Learning Descriptors for Object Recognition and 3D Pose Estimation

Paul Wohlhart and Vincent Lepetit
Institute for Computer Vision and Graphics, Graz University of Technology, Austria
{wohlhart,lepetit}@icg.tugraz.at

Abstract

Detecting poorly textured objects and estimating their 3D pose reliably is still a very challenging problem. We introduce a simple but powerful approach to computing descriptors for object views that efficiently capture both the object identity and 3D pose. By contrast with previous manifold-based approaches, we can rely on the Euclidean distance to evaluate the similarity between descriptors, and therefore use scalable Nearest Neighbor search methods to efficiently handle a large number of objects under a large range of poses. To achieve this, we train a Convolutional Neural Network to compute these descriptors by enforcing simple similarity and dissimilarity constraints between the descriptors. We show that our constraints nicely untangle the images from different objects and different views into clusters that are not only well-separated but also structured as the corresponding sets of poses: The Euclidean distance between descriptors is large when the descriptors are from different objects, and directly related to the distance between the poses when the descriptors are from the same object. These important properties allow us to outperform state-of-the-art object views representations on challenging RGB and RGB-D data.

1. Introduction

Impressive results have been achieved in 3D pose estimation of objects from images during the last decade. However, current approaches cannot scale to large-scale problems because they rely on one classifier per object, or multi-class classifiers such as Random Forests, whose complexity grows with the number of objects. So far the only recognition approaches that have been demonstrated to work on large scale problems are based on Nearest Neighbor (NN) classification [23, 16, 8], because extremely efficient methods for NN search exist with an average complexity of $O(1)$ [24, 21]. Moreover, Nearest Neighbor (NN) classification also offers the possibility to trivially add new objects, or remove old ones, which is not directly possible with neural networks, for example. However, to the best of our knowledge, such an approach has not been applied to the 3D pose estimation problem, while it can potentially scale to many objects seen under large ranges of poses. For example, [8] only focuses on object recognition without considering the 3D pose estimation problem.

For NN approaches to perform well, a compact and discriminative description vector is required. Such representations that can capture the appearance of an object under a certain pose have already been proposed [7, 14], however they have been handcrafted. Our approach is motivated by the success of recent work on feature point descriptor learning [5, 35, 20], which shows that it is possible to learn compact descriptors that significantly outperform handcrafted methods such as SIFT or SURF.

However, the problem we tackle here is more complex: While feature point descriptors are used only to find the points' identities, we here want to find both the object's identity and its pose. We therefore seek to learn a descriptor with the two following properties: a) The Euclidean distance between descriptors from two different objects should be large; b) The Euclidean distance between descriptors from the same object should be representative of the similarity between their poses. This way, given a new object view, we can recognize the object and get an estimate of its pose by matching its descriptor against a database of registered descriptors. New objects can also be added and existing ones removed easily. To the best of our knowledge, our method is the first one that learns to compute descriptors for object views.

Our approach is related to manifold learning, but the key advantage of learning a direct mapping to descriptors is that we can use efficient and scalable Nearest Neighbor search methods. This is not possible for previous methods relying on geodesic distances on manifolds. Moreover, while previous approaches already considered similar properties to a) and b), to the best of our knowledge, they never considered both simultaneously, while it is critical for efficiency. Combining these two constraints in a principled way is far from trivial, but we show it can be done by training a Convolutional Neural Network [18] using simple constraints to compute the descriptors. As shown in Fig. 1, this results in

