Searching and Matching Texture-free 3D Shapes in Images

Shuai Liao
QUVA Lab, University of Amsterdam
s.liao@uva.nl

Efstratios Gavves
QUVA Lab, University of Amsterdam
egavves@uva.nl

Cees G. M. Snoek
QUVA Lab, University of Amsterdam
cgmsnoek@uva.nl

ABSTRACT

The goal of this paper is to search and match the best rendered view of a texture-free 3D shape to an object of interest in a 2D query image. Matching rendered views of 3D shapes to RGB images is challenging because, 1) 3D shapes are not always a perfect match for the image queries, 2) there is great domain difference between rendered and RGB images, and 3) estimating the object scale versus distance is inherently ambiguous in images from uncalibrated cameras. In this work we propose a deeply learned matching function that attacks these challenges and can be used for a search engine that finds the appropriate 3D shape and matches it to objects in 2D query images. We evaluate the proposed matching function and search engine with a series of controlled experiments on the 24 most populated vehicle categories in PASCAL3D+. We test the capability of the learned matching function in transferring to unseen 3D shapes and study overall search engine sensitivity w.r.t. available 3D shapes and object localization accuracy, showing promising results in retrieving 3D shapes given 2D image queries.

KEYWORDS

3D-2D Matching; 3D Shape Retrieval; Texture-free

ACM Reference Format:
Shuai Liao, Efstratios Gavves, and Cees G. M. Snoek. 2018. Searching and Matching Texture-free 3D Shapes in Images. In *ICMR '18: 2018 International Conference on Multimedia Retrieval, June 11–14, 2018, Yokohama, Japan.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3206025.3206057

1 INTRODUCTION

The goal of this paper is to search and match the rendered view of a texture-free 3D shape to an object of interest in a 2D image. Matching shapes to image content has a long tradition in content-based image retrieval, *e.g.* [17, 27], where queries have been entered by sketching [13], by combining sketch and keywords [8], or by a provided 3D shape [1]. We build on this heritage and query a dataset of 3D shapes based on an image, but we also recognize and localize an object of interest in the image and match the object to its corresponding 3D model, so as to arrive at an alignment of the 3D shape in the 2D image as precise as possible, see Fig. 1.

Searching and matching a 3D shape to an object in a real-world 2D image is challenging because: *(a)* finding the 3D shape that has the exact shape as the objects in the image is not always possible, even when large 3D shape libraries, *e.g.* [4], are available, *(b)* matching a rendered image from a 3D shape with a real image is known to suffer from domain shift: the real image may have different texture, lighting condition, shadow and complex natural background, and *(c)* estimating the object scale versus distance is inherently ambiguous in images taken from uncalibrated cameras, where the camera intrinsic matrix is unknown. In this paper we study the influence of these challenges with respect to the searching and matching of 3D shapes in images.

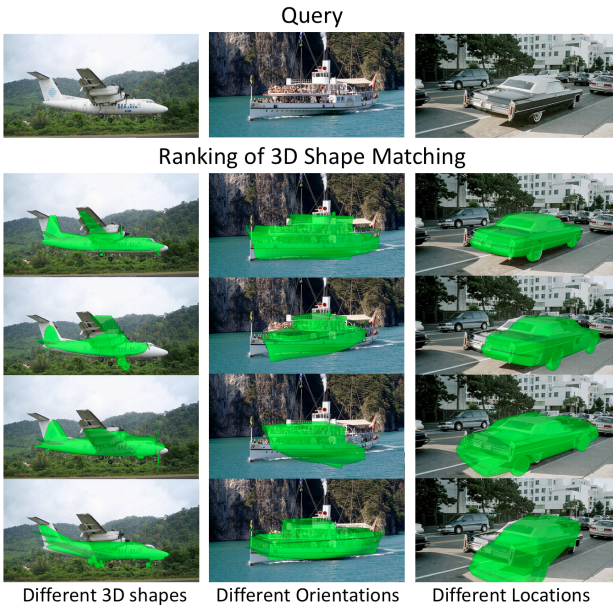

Figure 1: This paper strives to find the rendered view of a texture-free 3D shape that best matches an object of interest in a 2D query image. This matching is challenging because the 3D shapes are not always perfect for the image query (aeroplane, boat, car), there is an obvious domain difference between the appearance of the RGB object and the texture-free rendered image, and the intrinsic camera parameters are unknown. We propose and evaluate a deeply learned matching function that attacks these challenges and can be used as a search engine that finds and matches 3D shapes to objects in 2D images.

