Deep Exemplar 2D-3D Detection by Adapting from Real to Rendered Views

Francisco Massa [1] Bryan C. Russell [2] Mathieu Aubry[1,3]

[1] École des Ponts ParisTech * [2] Adobe Research [3] UC Berkeley

Abstract

This paper presents an end-to-end convolutional neural network (CNN) for 2D-3D exemplar detection. We demonstrate that the ability to adapt the features of natural images to better align with those of CAD rendered views is critical to the success of our technique. We show that the adaptation can be learned by compositing rendered views of textured object models on natural images. Our approach can be naturally incorporated into a CNN detection pipeline and extends the accuracy and speed benefits from recent advances in deep learning to 2D-3D exemplar detection. We applied our method to two tasks: instance detection, where we evaluated on the IKEA dataset [36], and object category detection, where we out-perform Aubry et al. *[3] for "chair" detection on a subset of the Pascal VOC dataset.*

1. Introduction

Recently, Aubry *et al*. [3] performed object category detection by exemplar alignment with a large library of 3D object models. The aligned models often approximately matched the style of the depicted objects and allowed 3D information, such as hidden object surfaces and object pose, to be propagated to the 2D images. Such a result is useful for 3D scene reasoning and may potentially be used in applications such as object manipulation in robotics and model-based object image editing in computer graphics [30].

Despite recent progress on 2D-3D matching and retrieval [27, 35, 48], detection by 2D-3D alignment lags behind state-of-the-art object detection systems based on annotated images, e.g., R-CNN [18], in terms of accuracy and speed. We see two primary reasons for this gap in performance: (i) there is a large appearance gap between views rendered from CAD models and real images; and (ii) 2D-based object detection has benefited from recent successes of convolutional neural networks (CNNs) [31, 32]. This work addresses both issues.

The appearance gap across two different domains encountered in 2D-3D alignment is not unique to our problem and can be found in other tasks, e.g., when learning on one dataset and testing on another [53]. To bridge such appearance gaps, a number of cross-domain adaptation algorithms have been developed, e.g. [54]. Building on the successes of these methods, we present an approach that learns to adapt natural image features for the task of 2D-3D exemplar detection. We hypothesize that, given the features of a natural image depicting an object, it is possible to infer the features of a corresponding rendered view of an object CAD model with similar style and pose. Note that similar reasoning has been explored in recent work to predict CAD object features for a different view [50].

To achieve our adaptation learning goal, we need a large training set of aligned natural image and rendered view pairs depicting a similar object. While there are existing datasets with aligned pairs, e.g., IKEA [36] and Pascal3D [57], such datasets are either relatively small or have aligned models that coarsely approximates the object style. To overcome these challenges, we make use of the ability to render views from CAD models and composite with natural images, which allows us to create a large training set. The composite image and rendered view pairs form training data with which to learn the feature adaptation, and have been similarly employed in prior work to train 2D object detectors over CAD renders [40, 41] and predict object pose [49].

In learning the adaptation, we adopt a formulation similar to Lenc and Vedaldi [33], which studied the equivariance of image features under geometric deformations of the image. Our work can be seen as an extension of their approach beyond geometric transformations. We show that the adaptation can be incorporated as a module in a CNN-based object detection pipeline. Furthermore, we show that precomputed features of the rendered views can be added as a fully-connected layer in a CNN, which brings the benefits of accuracy and speed from recent advances in deep learning to 2D-3D exemplar detection.

Contributions. Our contributions are twofold:

*Université Paris-Est, LIGM (UMR CNRS 8049), ENPC, F-77455 Marne-la-Valle. This work was carried out in IMAGINE, a joint research project between Ecole des Ponts ParisTech (ENPC) and the Centre Scientifique et Technique du Bâtiment (CSTB).

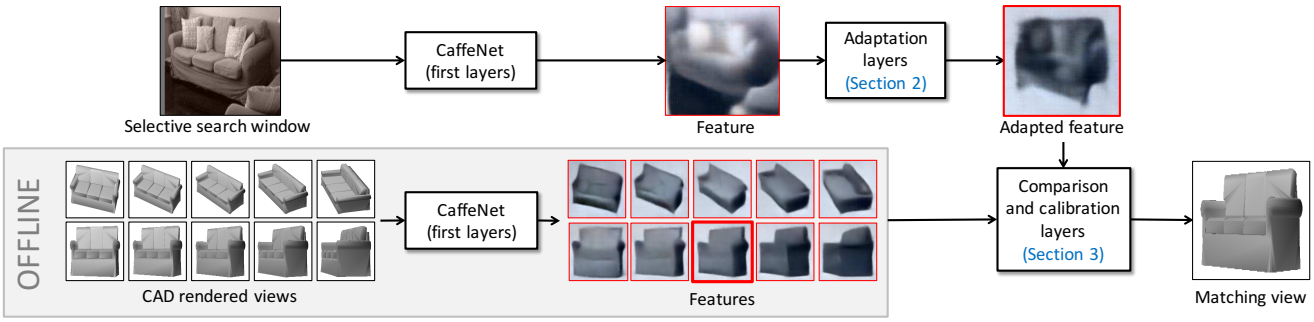


Figure 1: **System overview.** Our system takes as input individual 2D image object proposal windows (top-left) generated by the selective search algorithm [55]. The image window is passed through the initial layers of a pre-trained CaffeNet model [29] to generate a feature vector (top-middle). Here, we visualize CNN features using the inversion network of [13] (outlined in red), which infers the original image given a CNN layer's response. In an offline step (bottom-left), we similarly pass rendered views of a library of 3D object CAD models through the initial layers of CaffeNet and record their responses. As there is a domain gap between the appearance of natural images and rendered views of CAD models, we learn to adapt the features for a natural image to better align to those of CAD models (top-right). We compare the features and return the view that best matches the style and pose of the input image (bottom-right).