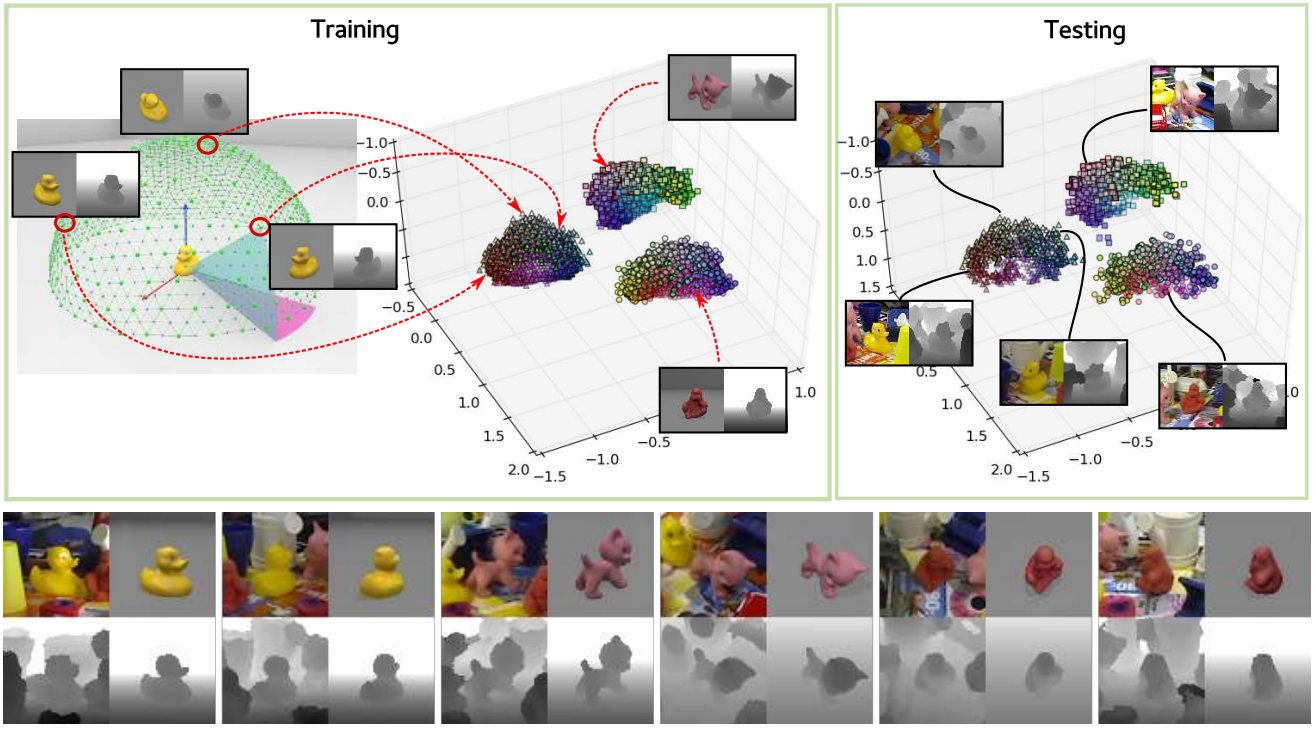


Figure 1. Three-dimensional descriptors for several objects under many different views computed by our method on RGB-D data. **Top-left:** The training views of different objects are mapped to well-separated descriptors, and the views of the same object are mapped to descriptors that capture the geometry of the corresponding poses, even in this low dimensional space. **Top-right:** New images are mapped to locations corresponding to the object and 3D poses, even in the presence of clutter. **Bottom:** Test RGB-D views and the RGB-D data corresponding to the closest template descriptor.

a method that nicely untangles the views of different objects into descriptors that capture the identities and poses of the objects.

We evaluate our approach on instance recognition and pose estimation data with accurate ground truth and show significantly improved results over related methods. Additionally we perform experiments assessing the ability of the method to generalize to unseen objects showing promising results.

2. Related Work

Our work is related to several aspects of Computer Vision, and we focus here on the most relevant and representative work. Our approach is clearly in the framework of 2D view specific templates [15], which is conceptually simple, supported by psychophysical experiments [33], and was successfully applied to various problems and datasets over the last years [22, 19, 13, 11, 8, 28].

However, most of these works rely on handcrafted representations of the templates, for example HOG [7] or LineMOD [14]. In particular, LineMOD was designed explicitly in the context of object detection and pose estimation. However these handcrafted representations are suboptimal compared to statistically learned features. [19, 11, 28] show how to build discriminative models based on these representations using SVM or boosting applied to training data. [19, 28] do not consider the pose estimation problem, while [11] focuses on this problem only, with a discriminatively trained mixture of HOG templates. Exemplars were recently used for 3D object detection and pose estimation in [1], but still rely on a handcrafted representation.

As mentioned in the introduction, our work is influenced by work developed for keypoint descriptor learning. Some of these methods are applied to existing descriptors to make them more discriminative, such as in [10, 31], but others are trained directly on image data. [5] introduces datasets made of "positive pairs" of patches corresponding to the same physical points and "negative pairs" of patches corresponding to different points. It is used for example in [35] to learn a binary descriptor with boosting. [20] uses a "siamese" architecture [6] to train a neural network to compute discriminative descriptors. Our approach is related to this last work, but the notion of pose is absent in their case. We show how to introduce this notion by using triplets of training examples in addition to only pairs.

Instead of relying on rigid templates as we do, many works on category recognition and pose estimation rely on part-based models. [30] pioneered this approach, and

learned canonical parts connected by a graph for object recognition and pose estimation. [26] extends the Deformable Part Model to 3D object detection and pose estimation. [25] uses contours as parts. One major drawback of such approaches is that the complexity is typically linear with the number of objects. It is also not clear how important the "deformable" property really is for the recognition, and rigid templates seem to be sufficient [9].

Our approach is also related to manifold learning [27]. For example, [29] learns an embedding that separates extremely well the classes from the MNIST dataset of digit images, but the notion of pose is absent. [12] learns either for different classes, also on the MNIST dataset, or for varying pose and illumination, but not the two simultaneously. More recently, [2] proposes a method that separates manifolds from different categories while being able to predict the object poses, and also does not require solving an inference problem, which is important for efficiency. However, it relies on a discretisation of the pose space in a few classes, which limits the possible accuracy. It also relies on HOG for the image features, while we learn the relevant image features.

Finally, many works focus as we do on instance recognition and pose estimation, as it has important applications in robotics for example. [14] introduced LineMOD, a fast but handcrafted presentation of template for dealing with poorly textured objects. The very recent [4, 34] do not use templates but rely on recognition of local patches instead. However they were demonstrated on RGB-D images, and local recognition is likely to be much more challenging on poorly textured objects when a depth information is not available. [17] also expects RGB-D images, and uses a tree for object recognition, which however still scales linearly in the numbers of objects, categories, and poses.

3. Method

Given a new input image x of an object, we want to correctly predict the object's class and 3D pose. Because of the benefits discussed above, such as scalability and ability to easily add and remove objects, we formulate the problem as a k-nearest neighbor search in a descriptor space: For each object in the database, descriptors are calculated for a set of template views and stored along with the object's identity and 3D pose of the view. In order to get an estimate for the class and pose of the object depicted in the new input image, we can compute a descriptor for x and search for the most similar descriptors in the database. The output is then the object and pose associated with them.