1.1 Related Work

We are inspired by recent progress, mostly reported in computer vision venues, were several sophisticated methods for 3D to 2D object matching have been presented, *e.g.* [2, 3, 9]. Most methods rely their matching on exemplar classifiers, which are trained to learn an absolute relation between particular 3D model views and

ICMR '18, June 11–14, 2018, Yokohama, Japan

ACM ISBN 978-1-4503-5046-4/18/06…$15.00
https://doi.org/10.1145/3206025.3206057

particular RGB appearances via texture-sensitive features, such as HOG [2, 3]. In the seminal work of [2], for example, Aubry *et al.* rely on 3D models of chairs from image search engines to align 3D models to 2D objects. The authors propose an LDA-based exemplar classifier trained on HOG features of textured 3D models to detect the 3D model and the pose that best matches real chair images. Unfortunately, the exemplar classifiers are trained for specific models, textures, and specific poses, namely they learn a specific classifier for each 3D model, its different poses and RGB appearances separately. Hence, novel 3D models or poses cannot be accommodated without retraining, limiting the applicability of the learned model for general-purpose 3D shape retrieval.

In [18] a method is described for aligning perfectly matched 3D shapes of IKEA furniture on RGB living room images, be it they rely on parts that must be defined and annotated a priori on the perfect matched 3D shapes. In [21] 3D models are reduced to 3D cuboids, for object categories with box-like geometry, like cars, thus making inference easier. These methods learn absolute relations between *textured* 3D models and the respective 2D objects available at training. Having textured 3D models can be beneficial, *e.g.* if the texture of the object in the 2D image happens to be similar. But it can also limit the algorithm, by biasing it to retrieving 3D models that only share similar textures regardless of apparent differences in shape geometry. Even with large-scale 3D shape datasets such as [4], finding similar textured 3D models that match objects as they appear in images in the wild is still limited by the stored variation in the dataset. What is more, the texture quality and style is not necessarily consistent, as they are typically designed by different 3D artists. Consequently, texture-based methods are constrained to textures already observed, and they are unable, nor intended, to transfer their knowledge to new 3D models or poses at test time. An alternative [3] is to rely on depth RGB images to learn the 3D-to-2D matching function, limiting, however, the applicability of the algorithm to RGB-D cameras only. Unlike these approaches, our focus is a search engine of the best matching 3D shapes given an object in a 2D images. Thus, we are also interested in transferability, even for unseen 3D shapes. In this work we focus on texture-free 3D shapes and standard RGB images as queries.

To alleviate the dependence on texture appearance, Choy *et al.* [5] first render textured 3D models into background-free images from a set of discretized viewpoints. Then, they compute synthesized detection templates from the extracted HOG features from these rendered views. At inference time, a detector is applied to 2D images in a sliding window fashion with multiple scales. When the detector is activated, the pose of 3D shape as well as its 2D location is transferred to the target object. Similar to [5] we also aim for localization and matching of objects in images, but rather than striving to limit the dependence on texture, we prefer to exclude texture completely.

In a recent work from the multimedia retrieval community, Junkert *et al.* [11] limit the dependency on domain shift of the generated textures of 3D shapes by relying on a neural transfer learning scheme rather than HOG descriptors. Their synthesized high quality image renderings of 3D shapes with texture, background and casted shadows. Once these synthesized images are obtained, they extract intermediate feature from an Inception [26] model pre-trained on ImageNet. Given a real image at test time, also represented by the same features, they search for the best matching 3D model by a k nearest neighbor search. As they assume all real and rendered objects have been centralized in the image, they are only able to return viewpoint angles without the location of objects. Similar to [11] we also exploit the transfer abilities of neural networks. Rather than matching the feature representations of textured 3D model renderings and the query image as is, we prefer to learn the matching function between an image and a texture-free 3D shape.

1.2 Contributions

We propose a search engine that given a query object image, localizes and matches the object to the appropriate 3D shape. Different from related approaches, we directly operate on texture-free 3D shapes, enabled by a novel learned matching function. The learned function is a shallow convolutional neural network, merged from a deep two-stream network, encapsulating the relative differences between texture-free 3D shape views and RGB objects. As we are interested in understanding the possibilities as well as the limitations of searching and matching texture-free 3D shapes in images, we rely on a controlled experimental setup on a large and diverse set of texture-free vehicle categories and sub-categories from PASCAL3D+ [29], where we evaluate its sensitivity w.r.t. available 3D shapes and object localization accuracy.