- We introduce a cross-domain adaptation approach for 2D-3D exemplar detection using generated pairs of rendered views of CAD models and composite views with natural background. Our adaptation routine adapts features of natural images depicting objects to more closely match features of CAD model rendered views.
- We show how our adaptation routine can be incorporated into a CNN-based detection pipeline, which leads to an increase in accuracy and speed for 2D-3D exemplar detection.

We evaluated our method on the tasks of CAD instance retrieval on the IKEA dataset [36] and on 2D-3D object class detection on the Pascal VOC subset used in Aubry *et al*. [3]. We show state-of-the-art exemplar detection performance on IKEA instances and out-perform the discriminative element approach of Aubry *et al*. [3] both in terms of accuracy and speed. The extended annotations for the IKEA object dataset, a new diverse dataset of textured and non-textured rendered views of CAD models we used to learn the adaptation, and our full code are available at `http://imagine.enpc.fr/~suzano-f/exemplar-cnn/`.

1.1. Related Work

A 3D understanding of 2D natural images has been a problem of interest in computer vision since its very beginning [43]. Our work is in line with traditional geometry-centric approaches for object recognition based on alignment [39]. There has been a number of successful approaches for instance-level recognition, e.g., [10, 34, 45], typically based on SIFT matching [37] with geometric constraints. More recent approaches have leveraged contour-based representation to align skylines [5] and statues [2]. Furthermore, simplified or parametric geometric models have been used for category recognition/detection [15, 20, 24, 42, 58, 60]. We will focus our discussion in this section on prior work using CAD models for category recognition and 2D-3D alignment.

Rendered views from CAD models have been used as input for training an object class detector [40, 41, 51] or for viewpoint prediction [49]. Most similar to us are approaches that align models directly to images. Examples include alignment of IKEA furniture models to images [36], exemplar-based object detection [38] by matching discriminative elements [3, 9], and using hand-crafted features for retrieving CAD models for depth prediction [48] and compositing from multiple models [27]. Also related are approaches for CAD retrieval given RGB-D images (e.g., from Kinect scans) [21, 47]. More recently there has been work to enrich the feature representation for matching and alignment using CNNs, which include CAD retrieval based on CNN responses (e.g., AlexNet [31] "pool5" features) [4], learning a transformation from CNN features to light-field descriptors for 3D shapes [35], and training a Siamese network for style retrieval [6]. Building on efficient CNN-based object class detection, e.g., R-CNN [18], our approach extends the above CNN-based approaches for efficient CAD-exemplar detection.

Bridging two very different image modalities is a classic problem for alignment [28]. Past approaches have addressed this problem using two main strategies. A first line of work has used manually-designed feature detectors and adapted them, for example by adding a mask, so that they focus on the information available in both CAD models and real images [3, 9, 56]. Another line of work has focused on increasing the realism of rendered views, e.g., by extracting likely textures and background from annotated images [40, 41, 49, 51]. Domain adaptation approaches have been formulated for

CNNs [7, 25, 44, 16], most recently for object detection [26], fine tuning across tasks [54], and, in a contemporary work, transfer learning from RGB to optical flow and depth [22]. Most similar to our approach is domain adaptation with CAD [51], which adapted hand-crafted features (HOG [12]) for object detection. We formulate a generic domain adaptation approach over image features, which can be applied to hand-crafted features, e.g., HOG [12] or CNN responses.

1.2. Overview

Figure 1 shows our 2D-3D exemplar detection pipeline. We start by computing CNN features for an image corresponding to a selective search window, along with CNN features for rendered views of CAD models. Due to the large appearance gap across the two domains, we learn how to adapt features of natural images to better match features for rendered views (Section 2). We then compare the adapted features with calibrated rendered view features to obtain matching scores for each rendered view (Section 3). Note that our detection pipeline can be implemented as a CNN. An evaluation of our approach is in Section 4.

2. Adapting from real to rendered views

In this section we describe our approach for adapting features extracted from real images to better correspond to features extracted from rendered views of CAD models. Our approach is general and can be applied to any image feature set, e.g., HOG [12] and CNN-based features [32]. We adapt from real images to rendered views (and not from rendered to real) since it is likely more difficult to hallucinate features corresponding to missing image details, such as the surrounding context of an object and its texture, than to remove them.

Formally, we seek to learn a transformation ϕ over the features of real images. Intuitively ϕ is a projection of the real image feature space to the space of features from CAD rendered views. Ideally, ϕ has the property of mapping a given real image feature depicting an object of interest to features of rendered views of CAD object models with the same geometry, style, and pose.