We therefore introduce a method to efficiently map an input image to a compact and discriminative descriptor that can be used in the nearest neighbor search according to the Euclidean distance. For the mapping, we use a Convolutional Neural Network (CNN) that is applied to the raw image patch as input and delivers the descriptor as activations of the last layer in one forward pass.

We show below how to train such a CNN to enforce the two important properties already discussed in the introduction: a) The Euclidean distance between descriptors from two different objects should be large; b) The Euclidean distance between descriptors from the same object should be representative of the similarity between their poses.

3.1. Training the CNN

In order to train the network we need a set $\mathcal{S}_{\text{train}}$ of training samples, where each sample $s = (x, c, p)$ is made of an image x of an object, which can be a color or grayscale image or a depth map, or a combination of the two; the identity c of the object; and the 3D pose p of the object relative to the camera.

Additionally, we define a set $\mathcal{S}_{\text{db}}$ of templates where each element is defined in the same way as a training sample. Descriptors for these templates are calculated and stored with the classifier for k-nearest neighbor search. The template set can be a subset of the training set, the whole training set or a separate set. Details for the creation of training and template data are given in the implementation section.

3.2. Defining the Cost Function

We argue that a good mapping from the images to the descriptors should be so that the Euclidean distance between two descriptors of the same object and similar poses are small and in every other case (either different objects or different poses) the distance should be large. In particular, each descriptor of a training sample should have a small distance to the one template descriptor from the same class with the most similar pose and a larger distance to all descriptors of templates from other classes, or the same class but less similar pose.

We enforce these requirements by minimizing the following objective function over the parameters w of the CNN:

$$\mathcal{L} = \mathcal{L}_{\text{triplets}} + \mathcal{L}_{\text{pairs}} + \lambda ||w'||_2^2 \ . \tag{1}$$

The last term is a regularization term over the parameters of the network: w' denotes the vector made of all the weights of the convolutional filters and all nodes of the fully connect layers, except the bias terms. We describe the first two terms $\mathcal{L}_{\text{triplets}}$ and $\mathcal{L}_{\text{pairs}}$ below.

3.2.1 Triplet-wise terms

We first define a set $\mathcal{T}$ of triplets (s_i, s_j, s_k) of training samples. Each triplet in $\mathcal{T}$ is selected such that one of the two following conditions is fulfilled:

- either s_i and s_j are from the same object and s_k from another object, or

- the three samples s_i, s_j, and s_k are from the same object, but the poses p_i and p_j are more similar than the poses p_i and p_k.

These triplets can therefore be seen as made of a pair of similar samples (s_i and s_j) and a pair of dissimilar ones (s_i and s_k). We introduce a cost function for such a triplet:

$$c(s_i, s_j, s_k) = \max\left(0, 1 - \frac{||f_w(x_i) - f_w(x_k)||_2}{||f_w(x_i) - f_w(x_j)||_2 + m}\right) \quad , \tag{2}$$

where $f_w(x)$ is the output of the CNN for an input image x and thus our descriptor for x, and m is a margin. We can now define the term $\mathcal{L}_{\text{triplets}}$ as the sum of this cost function over all the triplets in $\mathcal{T}$:

$$\mathcal{L}_{\text{triplets}} = \sum_{(s_i,s_j,s_k)\in\mathcal{T}} c(s_i, s_j, s_k) \quad . \tag{3}$$

It is easy to check that minimizing $\mathcal{L}_{\text{triplets}}$ enforces our two desired properties in one common framework.

The margin m serves two purposes. First, it introduces a margin for the classification. It also defines a minimum ratio for the Euclidean distances of the dissimilar pair of samples and the similar one. This counterbalances the weight regularization term, which naturally contracts the output of the network and thus the descriptor space. We set m to 0.01 in all our experiments.

The concept of forming triplets from similar and dissimilar pairs is adopted from the field of metric learning, in particular, the method of [37], where it is used to learn a Mahalanobis distance metric. Note also that our definition of the cost is slightly different from the one in [36], which uses $c(s_i, s_j, s_k) = \max\left(0, m + ||f_w(x_i) - f_w(x_j)||_2^2 - ||f_w(x_i) - f_w(x_k)||_2^2\right)$, where m is set to 1. Our formulation does not suffer from a vanishing gradient when the distance of the dissimilar pair is very small (see suppl. material). Also the increase of the cost with the distance of the similar pair is bounded, thus putting more focus on local interactions. In practice, however, with proper initialization and selection of m both formulations deliver similar results.

3.2.2 Pair-wise terms

In addition to the triplet-wise terms, we also use pair-wise terms. These terms make the descriptor robust to noise and other distracting artifacts such as changing illumination. We consider the set $\mathcal{P}$ of pairs (s_i, s_j) of samples from the same object under very similar poses, ideally the same, and we define the $\mathcal{L}_{\text{pairs}}$ term as the sum of the squared Euclidean distances between the descriptors for these samples:

$$\mathcal{L}_{\text{pairs}} = \sum_{(s_i,s_j)\in\mathcal{P}} ||f_w(x_i) - f_w(x_j)||_2^2 \quad . \tag{4}$$

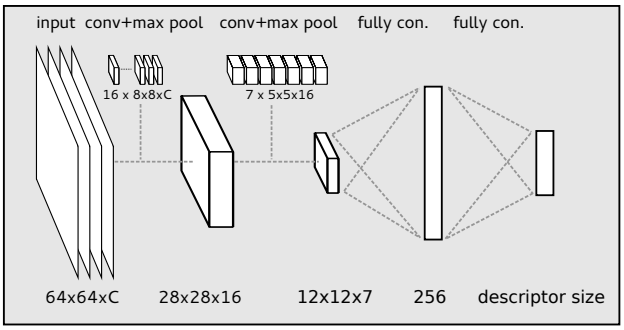


Figure 2. Network structure: We use a CNN made of two convolutional layers with subsequent 2×2 max pooling layers, and two fully connected layers. The activations of the last layer form the descriptor for the input image.