2 TOWARDS A 3D-TO-2D SEARCH ENGINE

We cast the problem of searching and matching a 3D shape to a 2D object in an image as a supervised optimization problem. In the offline phase, we assume we have a library of 3D shapes $g^{train} = \{g_1, ..., g_K\}$ describing a variety of object categories and their fine-grained sub-categories, where K is the total number of fine-grained categories. At query time, we start from a single RGB image containing an object of interest, that is either provided or detected automatically. We denote the appearance of the object with x, and in practice it can be the feature activations from one of the layers of a deep convolutional neural network. For the object of interest we assume there is an optimal, albeit hypothetical texture-free 3D model g^*_x. Given a previously unseen image x at query time, our goal is to search among the set of possible 3D shapes and their poses, g^{test}, ϕ^{test}, and place the optimal (g^*_x, ϕ^*_x) on top. To match an RGB image with a texture-free rendering, we introduce a geometric compatibility function $G(\cdot)$, which we want to maximize:

$$\phi_x = \arg\max_{\phi} G(\phi; x, x_{CAD}, g^{test}_k, \phi^{test}_{CAD}) \ , \tag{1}$$

where x_{CAD} is the rendered image given a texture-free 3D shape g^{test}_k and pose transformation ϕ_{CAD}. The $g^{test}_k, \phi^{test}_k$ can be equivalent to the shapes observed at training, or expanded to contain more 3D shapes or their poses, $g^{train}_k \subseteq g^{test}_k, \phi^{train}_k \subseteq \phi^{test}_k$. We summarize the data flow of our search engine in Fig. 2 and detail its main components next.

2.1 Searching and Matching 3D Shapes

Object Recognition and Localization In the literature on 3D to 2D matching, the recognition and localization of the object is often provided in the form of ground truth, *e.g.* [5]. The rationale

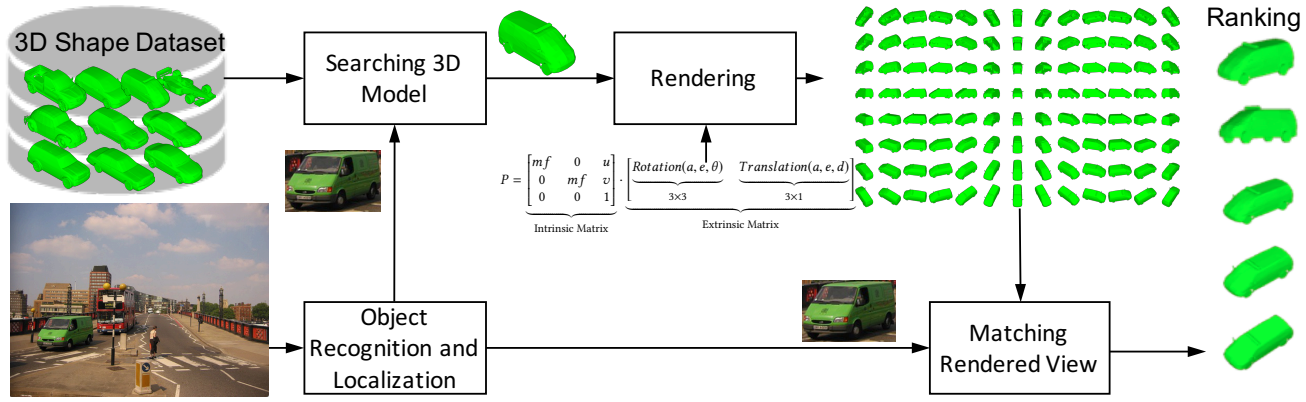

Figure 2: Dataflow for searching and matching texture-free 3D shapes in images. Our main innovation is in matching the rendered view, where we learn to match a texture-free rendering with an RGB image patch.

being that good object detectors are available [20, 22, 23]. Relevant to object recognition and localization, especially for 3D shape matching, are also the so-called amodal bounding boxes [12, 21]. Amodal boxes aim at detecting the full extent of the object, even if the bounds of the surrounding box extent the boundaries of the image. Amodal boxes capture the object centre location (u, v) and corresponding scale d, also including parts that might be not visible because of occlusion or truncation. In our experiments we quantify the sensitivity of our approach w.r.t. the quality of the (amodal) object recognition and localization using both (perturbed) groundtruth as well as a automatic object detection.

Searching 3D Model Once we have obtained the (fine-grained) object category and its enclosing box, we simply query the database of texture-free 3D shapes and forward the selected 3D shape to the rendering stage.

Rendering Each of the 3D shapes g_k can undergo various viewpoint transformations and 3D-to-2D projection. The viewpoint transformation includes rotation and translation that is controlled by extrinsic parameters, *i.e.* azimuth a, elevation e, in-plane rotation θ, camera distance d, whereas intrinsic parameters, *i.e.* principle offset (u, v), focal length f and viewport m, define the camera intrinsic matrix for 3D-to-2D projection. We aggregate the pose transformation parameters to $\phi_{CAD}^{train} = \{\phi_{CAD}^1, ..., \phi_{CAD}^M\}$, where M is the number of poses seen during training. In the following, to reduce notation clutter we drop the superscript *"train"* whenever it can be derived from the context.