Suppose we have as input a set of N pairs of features $\{(x_i, y_i)\}_{i=1}^N$ corresponding to examples of real images and rendered views of well-aligned CAD models, respectively. We seek to minimize the following cost over ϕ:

$$L(\phi) = -\sum_{i=1}^{N} S\left(\phi\left(x_i\right), y_i\right) + R(\phi), \tag{1}$$

where S denotes a similarity between the two features $\phi(x_i)$ and y_i, and R is a regularization function over ϕ. Note that in the case where ϕ is an affine transformation, our formulation is similar to the one of Lenc and Vedaldi [33] where a mapping was learned given image pairs to analyze the equivariance of CNN features under geometric transformations.

Adaptation. While the simplest choice for ϕ is an affine transformation, which we use as a reference in our experiments, we also tested more constrained and complex transformations. We focused on transformations that could be formulated as CNN layers, and in particular successions of convolutional and ReLU layers. Note that considering more complex transformations also increases the risk of overfitting. Similar to Lenc and Vedaldi [33] we attempted to constrain the structure of the transformation and its sparsity. This is easily done in a CNN by replacing a fully-connected layer by a convolutional layer with limited support, which implies translation invariance in the adaptation. We found that the best-performing transformation was only a slight modification of the affine transformation:

$$\phi(x) = ReLU(Ax + b), \tag{2}$$

where $ReLU(x) = \max(0, x)$ is the element-wise maximum over zero. We observed that applying the $ReLU$ function consistently improved results, and is in agreement with state-of-the-art CNN architecture design choices for object recognition.

Similarity. We tried both L_2 and squared-cosine similarity to measure the similarity in Equation (1). We found that the squared-cosine similarity $S(a, b) = -\left(1 - \frac{a^T b}{\|a\|\|b\|}\right)^2$ leads to better results. This is expected, since cosine similarity is known to work better when comparing CNN features [4], but also because we later used the cosine distance to compare real and synthetic features (c.f. Section 4). This result is also consistent with the observation of the importance of task-specific similarities in Lenc and Vedaldi [33].

Training data details. Our adaptation formulation requires a large training set of well-aligned pairs of images and rendered views of CAD models matching the style and pose of depicted objects. Such a dataset is difficult to acquire. While existing datasets have object CAD models aligned to images closely matching the depicted object pose [57, 19], the models are often not similar in style. Recent work on accurate alignment to 3D models by composition [27] and semi-automatic 3-sweep modeling [8] are promising approaches for obtaining accurate image-model alignments, but no large-scale results are yet available.

Figure 2: Examples of image pairs used for learning the adaptation.

Instead, we build on recent approaches for effective training from rendered views [40, 49] to render views of CAD

models and composite on natural image backgrounds. This gives us access to virtually unlimited training data. The backgrounds provide "natural-looking" surrounding context and encourages the transformation ϕ to learn to subtract away the background context. To avoid color artifacts in the composite images, we used gray-scale image pairs and also used gray-scale images at test time. Note that contrary to prior approaches using manually-annotated scenes to increase the realism of the composite [40, 41], we do not directly use any object annotation in our background selection process. Figure 2 shows four representative image pairs from our adaptation data (top – object rendered views; bottom – rendered views composited with natural image backgrounds).

For the 3D models, we found that using a diverse database comprising several object categories produced better results than focusing on a target set of 3D models we aim to detect. We used as reference in all our experiments the textureless rendered views from Aubry and Russell [4] to train the adaptation.

Implementation details. We used a small L_2 regularization R in all our experiments and found that it improved our results despite our very large training sets. We trained ϕ using stochastic gradient descent within the Torch7 framework [11]. We used a weight decay of 5e-4, corresponding to the L_2 regularization, a momentum 0.9, and mini-batch size of 128. We started with a learning rate of 1 and reduced it every 15 epochs by a factor of 10 until convergence.

3. Exemplar detection with CNNs

In this section we show how the adaptation procedure in Section 2, together with feature computation and exemplar-based retrieval, can be incorporated into an efficient CNN-based detection routine, similar to R-CNN [18], for 2D-3D exemplar detection. For a given input image, we seek to detect the bounding box location of an object in the image and return a corresponding CAD model having similar style, along with the pose of the depicted object.

Exemplar-detection pipeline. Following the initial part of the R-CNN object detection pipeline [18], we first extract a set of selective search windows [55] and compute CNN responses x at an intermediate layer (e.g., CaffeNet *pool5* layer) for each window. We then apply our adaptation ϕ to these features and compare the results $\phi(x)$ to the features of different CAD model rendered views. Let $s_i(x) = S(\phi(x), y_i)$ be the similarity between $\phi(x)$ and the features y_i of the ith rendered view.

As shown in Aubry *et al*. [3], calibration is an important step for comparing similarity across different views and CAD models. Starting from the initial similarity score $s_i(x)$, we apply their affine calibration routine to compute a new calibrated similarity $s'_i(x) = c_i s_i(x) + d_i$. The scalar parameters c_i and d_i are selected using a large set of random patches such that $s'_i(x_0) = -1$ and $s'_i(x_1) = 0$, where x_0 and x_1 correspond to random patch features with mean and 99.99-percentile similarity scores, respectively.