This term therefore enforces the fact that for two images of the same object and same pose, we want to obtain two descriptors which are as close as possible to each other, even if they are from different imaging conditions: Ideally we want the same descriptors even if the two images have different backgrounds or different illuminations, for example. As will be discussed in more detail in Section 4.1, this also allows us to use a mixture of real and synthetic images for training.

Note that we do not consider dissimilar pairs unlike work in keypoint descriptors learning for example. With dissimilar pairs the problem arises how strong to penalize a certain distance between the two samples, given their individual labels. Using triplets instead gives the possibility to only consider relative dissimilarity.

3.3. Implementation Aspects

The exact structure of the network we train to compute the descriptors is shown in Figure 2. It consists of two layers that perform convolution of the input with a set of filters, max-pooling and sub-sampling over a 2×2 area and a rectified linear (ReLU) activation function, followed by two fully connected layers. The first fully connected layer also employs a ReLU activation, the last layer has linear output and delivers the final descriptor.

We optimize the parameters w of the CNN by Stochastic Gradient Descent on mini-batches with Nesterov momentum [32]. Our implementation is based on Theano [3].

The implementation of the optimization needs some special care: Since we are working with mini-batches, the data corresponding to each pair or triplet has to be organized such as to reside within one mini-batch. The most straightforward implementation would be to place the data for each pair and triplet after each other, calculate the resulting gradient wrt. the network's parameters individually and sum them up over the mini-batch. However, this would be inefficient since descriptors for templates would be calculated multiple times if they appear in more than one pair or triplet.

To assemble a mini-batch we start by randomly taking one training sample from each object. Additionally, for each of them we add its template with the most similar pose, unless it was already included in this mini-batch. This is iterated until the mini-batch is full. However, this procedure can lead to very unequal numbers of templates per object if, for instance, all of the selected training samples have the same most similar template. We make sure that for each object at least two templates are included by adding a random one if necessary. Pairs are then formed by associating each training sample with its closest template. Additionally, for each training sample in the mini-batch we initially create three triplets. In each of them the similar template is set to be the one with the closest pose and the dissimilar sample is either another, less similar template from the same object or any template of a different object.

During the optimization, after the first set of epochs, we perform boot-strapping of triplets within each mini-batch to focus on the difficult samples: For each training sample we add two additional triplets. The similar template is again the closest one. The dissimilar ones are those templates that currently have the closest descriptors, one from the same object but different pose and one from all the other objects.

Another aspect to take care of is the fact that the objective function must be differentiable with respect to the parameters of the CNN, while the derivative of the square root—used in the triplet-wise cost—is not defined for a distance of 0. Our solution is to add a small constant ϵ before taking the square root. Another possible approach [36] is to take the square of the norm. However, this induces the problem (mentioned in Section 3.2.1) that for very small distances of the dissimilar pair, the gradient becomes very small and vanishes for zero distance.

4. Evaluation

We compare our approach to LineMOD and HOG on the LineMOD dataset [13]. This dataset contains training and test data for object recognition and pose estimation of 15 objects, with accurate ground truth. It comes with a 3D mesh for each of the objects. Additionally, it also provides sequences of RGB images and depth maps recorded with a Kinect sensor.

4.1. Dataset Compilation

We train a CNN using our method on a mixture of synthetic and real world data. As in [14], we create synthetic training data by rendering the mesh available for each of the objects in the dataset from positions on a half-dome over the object, as shown in Fig. 1 on the left. The viewpoints are defined by starting with a regular icosahedron and recursively subdividing each triangle into 4 sub-triangles. For the template positions the subdivison is applied two times. After removing the lower half-sphere we end up with 301 evenly distributed template positions. Additional training data is created by subdividing one more time, resulting in 1241 positions.

From each pose we render the object standing on a plane over an empty background using Blender[1]. We parameterize the object pose with the azimuth and elevation of the camera relative to the object. We store the RGB image as well as the depth map.

For the real world data we split the provided sequences captured with the Kinect randomly into a training and a test set. We ensure an even distribution of the samples over the viewing hemisphere by taking two real world images close to each template, which results roughly in a 50/50 split of the data into training and test. Preliminary experiments showed very little to no variance over the different train/test splits and, thus, all results presented here report runs on one random split, fixed for each experiment.

The whole training data set is augmented by making multiple copies with added noise. On both RGB and depth channel we add a small amount of Gaussian noise. Additionally, for the synthetic images, we add larger fractal noise on the background, to simulate diverse backgrounds.

Note that the template views, which are ultimately used in the classification are purely synthetic and noise-free renderings on clean backgrounds. The algorithm, thus, has to learn to map the noisy and real world input data to the same location in descriptor space as the clean templates.