Following [10], the formulation of the projection matrix takes the following form:

$$P = \underbrace{\begin{bmatrix} mf & 0 & u \\ 0 & mf & v \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Intrinsic Matrix}} \cdot \underbrace{\begin{bmatrix} \underbrace{Rotation(a,e,\theta)}_{3\times3} & \underbrace{Translation(a,e,d)}_{3\times1} \end{bmatrix}}_{\text{Extrinsic Matrix}} , \quad (1)$$

where we assume a calibrated camera, namely the focal length and viewport are fixed to $f = 1, m = 2,000$ respectively [29]. Each set of ϕ_{CAD} values produces a new *3D model rendering*. Similarly, for the hypothetical 3D model g_x^* there exists an optimal transformation vector ϕ_x^*.

Matching Rendered View Ideally, we want to avoid learning an explicit mapping from the appearance modality to the geometric modality. The reason is that an explicit mapping would function well for the 3D model rendering that are observed during training but not generalize well for new 3D models. To this end we define the geometric compatibility function as a differential on the geometric modality.

During training we have two images, x and x_{CAD}, as well as two rotation matrices, R_x and R_{CAD}, one from the real and one from the rendered image. We define our geometric compatibility function as:

$$G_{rel}^* \propto \partial\phi = g(R_x - R_{CAD}) \ . \quad (2)$$

According to eq. (2) $\partial\phi \propto g(R_x - R_{CAD})$ is a function of difference between two geometries expressed by R_x and R_{CAD}. Unfortunately, at testing time we cannot have R_x, as this is what we are looking for. To this end we propose to approximate the geometric compatibility with an approximate geometric compatibility function, which receives as inputs the appearance features of the RGB and the CAD rendering, namely:

$$G_{rel}^* \approx G_{rel} \propto \partial x = f(x - x_{CAD}) \ , \quad (3)$$

In the spirit of [14], we define the approximate geometric compatibility function $f(\cdot)$ in eq. (3) to be a separate neural network module, which we coin *geometric differential module*. A geometric differential module is implemented as a shallow convolutional neural network composed of two layers. The first layer is convolutional with kernel size 1×1 and is fusing the activations from the two streams. The second layer is a fully connected layer with a single output, which approximates $\partial\phi$. Unlike the typical fully connected layer which operates as a function approximator, the geometric differential module approximates a function differential, see Fig. 3.

The geometric differential module does not output directly any rotation matrices R_x nor any 3D model rendering ϕ_x. Instead, it returns relative geometric compatibility values. They are relative, because they are supposed to capture the difference of the R_{CAD} from R_x, and by extension the difference of ϕ_{CAD} from ϕ_x, solely

on the basis of the appearance features x and x_{CAD}. Effectively, the network learns to estimate directly the matching between an RGB image and any 3D shape rendering, avoiding to compute the object's rotation matrix first. This has two advantages.

First, the network does not directly associate the compatibility score function in eq. (3) with the appearance of the particular renderings at training time. In fact, the network does not even make any strict assumptions regarding either the texture appearance of the 3D shape rendering, or the geometric precision of the 3D model for the given RGB object. Thus, even if the 3D model is not a perfect fit to the RGB object, either because the set of viewpoint was limited, or because the geometry of the 3D model is not exactly right for the object of interest, as in Fig. 1, the network can still predict the 3D model parameters ϕ_x that match the object of interest. For instance, assume our 3D shape library has models only of a *Boeing 707* and a *Boeing 717*, while our image depicts a *Boeing 747*. Obviously, none of the two available shapes are perfect for our image. Nonetheless, our network returns the best possible fit, as the matching maximizes the matching similarity with the available models, instead of directly classifying the object appearance [2, 5, 18].

Second, since the inference returns only a compatibility score for each provided 3D model rendering, there is no limit to the type of 3D models permissible at test time. The proposed network can return a compatibility score even for 3D models that were not observed during training. As expected, the accuracy of the compatibility score in these cases might be lower, since the network matches 3D shapes and RGB images of object categories it has never seen. Still, the approximate geometric compatibility function has learned *how to match at inference time* on the basis of any 3D model rendering provided at test time. Hence, in theory, the proposed network can cope with 3D shape rendering expanded dynamically, either by adding new 3D shapes or considering a finer discretization of the viewpoints.

An important choice for the geometric differential module is what distance measure it learns to imitate. Although any geometric distance can be used, in this work we opt for approximating the geodesic distance between two rotation matrices of the real object and the rendered image respectively, R, R_{CAD}. Namely, eq. (2) becomes

$$G^*_{rel} = \frac{\| \log(R_x^T R_{CAD}) \|_F^2}{\sqrt{2}} \quad . \tag{4}$$

Thereafter, a Euclidean loss is used to measure how accurately the geodesic differential module predicts G_{rel} by relying only on the appearance features x, x_{CAD}.