	L_2 distance	Dot product	Cosine distance
Pool3	57.7	46.0	61.3
Pool4	57.7	47.5	**65.0**
Pool5	38.7	54.7	60.6
fc6	38.7	59.9	59.9
fc7	48.2	61.3	52.6

Table 1: Instance retrieval accuracy on the IKEA dataset [36] over different CaffeNet layers (rows) and distances (cols).

We take advantage of the fact that in an exemplar-based detection setup the expected aspect ratio of the alignments are known. We remove candidate rendered-view alignments when the aspect ratio has a difference of more than 10% between the selective search window and rendered view. Finally, we rank the remaining alignments by their score $s'_i(x)$ and perform non-maximum suppression to obtain the final detections.

CNN implementation. Figure 1 shows our CNN for 2D-3D exemplar detection. Our network starts with layers corresponding to a CNN trained on a different task (e.g., CaffeNet [29] trained for ImageNet classification in our experiments) until an intermediate layer (e.g.,"pool5"). Next, the resulting features pass through the adaptation layers corresponding to ϕ, implemented as a fully-connected layer followed by a ReLU.

The resulting adapted features are compared to the exemplar rendered-view features. Several standard similarity functions, such as dot product and cosine similarity, can be implemented as CNN layers. For example, cosine similarity can be implemented by a feature-normalization layer followed by a fully-connected layer. The weights of the fully-connected layer correspond to a matrix Y of stacked unit-normalized features for the exemplar rendered views, computed in an offline stage. While the affine calibration could be implemented as an independent layer, we incorporated it directly into the fully-connected layer by replacing the matrix rows by $Y_i \leftarrow c_i Y_i$ and adding a bias d_i corresponding to each row i. The final exemplar rendered-view scores is $Y\phi(x) + d$ given image features x, and can be computed by a single forward pass in a CNN.

4. Experiments

In this section we qualitatively and quantitatively evaluate our method and analyze different design choices. First, we focus on a simpler retrieval task to select the features and similarity function for our detection task (Section 4.1). Then, we present our main results on object-instance and object-class

class	chair *poang*	bookcase *billy1*	sofa *ektorp*	table *lack*	bookcase *billy2*	desk *expedit*	bookcase *billy4*	bed *malm2*	stool *poang*	mAP
Original annotations of [36]										
Number of instances	40	18	13	20	10	8	6	7	7	
Lim *et al*. [36]	27.0	**24.3**	7.3	14.0	**26.6**	18.8	32.6	**22.6**	14.8	**20.89**
DPM [14]	27.5	**24.3**	**12.1**	10.8	13.5	**46.1**	0.1	1.0	0.5	15.10
Ours without adaptation	15.9	1.3	8.4	25.7	0.1	25.5	0.2	0.9	18.3	10.69
Ours with *fc* adaptation	31.1	1.2	9.1	27.0	0.0	13.8	0.4	1.1	23.7	11.94
Ours with *fc*+*ReLU* adaptation	**33.1**	1.1	9.5	**27.1**	0.1	14.5	0.4	1.2	**24.4**	12.38
New annotations										
Number of instances	56	35	15	34	17	7	17	24 *	18	
Lim *et al*. [36]	19.9	**14.8**	**6.4**	9.4	**15.7**	**15.4**	**13.4**	**7.6**	6.4	12.11
Ours with *fc*+*ReLU* adaptation	**35.4**	6.2	**8.2**	**21.4**	0.1	**16.5**	0.4	**10.4**	**28.4**	**14.11**

Table 2: Instance detection performance on the IKEA object dataset [36]. We report average precision using a bounding box overlap threshold of 0.5. Note that some categories reported in [36] have very few annotated examples. We report results for classes that include more than 3 annotated instances. The top part of the table presents results with the original annotation of [36] and the bottom part with our extended annotations. We evaluated the detection outputs provided from [36] using these extended annotations. * The dataset includes three different but similar sizes of the same bed. Since we were not able to differentiate visually between these three kind of beds, all were annotated.

detection by aligning to CAD rendered views, comparing against existing baselines (Section 4.2). Finally, we perform an ablative analysis of our algorithm (Section 4.3) and report computational running time (Section 4.4).

4.1. Instance retrieval

To select CNN features and a similarity function for comparing natural images and CAD rendered views, we consider a retrieval task where, given a cropped image depicting a query object, we seek to return a model corresponding to the object. We consider the IKEA dataset of Lim *et al*. [36], which has CAD models of IKEA object instances manually aligned to their location in images depicting cluttered scenes. The task allows us to compare the performance of different CNN layer responses and similarity functions. The retrieval task is difficult as there are a variety of object poses and perspective effects in the IKEA dataset. To handle the variation in object pose and perspective effects, we rendered 36 azimuth and 7 elevation angles and at 3 different distances for each object. Note that the rendered views cover many possible viewpoints and perspective effects, but it does not cover all cases.

