the dictionary trained on the Cornell dataset by separating the modalities into four parts: gray (K), color (RGB), depth (D) and depth normals ($N_xN_yN_z$). Each small square is a de-vectorized atom $D^{(j)}$ of size $6 \times 6 \times M$, when M is the number of modalities. Most squares show localized and oriented Gabor-like filters,

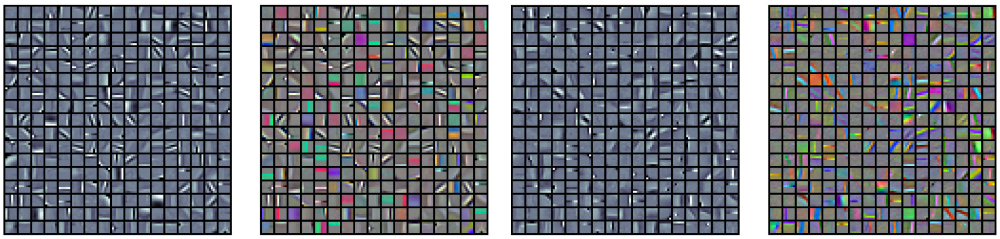

Figure 6. A dictionary D of 300 atoms (each square is an atom $D^{(j)}$) learned using the Cornell dataset shown in four distinct parts: K (gray), RGB, D (depth) and $N_x N_y N_z$ (depth normals). Most atoms show localized and oriented Gabor-like filters.

which has been reported several decades ago as being the building blocks of real world images [22]. The fact that DLSR can learn Gabor-like filters is a strong indication that it can extract proper features from the data.

6. Discussion

To obtain Table 2 high accuracy results in detection, we had to pay a computational price. Even though feature coding approaches have reasonable low computational complexity (LARS is $O(d^3 + nd^2)$, OMP is $O(\gamma nd)$ and ST is $O(nd)$), the exhaustive grid search in grasp rectangle space is computationally demanding. This translates into several minutes to complete a detection which is higher than Redmon *et al.*'s CNN with 76 ms per image. However, since we used a standard CPU and they used a high-end GPU, a DLSR GPU implementation would make a fairer time computation comparison. While implementing ST would be simple, this is not straightforward for LARS and OMP due to their recursive nature. A possible avenue for a useful GPU implementation may be to parallelize grid search by exploiting grasp rectangle candidate independence, i.e. by extracting and scoring all candidates in parallel. However, grid search in grasp rectangle space being more cumbersome than spatial convolutions, the computational time of such a parallelization would still be higher than CNN.

While DLSR is fairly slow during detection, the training phase is significantly faster than CNN. Training a CNN take several days with parallel high-end GPUs, and fine-tuning on Cornell dataset takes several hours [27]. In comparison, it took approximatively ten minutes to train our 300 atoms dictionaries on a standard CPU. While fast training phase is irrelevant in real-time test scenarios, it could be useful to train a CNN directly on RGBD images. Pre-training on ImageNet could be avoided by greedily stacking dictionaries learned on RGBD images, as previously proposed with auto-encoders in the context of RGB images [2]. Such a DLSR and CNN combination would bring the best of both approaches, in which DLSR would improve training while CNN would provide fast detection. We intend to investigate this avenue in future works.

Even though it achieved the lowest detection accuracies (apart from random dictionaries), the RP-N combination is appealing because training the dictionary is instantaneous, feature coding requires only a matrix multiplication, and there is no hyper-parameters. Due to its simplicity, integrating the approach to a grasp localization system in its early deployment phase is straightforward and can give a good glimpse of the overall system performance in later deployment phases. One interesting avenue for future work is to understand the reason why input decorrelation allows randomly sampled patches to make such good dictionaries. For instance, is linear independence sufficient, or better dictionaries could be recovered with non-linear independence? These are several avenues worth investigating.

7. Conclusion

A perception system that determines good grasping positions from Microsoft Kinect RGBD images is a key element toward automating the grasping of ordinary objects. Over the past decades, Dictionary Learning and Sparse Representation (DLSR) has attracted a lot of attention in the vision, signal and image processing communities, but has not been evaluated on the problem of grasp localization. In this paper, we proposed a DLSR framework for extracting relevant features from RGBD images. We showed that it has a fast training phase, do not require a lot of data, and is able to process observations corrupted by a mask-based noise without relying on sophisticated regularization terms. We demonstrated its applicability for recognizing and detecting grasp rectangles on the standard Cornell dataset. Our results suggest that the proposed DLSR framework is suitable for grasp localization. We also performed a comparative study of various dictionary learning and feature coding approach combinations. We analyzed the pros and cons of each alternatives by comparing recognition accuracies, computational complexity and the ability to be parallelized. Our results have shown that most approaches obtained similar recognition performances, but some are more appealing due to their fast speed and ease of use. Future work combining DLSR and deep learning (such as R-CNN) could be an interesting avenue.