2.2 Learning to Match

We implement the matching of rendered views as a two-stream architecture [24] with non-shared weights, where each stream is a convolutional network. The first stream receives as an input the cropped RGB images of the object of interest. The second stream receives as an input a cropped rendering from a particular texture-free 3D shape and essentially, describes the object's candidate geometry.

For both streams we adopt the convolutional layers 1 to 5 from AlexNet [16]. The geometric differential module is composed of a convolutional pooling layer [7] with kernel size 1×1, that fuses the two streams, followed by a fully connected module with a single output, predicting the geometric compatibility score between the RGB and the rendering streams. We initialize the weights for convolutional layers 1-5 from AlexNet, while we initialize the remaining layers with a Gaussian distribution with standard deviation 0.005. The network is trained with SGD for 70000 iterations at a learning rate of 0.0001.

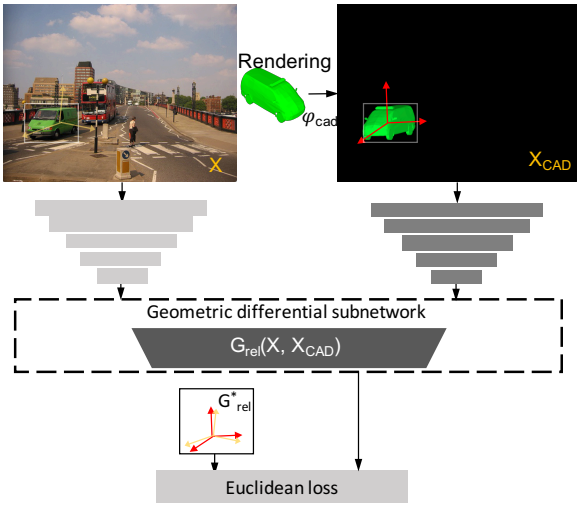

Figure 3: Matching a 3D model rendering to a 2D object in an image with geometric differentials. Starting from an image depicting a fine-grained object category and a texture-free 3D shapes, we derive a 3D model rendering and learn to match them to the localized object of interest. The network is composed of two streams. The first one processes the RGB input. The second one processes the rendered image produced by a candidate 3D model instantiation. The feature maps of the two inputs are fused together and then the geometric differential module estimates the geodesic distance between the two inputs based only on the appearances. It results in a ranking of the 3D model renderings for the localized object, ideally matching the pose of the object.

Data Preparation. The first stream receives a real image as input, while the second stream a rendered image from a 3D shape. We first create a rendering canvas with the same size as the real image. Given the selected 3D model and viewpoint annotation, we render it onto canvas. This results in a rendered image and a projected bounding box for the 3D model. We crop the real image and rendered image with the projected bounding box to obtain the input data for 2-stream network (see Fig. 3).

Training. During training we first sample a real image, for which we have the ground truth 3D model instantiation. As we want to learn to estimate the geodesic distance between different rotation matrices, we must give good estimates for both when the objects are close as well as far away, geometrically speaking. As our network is a regression model, there exist no positive or negative samples. However, to make sure that our regression model learns to approximate accurately enough across the whole spectrum of geodesic distances, we opt for a stratified sampling of the space of

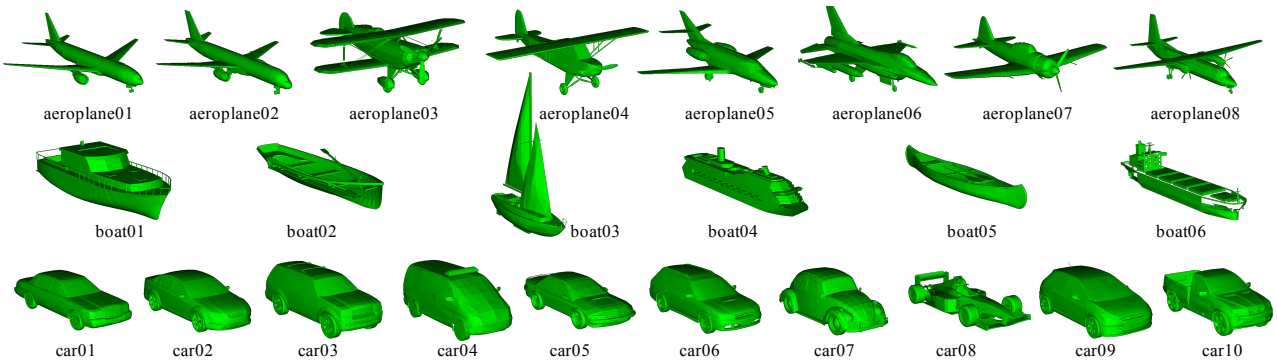

Figure 4: Texture-free 3D shapes for the 24 fine-grained *aeroplane, boat*, and *car* objects used in our experiments.

the rotation matrices to collect training examples. Specifically, per training image we sample (*i*) rotation matrices that are almost equal to the ground truth one, (*ii*) rotation matrices that are close enough but not equal, (*iii*) as well as rotation matrices that are far from the ground truth rotation matrix. In total, we end up with 30 rotation matrices per object of interest in training images. We then proceed with the training as usual, relying on SGD for backpropagation.