We extracted CNN features from CaffeNet [29] for our experiments. While more recent, deeper networks [46, 52] may yield better results (e.g., a boost of 4% is obtained for retrieval using the cosine distance on VGG *pool4* features), we illustrate the basic design choices using the shallower CaffeNet model. We also expect better results using the last layers of a network fine tuned for object-class detection, e.g., R-CNN fine tuned for Pascal detection [18]. We chose not to consider such a network to focus on the general case where natural images of related object classes do not have to be annotated for training. We performed retrieval using features extracted from the *conv3* to *fc7* layers of CaffeNet after ReLU (and without adaptation). We applied max-pooling to the *conv3* and *conv4* features to keep their dimensionality relatively small and avoid memory issues in our detection pipeline. We denote the resulting features after pooling as *pool3* and *pool4*. We compared three similarity functions for our experiments: L_2 distance, dot-product similarity, and cosine distance.

We report retrieval accuracy in Table 1. Notice that performance for cosine distance is best with *pool4* features, and decreases with the higher layers, while performance increases with the higher layers for dot-product similarity. Based on these results, we used cosine distance over *pool4* features in all our experiments. Moreover, *conv4* features are known to be relatively generic features [1, 59] and make little to no use of the network knowledge gained on specific objects, such as chairs, sofas, and beds, in ImageNet classification.

4.2. Detection

In this section, we demonstrate our feature-adaptation algorithm for 2D-3D detection. We consider two tasks: object-instance and object-category detection by 2D-3D alignment. For object-instance detection, we evaluated on the IKEA dataset [36]. For object-category detection, we evaluated on the subset of Pascal VOC containing "chairs" used in Aubry *et al*. [3]. We show qualitative and quantitative results on both benchmarks and compare against prior work.

Figure 3: Top 10 detections for the *billy1* IKEA model. Note that the first good detection is counted as negative with the original annotation because it was not annotated in the dataset. Most of our other detections are different bookcases or parts of bookcases.

4.2.1 Object-instance detection by 2D-3D alignment

For object-instance detection by 2D-3D alignment, we evaluated our approach on the IKEA dataset and followed the detection protocol outlined in Lim *et al*. [36]. We report average precision detection performance in Table 2(top), along with baselines for this task. It can be seen that we clearly improve over the baselines for several well-represented classes. However, our mAP is smaller than the baselines. We will show that this is due to two main effects: a chance factor for classes where very few objects were annotated or had missing annotations, and a failure of our algorithm on "bookcases", which we analyze in detail.

Dataset and additional annotations. Two important issues when using the IKEA object dataset for evaluating instance detection are (i) its relatively small size (we report the number of annotated instances in the first line of table 2), and (ii) the partial annotations made available, with a maximum of one object per image when several are often present. To partly address these issues, we annotated all instances in the 288 test images for the classes that included more than three instances in the original dataset (except for "Billy3", where the detections reported in [36] appear to correspond to a different model). This increases the number of annotated objects of the selected classes from 129 to 223. We report our results on our new extended annotation set in Table 2(bottom). We will release these new annotation to allow further comparisons. With these extended annotations our mAP is similar to [36], but with strong differences in the performance for the different objects. We have similar results or clear improvements (shown in blue in table 2) for most classes, but much lower performance for bookcases (shown in red in table 2).

Failures on bookcases. Here we analyze our failures for bookcases, which are very poor in contrast to other categories

Training with real data			
DPM [14]	41.0		
R-CNN [18]	44.8		
R-CNN + SVM [18]	54.5		
Training with CAD data			
Aubry *et al*. [3]	33.9		
Peng *et al*. [40] (W-UG)	29.6		
	Adaptation	No Adaptation	
		Comp.	White
Logistic *pool4*	12.9	3.7	1.4
Logistic *fc7*	26.6	9.2	14.0
Ours, no calibration	5.6	6.0	3.2
Ours with calibration	52.3	36.4	17.9

Table 3: Average precision for chair detection on Pascal VOC subset [3]. Our best method outperforms the baselines of [3] by 18%. "White" column corresponds to synthetic images on white background. "Comp" column corresponds to synthetic images composited on real-image backgrounds.

where they matched or exceeded the baselines. Inspecting the bookcases missed by our algorithm, which are available in the project webpage, almost all of them consist of highly cluttered examples, e.g., bookcases filled with books of different colors. We verified that for our extended annotations, only 14% of *billy1* bookcases are empty, whereas *billy2* and *billy4* do not have any non-cluttered examples in the dataset. Looking at our top false positives in Figure 3 confirms this, since we find many parts of empty bookcases or bookcases from other categories.

4.2.2 Object-category detection by 2D-3D alignment

For object-category detection by 2D-3D alignment, we evaluated our approach on the subset of the Pascal VOC dataset containing images of non-difficult, non-occluded, and non-truncated "chairs" used in Aubry *et al*. [3], and aligned to their chair rendered views. We followed their detection protocol and report average precision for the detection task. We compare our performance against the baseline of Aubry *et al*. [3], which also performs detection by 2D-3D alignment. We also report performance of DPM [14] and R-CNN [18] with and without SVM, both without bounding box regression, which were trained on natural images for 2D object detection. As another baseline, we report the performance of a logistic regression classifier trained using synthetic images (with and without adaptation), which is similar in spirit to recent approaches that trains a 2D object detector using synthetic training images [40, 41]. In order to better situate our work with respect to approaches that train a classifier using synthetic images with composite backgrounds [40, 41], we also report results for the following baselines using synthetic images composited with natural-image background as positives, and without adaptation: (a) logistic regression

(a) Without adaptation (b) With adaptation

Figure 4: Top detections without and with adaptation on the Pascal VOC chair subset [3]. Notice that while the alignments are good with and without adaptation, detection without adaptation returns dark chairs having "CAD-like" white backgrounds. Detections with adaptation include brighter objects and cluttered backgrounds.

classifier, (b) our exemplar detector. Finally, we report results for the best performing method of Peng *et al*. [40], corresponding to their W-UG synthetic images.