As pointed out in [14] some of the objects are rotationally invariant, to different degrees. Thus, the measure of similarity of poses used for the evaluation and, in our case to define pairs and triplets, should not consider the azimuth of the viewing angle for those objects. We treat the *bowl* object as fully rotationally invariant. The classes *eggbox*, *glue* are treated as symmetric, meaning a rotation by 180° around the z-axis shows the same pose again. The *cup* is a special case because it looks the same from a small range of poses, but from sufficient elevation such that the handle is visible, the exact pose could be estimated. We also treat it as rotationally invariant, mainly to keep the comparison to LineMOD fair.

We extract a patch centered at the object and capturing a fixed size window in 3D at the distance of the object's center. In order to also address the detection part in a sliding window manner, it would be necessary to extract and test several scales. However, only a small range of scales needs to be considered, starting with a maximal one, defined by the depth at the center point, and going down until the center of the object is reached.

Before applying the CNN we normalize the input images. RGB images are normalized to the usual zero mean, unit variance. For depth maps we subtract the depth at the center of the object, scale down such that 20cm in front and

[1] http://www.blender.org

behind the object's center are mapped to the range of $[-1, 1]$ and clip everything beyond that range.

The test sequences captured with the Kinect are very noisy. In particular, there are many regions with undefined depth, introducing very large jumps for which the convolutional filters with ReLU activation functions might output overly strong output values. Therefore, we pre-process the test data by iteratively applying median filters in a 3×3 neighborhood, but only on the pixels for which the depth is available, until all gaps are closed.

4.2. Network Optimization

For the optimization we use the following protocol: We initially train the network on the initial dataset for 400 epochs, an initial learning rate of 0.01 and a momentum of 0.9. Every 100 epochs the learning rate is multiplied by 0.9. Then we perform two rounds of bootstrapping triplet indices as explained in Section 3.3, and for each round we train the CNN for another 200 epochs on the augmented training set. In the end we train another 300 epochs with the learning rate divided by 10 for final fine-tuning. The regularization weight λ is set to 10^{-6} in all our experiments.

4.3. LineMOD and HOG

We compare our learned descriptors to the LineMOD descriptor and HOG as a baseline, as it is widely used as representation in the related work. For LineMOD we use the publicly available source code in OpenCV. We run it on the same data as our method, except for the median filter depth inpainting and normalization: LineMOD handles the missing values internally and performed better without these pre-processing operations.

For HOG we also use the publicly available implementation in OpenCV. We extract the HOG descriptors from the same data we use with our CNN. We use a standard setup of a 64×64 window size, 8×8 cells, 2×2 cells in a block and a block stride of 8, giving a 1764-dimensional descriptor per channel. We compute descriptors on each RGB and depth channel individually and stack them. For evaluation we normalize all descriptors to length 1 and take the dot product between test and template descriptors as similarity measure.

4.4. Manifolds

Figure 1 plots the views of three objects after being mapped into 3-dimensional descriptors, for visualization purposes. As can be seen, not only the descriptors from the different objects are very well separated, but they also capture the geometry of the corresponding poses. This means that the distances between descriptors is representative of the distances between the corresponding poses, as desired.

For longer descriptors we show an evaluation of the relation between distances of descriptors and similarity between poses in Figure 3. For each object, we computed the distances between every sample in the test set and every template for the same object in the training set, as well as the angles between their poses. We then plot a two-dimensional histogram over these angle/distance pairs. Correlation between small angles and large distances indicates the risk of missed target templates, and correlation between large angles and small distances the risk of incorrect matches. Ideally the histograms should therefore have large values only on the diagonal.

The histograms for the descriptors computed with our method clearly show that the distance between descriptors increase with the angle between the views, as desired, while the histograms for LineMOD and HOG show that these descriptors are much more ambiguous.

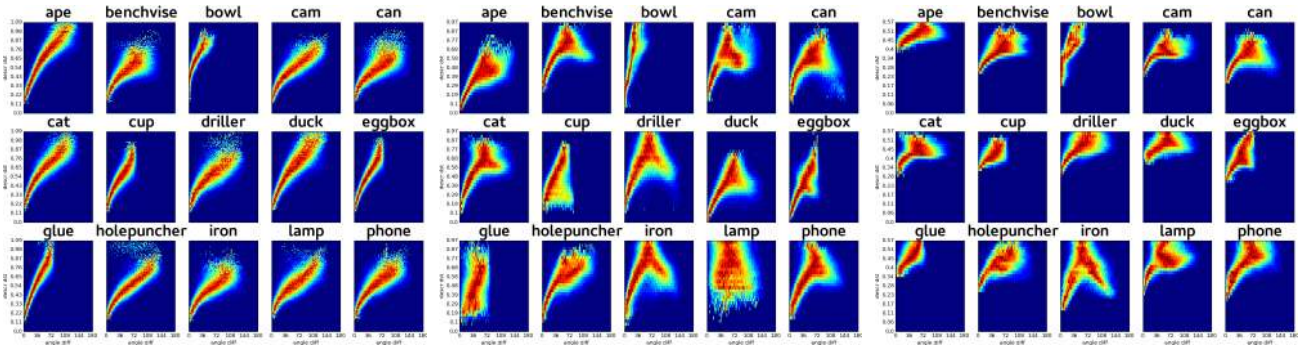
Additionally, the ability of the descriptors to separate the different classes is evaluated in Figure 4. For every test sample descriptor we compute the distance to the closest template descriptor of the same object and the closest from any other object and plot a histogram over those ratios. Clearly, descriptors obtained with our method exhibit a larger ratio for most samples and thus separate the objects better.