Inference. During inference and given one image, we traverse over possible renderings. Namely, we render all available 3D shapes in all desired viewpoints. We retain the most confident 3D model instantiation, namely, the 3D shape and viewpoint, whose rendering generates the smallest geodesic distance to the input RGB image.

3 EXPERIMENTAL SETUP

3.1 Texture-free 3D Shape Dataset

We evaluate our approach on the PASCAL3D+ dataset [29]. PASCAL3D+ extends the PASCAL VOC 2012 [6] by matching to every object location a corresponding texture-free 3D shape in its specific pose. As the 3D shapes are category- and not instance-specific and the size of the 3D shape library is finite, the matching quality varies across categories. We perform our experiments on the three vehicle categories with the most examples, namely *aeroplane*, *boat*, and *car*, for which 24 texture-free shapes exist in total, see Fig. 4. While this dataset is typically used for the problem of viewpoint estimation [25, 28], we rely on the texture-free 3D shapes for searching and matching. We follow the provided train/test splits. The statistics on the image/object data used in our experiments are summarized in Table 1.

3.2 Experiments

Experiment 1: Search and match the best rendered view given a specific 3D shape. In our first experiment, we assume the object is perfectly localized in the 2D image and the sub-category of the corresponding 3D shape is known. Thus we simply need to match the object of interest in the real image to a set of rendered views of the corresponding shape. Naturally this is an idealized setting, nonetheless it provides us with the opportunity to establish an upper-bound for our matching network to compare against in follow-up experiments.

Experiment 2: Search and match the best rendered view among a collection of 3D shapes. Although fine-grained object recognition approaches, *e.g.* [19], can be used for 3D shape selection directly, the matching module of our system should also be able to do this. In this experiment we study the performance of our system doing 3D shape search and rendered view matching at the same time. To be more specific, given a test object of interest in a 2D image, we only know the super category it belongs to (*e.g.* aeroplane), but without knowledge of whether it is an airliner or jet fighter. Hence, for each test object of interest, we have to match it with respect to all possible rendered views and all available fine-grained sub-category 3D shapes. This potentially add difficulties to our approach in distinguishing the best match especially when a few 3D shape candidates are of similar shape (*e.g.* car01, car02 and car05 in Fig. 4).

Experiment 3: Search and match the best rendered view from unseen 3D shapes Despite the discrepancy amongst 3D shapes under each category in experiment 1 and 2, all of them are seen both at training and test time. Thus, one interesting question is whether our approach is able to find and match unseen 3D shapes within one super category. Specifically, we consider using only half of the three super categories, aeroplane, boat and car, for training. At test time, we apply the trained network on an unseen sub-category to study whether our network is able to transfer the knowledge from the set of seen sub-categories. Note that, as an ablation study, we continue the setting from experiment 1, where location and scale of the 2D object is known.

Experiment 4: Search and match the best rendered view when object location and scale are imperfect.

In this experiment, we investigate how much localization noise our system can tolerate. We first study imperfect object location and scale separately. To simulate the object location error, we perturb the ground truth annotation (u, v) by adding random noise $(\Delta u, \Delta v)$ proportional to the size of ground truth amodal bounding box (width, height). We further evaluate inaccurate detection of object scale in terms of the camera distance d, by randomly scaling it down/up to a maximum of {80%, 90%, 100%, 110%, 120%} of the original value. This results in the 3D shape being rendered smaller/bigger than the object in the real image. Note that scaling down camera distance d means bringing an object closer to the camera and thus a larger amodal bounding box on the 2D space.

In the third sub-experiment, we perturb (u, v) and d at the same time, keeping the noise of $(\Delta u, \Delta v)$ at a level of 10% , while randomly varying the scale d to {90%, 100%, 110%} of the original. Note that the fine-grained shape is given in this experiment for a better understanding of the effect of noise on our system.