References

[1] M. Aharon, M. Elad, and A. Bruckstein. KSVD: An algorithm for designing overcomplete dictionaries for sparse sepresentation. *Signal Processing, Transactions on*, 54(11):4311–4322, 2006.

[2] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, et al. Greedy layer-wise training of deep networks. *NIPS*, 19:153, 2007.

[3] L. Bo, X. Ren, and D. Fox. Learning hierarchical sparse features for rgb-(d) object recognition. *IJRR*, 33(4):581–599, 2014.

[4] A. Coates and A. Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *ICML*, pages 921–928, 2011.

[5] A. Coates, A. Y. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, pages 215–223, 2011.

[6] R. Detry, C. H. Ek, M. Madry, and D. Kragic. Learning a dictionary of prototypical grasp-predicting parts from grasping experience. In *ICRA*, pages 601–608, 2013.

[7] D. L. Donoho and I. M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the american statistical association*, 90(432):1200–1224, 1995.

[8] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

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

[10] C. Eppner, S. Höfer, R. Jonschkowski, R. Martın-Martın, A. Sieverling, V. Wall, and O. Brock. Lessons from the amazon picking challenge: Four aspects of building robotic systems. In *Proceedings of Robotics: Science and Systems*, 2016.

[11] A. Gersho and R. M. Gray. *Vector quantization and signal compression*, volume 159. Springer Science & Business Media, 2012.

[12] C. Goldfeder, P. K. Allen, C. Lackner, and R. Pelossof. Grasp planning via decomposition trees. In *ICRA*, pages 4679–4684, 2007.

[13] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen. The columbia grasp database. In *ICRA*, pages 1710–1716. IEEE, 2009.

[14] D. Guo, T. Kong, F. Sun, and H. Liu. Object discovery and grasp detection with a shared convolutional neural network. In *ICRA*, pages 2038–2043. IEEE, 2016.

[15] Y. Jiang, S. Moseson, and A. Saxena. Efficient grasping from RGBD images: Learning using a new rectangle representation. In *ICRA*, pages 3304–3311, 2011.

[16] D. Kappler, J. Bohg, and S. Schaal. Leveraging big data for grasp planning. In *ICRA*, pages 4304–4311. IEEE, 2015.

[17] K. Lai, L. Bo, and D. Fox. Unsupervised feature learning for 3d scene labeling. In *ICRA*, pages 3050–3057. IEEE, 2014.

[18] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, volume 2, pages 2169–2178, 2006.

[19] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *IJRR*, 34(4-5):705–724, 2015.

[20] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen. Learning hand-eye coordination for robotic grasping with largescale data collection. In *International Symposium on Experimental Robotics*, 2016.

[21] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *ICML*, pages 689–696, 2009.

[22] S. Marčelja. Mathematical description of the responses of simple cortical cells. *JOSA*, 70(11):1297–1300, 1980.

[23] A. T. Miller and P. K. Allen. Graspit!: A versatile simulator for robotic grasping. *Robotics & Automation Magazine*, 11(4):110–122, 2004.

[24] Y. C. Pati, R. Rezaiifar, and P. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Signals, Systems and Computers. Twenty-Seventh Asilomar Conference on*, pages 40–44, 1993.

[25] R. Pelossof, A. Miller, P. Allen, and T. Jebara. An SVM learning approach to robotic grasping. In *ICRA*, volume 4, pages 3512–3518, 2004.

[26] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *ICRA*, 2016.

[27] J. Redmon and A. Angelova. Real-time grasp detection using convolutional neural networks. In *ICRA*, pages 1316–1322, 2015.

[28] A. Saxe, P. W. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Y. Ng. On random weights and unsupervised feature learning. In *ICML*, pages 1089–1096, 2011.

[29] A. Saxena, J. Driemeyer, and A. Y. Ng. Robotic grasping of novel objects using vision. *IJRR*, 27(2):157–173, 2008.

[30] M. Schmidt. minfunc: unconstrained differentiable multivariate optimization in matlab, 2005.

[31] A. M. Tillmann. On the computational intractability of exact and approximate dictionary learning. *IEEE Signal Processing Letters*, 22(1):45–49, 2015.

[32] Z. Wang, Z. Li, B. Wang, and H. Liu. Robot grasp detection using multimodal deep convolutional neural networks. *Advances in Mechanical Engineering*, 8(9):1687814016668077, 2016.

[33] R. Xiaofeng and L. Bo. Discriminatively trained sparse code gradients for contour detection. In *NIPS*, pages 584–592, 2012.

[34] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, pages 1794–1801, 2009.

[35] Q. Zhang and B. Li. Discriminative K-SVD for dictionary learning in face recognition. In *CVPR*, pages 2691–2698, 2010.

[36] Z. Zhang. Microsoft kinect sensor and its effect. *IEEE multimedia*, 19(2):4–10, 2012.

[37] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang. A survey of sparse representation: algorithms and applications. *IEEE Access*, 3:490–530, 2015.