4.5. Retrieval Performance

What we ultimately want from the descriptors is that nearest neighbors are from the same class and have similar poses. In order to evaluate the performance we thus perform the following comparisons.

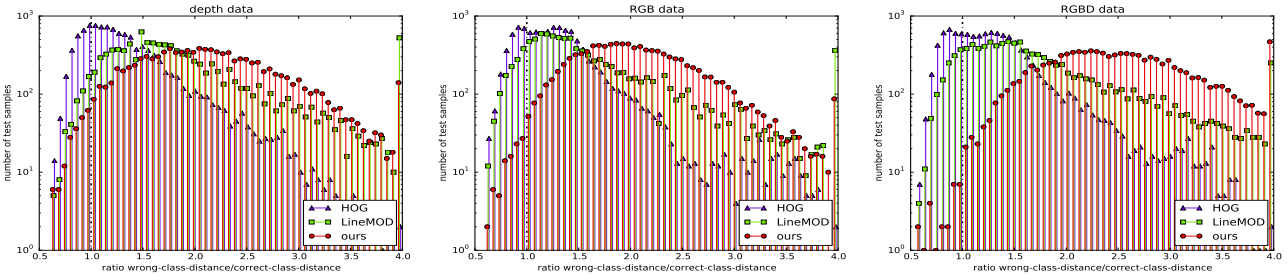
The scores reported for LineMOD in [14] represent the accuracy of the output of the whole processing pipeline, including the descriptor calculation, retrieval of similar templates, pruning the set with heuristics and refinement of the pose for a set of candidate matches by aligning a voxel model of the object. The contribution of this work is to replace the descriptors for the retrieval of templates with similar pose. Thus, we evaluate and compare this step in separation of the rest of the pipeline.

Evaluation Metric For each test sample we consider the k-nearest neighbors according to the descriptors and similarity metric of each method, the Euclidean distance in our case, the dot product for HOG, and the matching score of LineMOD. Among those k nearest templates we search for the one with the best closest pose to the test sample's pose, assuming that this one would perform best in the subsequent refinement process and thus finally be selected. The pose error is measured by the angle between the two positions on the viewing half-sphere. We define the accuracy as the percentage of test images for which the best angle error is below a certain threshold. The minimum angle error for which perfect accuracy can theoretically be reached is 5°, because that is the maximal distance of a test image to its closest template.



(a) ours (b) LineMOD (c) HOG

Figure 3. Histograms of the correlations between Pose Similarity (x-axis) and Descriptor Distance (y-axis) for each of the 15 objects of the LineMOD dataset on RGB-D data, as described in Section 4.4. Distances in the descriptor space are much more representative of the similarity between poses with our method than with LineMOD or HOG.



(a) depth (b) RGB (c) RGB-D

Figure 4. Evaluation of the class separation, over the 15 objects of the LineMOD dataset. The histograms plot the ratios between the distance to the closest template from the correct object and the distance to the closest template from any other object, for depth, RGB, and RGB-D data. Ratios above 4 are clipped; the y-axis is scaled logarithmically. Our method has considerably fewer samples for which the ratio is below one, which indicates less confusion between the objects.

Descriptor Length In Figure 5 we evaluate the influence of the length of the descriptors learned on depth data. As can be seen the maximal performance is already reached with a 16 dimensional descriptor, while the length of the HOG descriptor is 1764. Thus, we use a 16 dimensional descriptor for all of the following experiments, including for the RGB and RGB-D data.

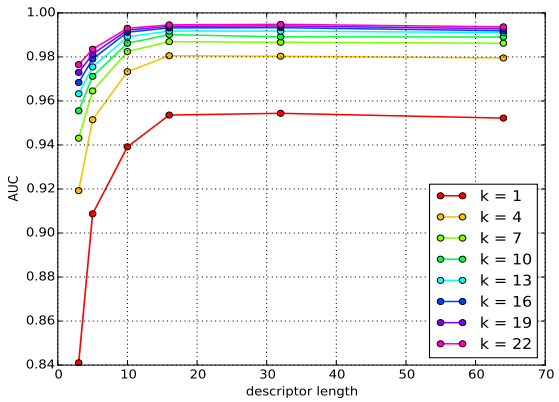
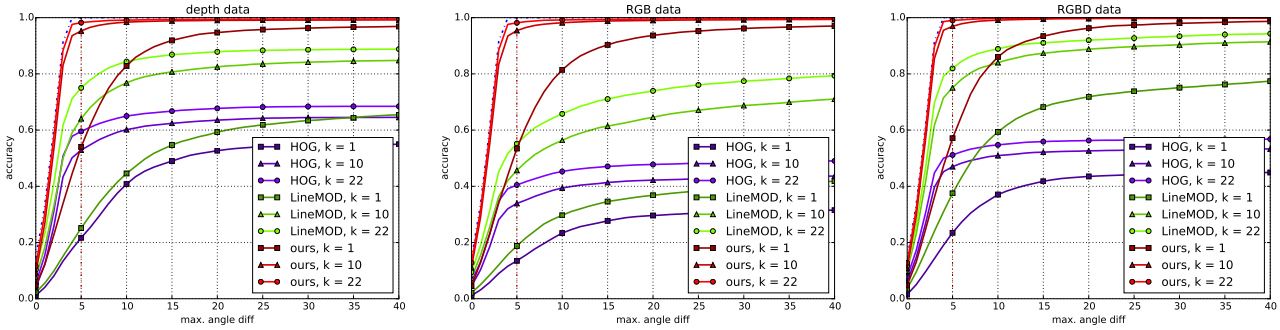


Figure 5. Evaluation of the descriptor length on depth data. Only 16 values are sufficient to reliably represent an object view, after which the performance plateaus.