Experiment 5: Search engine comparison. In our last experiment, we compare our system with Choy *et al.* [5]. The comparisons are conducted with two settings: (1) search and match given the ground truth object location and (2) fully automatic search and match. As an object detector returns the bounding box coordinates, but not the full object extent in terms of principle point u, v and distance d, we simply choose the center of the bounding box provided by [20] and a scale value d that best matches the bounding box. In contrast, [5] relies on the detected bounding box and a local search is conducted with a multi-scaled detector template. For a fair comparison with [5], we run their software on our rendered images from the 24 texture-free 3D shapes and train both methods in the same way.

3.3 Evaluation Criteria

To test the searching and matching accuracy of our network, we first discretize the rotation parameter space of azimuth a, elevation e and in-plane rotation θ uniformly into 21, 11, 11 bins respectively. This results in 2,541 rendering views in total. At test time we report results when considering the ground truth rendering as well, resulting in 2,542 rendering views and thus 2,542 predictions of the matching score. We report the *precision*@5, namely a prediction is correct if it contains the ground truth 3D shape in the top-5 ranked positions. On top of the alignment precision at 5 we also measure the *amodal intersection over union (AIOU)*, see Fig. 5, to quantify the shape disagreement given a 2D object of interest and a 3D shape. To compute the AIOU we measure the intersection over union overlap between the 2D ground truth bounding box and the amodal bounding box[12] derived by the optimal 3D shape from the annotators. A perfectly fitting 3D shape will have very high IOU with the 2D box, while a poor fitting 3D shape will return a very different amodal bounding box and thus low AIOU. We also report the mean over all test queries, indicated by Query Mean.

For the comparison with Choy *et al.* [5], we follow their setup and report accuracy at θ ($ACC@\theta$) when the ground truth bounding box is given. This metric evaluates the fraction of viewpoint predictions that are within a fixed threshold (θ) of its ground truth. When incorporating automatic objection detection, we report Average Viewpoint Precision (AVP) for evaluation. It is similar to Average Precision (AP) in object detection, but it only counts detections as correct when the bounding box overlap ratio exceeds 0.5 and when the prediction of the discretized azimuth angle is in the right bin. We report performance on different levels of azimuth angle discretization, {4,8,16,24} bins. As discretization goes finer, the more difficult this task becomes.

4 RESULTS

4.1 Search and match specific 3D shape

We present the matching accuracy and AIOU results in Table 1. Cars are matched the best, as cars generally have a simple box-like shape. In contrast, boats are more challenging . For one, boats exhibit large variation in shape and object size/scale: a boat includes sub-categories from tanker to canoes, see Fig. 4. Moreover, for boats occlusion exists naturally since half of the boat is almost always underwater, thus confusing the matching network that expects the object to be fully visible. When excluding the ground truth pose from the rendered view search space, the *precision*@5 drops from 0.64 to 0.48 for aeroplanes, from 0.44 to 0.27 for boats and from 0.75 to 0.68 for cars. In this scenario, a poor-fitted amodal bounding box hurts even more. To decouple the matching from the rendered view search strategy, we include the ground truth rendering in the rendering search space in the remaining experiments.

When considering individual sub-categories we observe that apart from having a sufficient number of examples available for training, the consistency in shape appearance is important. For example, boat03 is relatively hard. This is due to the fact that sailing boats can have large shape variance and the sail may have different orientation from its body. In contrast, boat04 and boat06 are relatively easy because both of them are big ships resembling a 3D box floating on the water, which facilitates the matching. Similar observations hold for airplanes. Cars generally have more consistent accuracy. Interesting cases are car07 and car04, with car07 being better matched than car04 despite having fewer training samples. The reason is the frontal and rear view of car07 are quite distinguishable, whereas car04 looks like a box, with front and rear poses being often confused.

4.2 Search and match among 3D shapes

We show results in Table. 2. When the fine-grained sub-category of the object is unknown the network must iterate over all possible 3D shapes. This amounts to an N-fold increase of possible 3D shape instantiations, where N is the number of possible sub-categories. Despite this N-fold increase, we observe that the matching network accuracy drops only 2-fold, approximately. As a reference, a random baseline is about $2 \cdot 10^{-4}$ for aeroplanes, where our matching network scores 0.32. Loosely inspired by the work of Junkert *et al.* [11], intended for textured renderings, we also report a baseline based on a nearest neighbor search strategy. Specifically, we

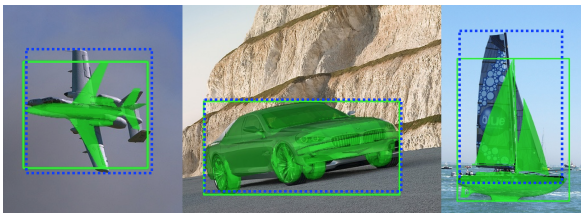

Figure 5: 2D annotation bounding box (dashed blue) vs amodal bounding box (solid green). The difference of the two bounding boxes indicates the shape disagreement between a 3D model to the 2D object in image. For the car the two boxes align almost perfectly, while for the sailing boat there is a noticeable discrepancy. In general, in the PASCAL3D+ dataset, the 3D shapes of cars have the best agreement with the 2D object in an image, while boats have the worst. We show statistics of the amodal intersection over union (AIOU) in Table 1.