We report our results in Table 3. With our reference adaptation, our method outperforms all baselines except R-CNN + SVM. We obtain an average precision of 52.3% compared to 41% for DPM, 33.9% for Aubry *et al*. [3] and 29.6% for Peng *et al*. [40]. We also tried using the method of [40] with the chairs from [3], which resulted in 9.0 AP. This difference in performance is likely due to their manual selection of realistic viewpoints and models in the W-UG set.

A more detailed analysis reveals the importance of the adaptation for all the methods based only on CNN features from CAD models. Note that the benefit of using the adaptation is less important when using the fc7 layer for logistic regression. This shows that unsurprisingly *fc7* is less sensitive to the type of representation than *conv4*, and may explain the good results obtained by [40, 41] using the fc7 layers directly. An interesting question is whether the adaptation could be replaced by synthetic images composited with natural-image backgrounds. As can be seen from Table 3, even though the composites help in some cases (notably in our exemplar detector), its performance still lags behind the performance obtained using the adaptation. Note that we used a single background per exemplar view. While one could include more composites per exemplar, this would increase the memory requirements as one would need to store all of the additional exemplars.

(a) Top false positives without adaptation

(b) Top false positives with adaptation

Figure 5: Top-ranked false positives without and with adaptation. Since there were several false positives per image without adaptation, we only show the best ranked for each image. The false positives without adaptation occur on uniform background patches. With adaptation, this effect largely disappears and the false positives correspond to patches that look like chairs or chair parts.

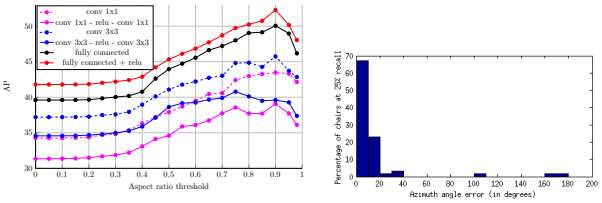
4.3. Ablative analysis

In this section we perform an ablative study of different design choices of our approach.

Influence of adaptation on alignment. In Figure 4, we show the top detections with and without adaptation. Notice that while the non-adapted features have higher detection scores for "CAD-like" images of darker chairs on mostly white background (Fig. 4(a)), the adaptation allows us to detect chairs of all colors in natural cluttered scenes (Fig. 4(b)). Similarly, we show the top false positives in Figure 5. Notice that without adaptation the top false positives correspond to regions with uniform background (Fig. 5(a)), while adaptation has chair-shaped false positives similar to an object detector trained on natural images only (Fig. 5(b)).

Adaptation design. As discussed in Section 3, the adaptation ϕ in Equation (2) can be implemented in a CNN as a fully-connected layer, followed by a ReLU nonlinearity. We seek to study variants of ϕ. Since the *pool4* CaffeNet features maintain spatial bin structure, we consider adaptations with limited spatial support via convolution with 1×1 and 3×3 kernels. We also consider whether to use the ReLU nonlinearity and whether to consider multiple convolutional layers in the adaptation.

Figure 6a shows the average precision for different variants of ϕ as a function of the aspect ratio threshold. Notice that all of the adaptation variants we tried performed better than without adaptation (17.9% AP). Imposing adaptations with limited spatial support (*conv*) performed worse than a fully-connected layer. This can be understood by considering that the effect of the projection depends on the interpretation of the image as foreground object and background as clutter, a task that can be better performed globally. Using two lay-



(a) (b)

Figure 6: (a) Average precision for different adaptations as a function of the aspect ratio threshold. (b) Azimuth angle error. Best viewed in the electronic version.

ers for the adaptation degraded performance. Note that we observed the validation loss was better optimized using two layers. We believe this effect is due to the synthetic nature of our training data, which only approximates the relation between real and synthetic images. Finally, we found that adding a ReLU after the convolutional layer consistently increased the performance. The use of a single fully-connected layer followed by a ReLU produced the best performance.

Aspect ratio. Figure 6a shows the evolution of the average precision as a function of aspect ratio threshold for different projections on the Pascal VOC subset detection experiment. As expected, increasing the threshold first improves the results because it removes many false positives.The results are then relatively stable between 0.75 and 0.9 since both positives and negatives are discarded. Finally, the performance drops for higher thresholds as more true positives get discarded. In all our experiments, we used an aspect-ratio threshold of 0.90.

Evaluation of the retrieved pose. We conducted the same experiment as in Aubry *et al*. [3] to evaluate the quality of the retrieved poses. For ground truth we used the pose annotations from Pascal3D [57]. Figure 6b shows a histogram of azimuth angle errors at 25% recall (similar to Fig. 6 in Aubry *et al*. [3]). Our algorithm returns an azimuth angle within 20° of the ground truth for 90% of the examples, compared with 87% for Aubry *et al*. [3].