Results We evaluate all three approaches on depth, RGB, and RGB-D data. Figure 6 and Table 1 summarize the results. For *depth maps*, results are shown in Figure 6 (a). When only considering 1 nearest neighbor we achieve a recognition rate of 98.1%, as opposed to the 69.5% achieved by the LineMOD descriptor and a pose error of less than 20° for 94.7% of the test samples (59.3% for LineMOD). Figure 6 (b) shows results for training and testing on *color images*. While both LineMOD and HOG cannot reach the performance they obtain on the depth data on RGB alone, our descriptor performs almost identically in this setup. Finally, Figure 6 (c) shows results for training and testing on the combination of color images and depth maps. While LineMOD takes advantage of the combination of the two modalities, it is clearly outperformed by our descriptor, as taking the single nearest neighbor exhibits a pose error below 20° for 96.2% of the test samples and an overall recognition rate of 99.8%, an almost perfect score.



(a) depth (b) RGB (c) RGB-D

Figure 6. Performance evaluation and comparison to LineMOD and HOG on all 15 objects of the LineMOD dataset and depth, RGB, and RGB-D data. Each graph plots the accuracy over the y-axis for given a maximal allowed error of the resulting object's pose on the x-axis. Curves for different k are computed by taking k-nearest neighbors and selecting the one with the best matching pose. For our method, the descriptor length was set to 32 for depth, RGB, and RGB-D data. HOG uses 1764 values per channel, and LineMOD uses a length of 63 for depth and RGB data, and of 126 for RGB-D data.

		Depth				RGB				RGB-D			
	k	5°	20°	40°	180°	5°	20°	40°	180°	5°	20°	40°	180°
ours	1	**54.4**	**94.7**	**96.9**	**98.1**	**53.4**	**93.7**	**97.0**	**99.1**	**57.1**	**96.2**	**98.7**	**99.8**
LineMOD	1	25.1	59.3	65.4	69.5	18.8	36.8	41.9	49.6	37.5	71.8	77.4	83.7
HOG	1	21.7	52.7	54.9	55.3	13.5	29.6	31.5	33.6	23.5	43.5	44.9	46.2
ours	22	**98.2**	**99.4**	**99.5**	**99.6**	**98.2**	**99.5**	**99.6**	**99.7**	**99.0**	**99.9**	**99.9**	**99.9**
LineMOD	22	75.0	87.9	88.8	89.5	55.2	74.0	79.3	83.5	81.9	92.0	94.3	96.4
HOG	22	59.5	67.7	68.5	68.9	40.5	47.8	49.0	50.9	51.1	56.3	56.8	57.5

Table 1. Performance comparison to LineMOD and HOG on all 15 objects of the LineMOD dataset and depth, RGB, and RGB-D data, for several tolerated angle errors. Our method systematically outperforms the other representations. The value at 180° indicates the object recognition rate when the pose is ignored.

4.6. Generalization

As a last experiment, we show that our descriptor can generalize to unseen objects. This evaluation was performed using depth only. To do so, we train the CNN on 14 out of the 15 objects. We then perform the evaluation just as above by computing descriptors for the new object. As can be seen from the histogram of Fig. 7-left, our method generalizes well to this unseen object. The overall performance rate is slightly reduced since the network could not learn the subtle differences between the unseen object and the others. Most of the miss-classifications are with the ape, whose shape looks similar to the duck's under some viewpoints, as shown in Fig. 7-right.

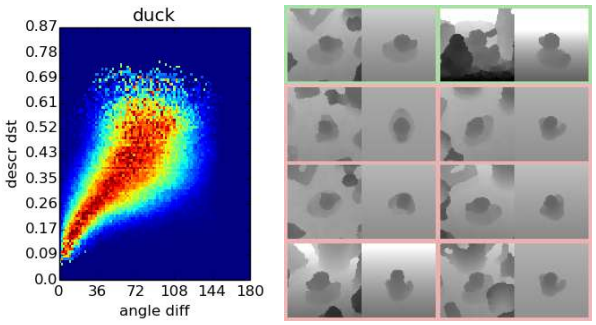


Figure 7. Generalization to objects not seen during training. **Left:** Histogram of correlation between Pose Similarity and Descriptor Distance for the *duck* that was not included during training for this experiment. The recognition rate is only slightly reduced compared to when the duck is used during training. **Right:** Difficult examples of correct and mis-classifications. The duck is mostly confused with the ape, which looks similar in the depth image under some angles.

5. Conclusion

We have shown how to train a CNN to map raw input images from different input modalities to very compact output descriptors using pair-wise and triplet-wise constraints over training data and template views. Our descriptors significantly outperform LineMOD and HOG, which are widely used for object recognition and 3D pose estimation, both in terms of accuracy and descriptor length. Our representation therefore replaces them advantageously. Tests on the capability to generalize to unseen objects also have shown promising results. For further investigation we will make our code available upon request.

References

[1] M. Aubry, D. Maturana, A. Efros, B. Russell, and J. Sivic. Seeing 3D Chairs: Exemplar Part-Based 2D-3D alignement Using a Large Dataset of CAD Models. In *Conference on Computer Vision and Pattern Recognition*, 2014.