Number of rendered views. We studied the relative importance of the CAD model dataset size on the final detection performance by conducting experiments over the set of 86K renders from Aubry *et al*. [3]. We randomly selected increasing subsets of all rendered views (Table 4(a)), and randomly selected increasing numbers of CAD models and used all their 62 rendered views (Table 4(b)). Notice that performance increases with the number of CAD renders, as expected. Interestingly, the diversity of the CAD models plays an important role in the final detection score. For roughly the same number of rendered views, 5 CAD models (for a total of 310 views) performs considerably worse than

(a) Number of rendered views						
n	200	500	1k	2k	10k	86k
AP	33.3	37.6	41.3	44.8	45.7	50.0
(b) Number of CAD models						
n	5	10	20	40	160	1393
AP	21.7	26.6	29.8	33.9	44.6	50.0

Table 4: Detection AP in the subset of Pascal VOC chair subset [3] for the fully-connected projection as a function of (a) the number of CAD rendered views and (b) the number of unique CAD models used.

200 random views.

4.4. Computational run time

Our system runs in computational time similar to R-CNN [18] if all the CAD rendered views fit into GPU memory. Excluding the time to compute bounding box proposals, we can align a test image to 2K rendered views in approximately 9.5 seconds on a GeForce GTX980 graphics card. We can align to more views at the expense of copying precomputed rendered view features to the GPU memory. This can be overcome with larger-memory graphics cards or by running on parallel cards. For 80K rendered views, our approach takes around 52 seconds. Similar to recent fast CNN detection pipelines [23, 17], our timings could be further optimized by reusing the convolutional features for each bounding box, which could potentially reduce the computational time to a fraction of a second. Filtering by aspect ratio before comparing the features could also reduce the number of tests to perform, especially in the case of very large number of 3D views. Note that even without these improvements, our computational run times are much faster than those presented in Aubry *et al*. [3].

5. Conclusion

We demonstrated an end-to-end CNN for 2D-3D exemplar detection. We showed that an adaptation of image features to closely match features of rendered views of CAD models is essential to its success. Our adaptation approach is agnostic to the feature set and could potentially benefit other 2D-3D detection methods.

6. Aknowledgments

We thank Joseph Lim, who shared with us his IKEA detection outputs, which allowed us to compare against his approach using our extended annotations. We also wish to thank Alyosha Efros and Renaud Marlet for fruitful discussions. This work was partly supported by ANR project Semapolis ANR-13-CORD-0003, Intel, a gift from Adobe, and hardware donation from Nvidia.

References

[1] P. Agrawal, R. Girshick, and J. Malik. Analyzing the performance of multilayer neural networks for object recognition. In *Computer Vision–ECCV 2014*, pages 329–344. Springer, 2014. 5

[2] R. Arandjelović and A. Zisserman. Smooth object retrieval using a bag of boundaries. In *ICCV*, 2011. 2

[3] M. Aubry, D. Maturana, A. A. Efros, B. C. Russell, and J. Sivic. Seeing 3D chairs: Exemplar part-based 2D-3D alignment using a large dataset of CAD models. In *CVPR*, 2014. 1, 2, 4, 5, 6, 7, 8

[4] M. Aubry and B. C. Russell. Understanding deep features with computer-generated imagery. In *ICCV*, 2015. 2, 3, 4

[5] G. Baatz, O. Saurer, K. Köser, and M. Pollefeys. Large scale visual geo-localization of images in mountainous terrain. In *ECCV*, 2012. 2

[6] S. Bell and K. Bala. Learning visual similarity for product design with convolutional neural networks. *ACM Transactions on Graphics (Proceeding of SIGGRAPH)*, 2015. 2

[7] Y. Bengio. Deep learning of representations for unsupervised and transfer learning. In *JMLR Workshop on Unsupervised and Transfer Learning*, 2012. 3

[8] T. Chen, Z. Zhu, A. Shamir, S.-M. Hu, and D. Cohen-Or. 3-sweep: extracting editable objects from a single photo. *ACM Transactions on Graphics (TOG)*, 32(6):195, 2013. 3

[9] C. B. Choy, M. Stark, S. Corbett-Davies, and S. Savarese. Object detection with 2D-3D registration and continuous viewpoint estimation. In *CVPR*, 2015. 2

[10] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total Recall: Automatic query expansion with a generative feature model for object retrieval. In *ICCV*, 2007. 2

[11] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011. 4

[12] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *CVPR*, 2005. 3

[13] A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. In *CVPR*, 2016. 2

[14] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010. 5, 6

[15] S. Fidler, S. Dickinson, and R. Urtasun. 3D object detection and viewpoint estimation with a deformable 3D cuboid model. In *NIPS*, 2012. 2

[16] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1180–1189, 2015. 3

[17] R. Girshick. Fast r-cnn. In *International Conference on Computer Vision (ICCV)*, 2015. 8

[18] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 2, 4, 5, 6, 8

[19] R. Guo and D. Hoiem. Beyond the line of sight: labeling the underlying surfaces. In *Computer Vision–ECCV 2012*, pages 761–774. Springer, 2012. 3