[2] A. Bakry and A. Elgammal. Untangling Object-View Manifold for Multiview Recognition and Pose Estimation. In *European Conference on Computer Vision*, 2014.

[3] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference*, June 2010.

[4] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6D Object Pose Estimation using 3D Object Coordinates. In *European Conference on Computer Vision*, 2014.

[5] M. Brown, G. Hua, and S. Winder. Discriminative Learning of Local Image Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[6] S. Chopra, R. Hadsell, and Y. LeCun. Learning a Similarity Metric Discriminatively, with Application to Face Verification. In *Conference on Computer Vision and Pattern Recognition*, 2005.

[7] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *Conference on Computer Vision and Pattern Recognition*, 2005.

[8] T. Dean, J. Yagnik, M. Ruzon, M. Segal, J. Shlens, and S. Vijayanarasimhan. Fast, Accurate Detection of 100,000 Object Classes on a Single Machine. In *Conference on Computer Vision and Pattern Recognition*, 2013.

[9] S. Divvala, A. Efros, and M. Hebert. How Important Are 'Deformable Parts' in the Deformable Parts Model? In *ECCV, Parts and Attributes Workshop*, 2012.

[10] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[11] C. Gu and X. Ren. Discriminative Mixture-of-Templates for Viewpoint Classification. In *European Conference on Computer Vision*, 2010.

[12] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality Reduction by Learning an Invariant Mapping. In *Conference on Computer Vision and Pattern Recognition*, 2006.

[13] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit. Gradient Response Maps for Real-Time Detection of Textureless Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):876–888, May 2012.

[14] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes. In *Asian Conference on Computer Vision*, 2012.

[15] D. Hoiem and S. Savarese. *Representations and Techniques for 3D Object Recognition and Scene Interpretation*. Morgan Claypool Publishers, 2011.

[16] H. Jégou, M. Douze, and C. Schmid. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, January 2011.

[17] K. Lai, L. Bo, X. Ren, and D. Fox. A Scalable Tree-based Approach for Joint Object and Pose Recognition. In *American Association for Artificial Intelligence Conference*, 2011.

[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. *IEEE*, 1998.

[19] T. Malisiewicz, A. Gupta, and A. Efros. Ensemble of Exemplar-SVMs for Object Detection and Beyond. In *International Conference on Computer Vision*, 2011.

[20] J. Masci, D. Migliore, M. M. Bronstein, and J. Schmidhuber. *Registration and Recognition in Images and Videos*, chapter Descriptor Learning for Omnidirectional Image Matching. R. Cipolla et al. Eds., Springer, 2014.

[21] M. Muja and D. Lowe. Scalable Nearest Neighbor Algorithms for High Dimensional Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36, 2014.

[22] S. K. Nayar, S. A. Nene, and H. Murase. Real-Time 100 Object Recognition System. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12):1186–1198, 1996.

[23] D. Nister and H. Stewenius. Scalable Recognition with a Vocabulary Tree. In *Conference on Computer Vision and Pattern Recognition*, 2006.

[24] M. Norouzi, A. Punjanu, and D. Fleet. Fast Exact Search in Hamming Space with Multi-Index Hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(6):1107–1119, 2014.

[25] N. Payet and S. Todorovic. From Contours to 3D Object Detection and Pose Estimation. In *International Conference on Computer Vision*, 2011.

[26] B. Pepik, M. Stark, P. Gehler, and B. Schiele. Teaching 3D Geometry to Deformable Part Models. In *Conference on Computer Vision and Pattern Recognition*, 2012.

[27] R. Pless and R. Souvenir. A survey of manifold learning for images. *IPSJ Transactions on Computer Vision and Applications*, 1:83–94, 2009.

[28] R. Rios-Cabrera and T. Tuytelaars. Boosting Masked Dominant Orientation Templates for Efficient Object Detection. *Computer Vision and Image Understanding*, 120:103–116, 2014.

[29] R. Salakhutdinov and G. Hinton. Learning a Nonlinear Embedding by Preserving Class Neighbourhood Structure. In *International Conference on Artificial Intelligence and Statistics*, 2007.

[30] S. Savarese and L. Fei-Fei. 3D Generic Object Categorization, Localization, and Pose Estimation. In *International Conference on Computer Vision*, 2007.

[31] C. Strecha, A. Bronstein, M. Bronstein, and P. Fua. LDA-Hash: Improved Matching with Smaller Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1), January 2012.

[32] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of momentum and initialization in deep learning. In *Proc. IEEE Intern. Conf. on Machine Learning*, 2013.

[33] M. Tarr and S. Pinker. Mental Rotation and Orientation-Dependence in Shape Recognition. *Cognitive Phycology*, 21(2):233–282, 1 1989.

[34] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim. Latent-Class Hough Forests for 3D Object Detection and Pose Estimation. In *European Conference on Computer Vision*, 2014.

[35] T. Trzcinski, M. Christoudias, P. Fua, and V. Lepetit. Boosting Binary Keypoint Descriptors. In *Conference on Computer Vision and Pattern Recognition*, June 2013.

[36] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *Conference on Computer Vision and Pattern Recognition*, 2014.

[37] K. Q. Weinberger and L. K. Saul. Fast solvers and efficient implementations for distance metric learning. In *Proc. IEEE Intern. Conf. on Machine Learning*, 2008.