[20] A. Gupta, A. A. Efros, and M. Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *ECCV*, 2010. 2

[21] S. Gupta, P. A. Arbeláez, R. B. Girshick, and J. Malik. Aligning 3D models to RGB-D images of cluttered scenes. In *CVPR*, 2015. 2

[22] S. Gupta, J. Hoffman, and J. Malik. Cross modal distillation for supervision transfer. In *CVPR*, 2016. 3

[23] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Computer Vision–ECCV 2014*, pages 346–361. Springer, 2014. 8

[24] M. Hejrati and D. Ramanan. Analyzing 3D objects in cluttered images. In *NIPS*, 2012. 2

[25] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *NIPS Deep Learning Workshop*, 2014. 3

[26] J. Hoffman, S. Guadarrama, E. Tzeng, R. Hu, J. Donahue, R. Girshick, T. Darrell, and K. Saenko. LSDA: Large scale detection through adaptation. In *NIPS*, 2014. 3

[27] Q. Huang, H. Wang, and V. Koltun. Single-view reconstruction via joint analysis of image and shape collections. *ACM Transactions on Graphics (Proceeding of SIGGRAPH)*, 34(4), 2015. 1, 2, 3

[28] M. Irani and P. Anandan. Robust multi-sensor image alignment. In *ICCV*, 1998. 2

[29] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM, 2014. 2, 4, 5

[30] N. Kholgade, T. Simon, A. Efros, and Y. Sheikh. 3d object manipulation in a single photograph using stock 3d models. *ACM Transactions on Graphics (TOG)*, 33(4):127, 2014. 1

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2

[32] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551, Dec. 1989. 1, 3

[33] K. Lenc and A. Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *CVPR*, 2015. 1, 3

[34] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua. Worldwide pose estimation using 3D point clouds. In *ECCV*, 2012. 2

[35] Y. Li, H. Su, C. R. Qi, N. Fish, D. Cohen-Or, and L. J. Guibas. Joint embeddings of shapes and images via CNN image purification. *ACM Transactions on Graphics (Proceeding of SIGGRAPH Asia)*, 2015. 1, 2

[36] J. J. Lim, H. Pirsiavash, and A. Torralba. Parsing IKEA objects: Fine pose estimation. In *ICCV*, 2013. 1, 2, 4, 5, 6

[37] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 2

[38] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011. 2

[39] J. L. Mundy. Object recognition in the geometric era: A retrospective. In *Toward Category-Level Object Recognition, volume 4170 of Lecture Notes in Computer Science*, pages 3–29. Springer, 2006. 2

[40] X. Peng, K. Saenko, B. Sun, and K. Ali. Learning deep object detectors from 3D models. In *ICCV*, 2015. 1, 2, 3, 4, 6, 7

[41] B. Pepik, R. Benenson, T. Ritschel, and B. Schiele. What is holding back convnets for detection? In *Pattern Recognition*, pages 517–528. Springer, 2015. 1, 2, 4, 6, 7

[42] B. Pepik, M. Stark, P. Gehler, and B. Schiele. Teaching 3D geometry to deformable part models. In *CVPR*, 2012. 2

[43] L. Roberts. Machine perception of 3-D solids. In *PhD. Thesis*, 1965. 2

[44] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014. 3

[45] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3D object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *IJCV*, 66(3):231–259, 2006. 2

[46] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 5

[47] S. Song and J. Xiao. Sliding shapes for 3d object detection in depth images. In *ECCV*, 2014. 2

[48] H. Su, Q. Huang, N. Mitra, Y. Li, and L. Guibas. Estimating image depth using shape collections. *ACM Transactions on Graphics (Proceeding of SIGGRAPH)*, 33(4), 2014. 1, 2

[49] H. Su, C. Qi, Y. Li, and L. Guibas. Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views. In *ICCV*, 2015. 1, 2, 3

[50] H. Su, F. Wang, E. Yi, and L. J. Guibas. 3d-assisted feature synthesis for novel views of an object. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2677–2685, 2015. 1

[51] B. Sun and K. Saenko. From virtual to reality: Fast adaptation of virtual object detectors to real domains. In *BMVC*, 2014. 2, 3

[52] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. 5

[53] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *CVPR*, 2011. 1

[54] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks. In *ICCV*, 2015. 1, 3

[55] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *IJCV*, 2013. 2, 4

[56] D. Vazquez, A. M. Lopez, J. Marin, D. Ponsa, and D. Geronimo. Virtual and real world adaptation for pedestrian detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(4):797–809, 2014. 2

[57] Y. Xiang, R. Mottaghi, and S. Savarese. Beyond PASCAL: A Benchmark for 3D Object Detection in the Wild. In *WACV*, 2014. 1, 3, 8

[58] J. Xiao, B. Russell, and A. Torralba. Localizing 3D cuboids in single-view images. In *NIPS*, 2012. 2

[59] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 27 (NIPS '14)*, pages 3320–3328. Curran Associates, Inc., 2014. 5

[60] M. Zia, M. Stark, B. Schiele, and K. Schindler. Detailed 3D representations for object recognition and modeling. *IEEE PAMI*, 2013. 2