Table 1: Experiment 1: Search and match the best rendered view given a specific 3D shape. Note the correlation between precision at 5 and quality of the amodal bounding box per category measured by AIOU.

aeroplane					boat					car				
Subtype	#Train	#Queries	AIOU	p@5	Subtype	#Train	#Queries	AIOU	p@5	Subtype	#Train	#Queries	AIOU	p@5
aeroplane01	761	128	0.80	0.73	boat01	572	123	0.60	0.51	car01	696	97	0.84	0.78
aeroplane02	105	0	0.81	-	boat02	363	24	0.73	0.25	car02	729	37	0.87	0.73
aeroplane03	80	11	0.75	0.36	boat03	519	46	0.68	0.22	car03	932	35	0.87	0.63
aeroplane04	82	26	0.75	0.50	boat04	530	16	0.78	0.62	car04	141	24	0.82	0.54
aeroplane05	88	45	0.76	0.64	boat05	40	11	0.66	0.55	car05	139	17	0.82	0.76
aeroplane06	478	44	0.76	0.50	boat06	492	12	0.69	0.58	car06	1261	18	0.85	0.83
aeroplane07	392	21	0.73	0.62						car07	137	5	0.79	1.00
aeroplane08	88	0	0.80	-						car08	107	8	0.83	0.75
										car09	884	42	0.82	0.98
										car10	641	25	0.86	0.56
Query Mean			0.78	0.64	Query Mean			0.69	0.44	Query Mean			0.85	0.75

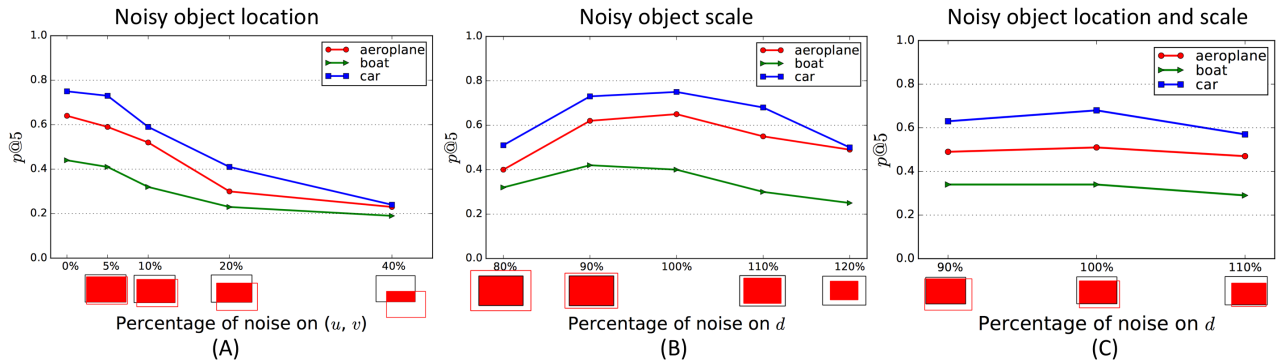

Figure 6: Experiment 4: Search and match the best rendered view when object location (u, v) and d scale are imperfect. On the x axis we visualize the indicative overlap displacement with respect to the perfect box. For moderate noise (5-10%) the model recovers a good match.

extract fc7 features of a pre-trained AlexNet [16] from the object of interest in both the real and rendered images with $K = 2,542$ different viewpoints, and rank them based on cosine similarity. The results show this approach does not work well in our setting, with p@5 around 0.07, as the domain shift from real images to rendered images without texture and background context is simply too large.

4.3 Search and match unseen 3D shapes

We report the performance of our network trained on both seen and unseen sub-categories in Table 3. The network is able to transfer its matching knowledge to unseen shapes to some extent. Note that this experimental setting is quite close to zero-shot image classification [15] where one classifies a category without having seen its visual examples, a challenging task where accuracy drops are generally high. Focusing on sub-categories, aeroplane05 is matched best, while aeroplane06 is affected more. Likely because aeroplane05 looks similar to both aeroplane01 and aeroplane02, while aeroplane06 (jet fighter) is more different. Similarly, boat05 is well matched because it resembles boat02. However, boat04 and boat06 refer to big ships, and transferring the matching knowledge from the considerably smaller boats (yachts, canoes and sailing boats) is harder. For cars, where all sub-categories are quite similar

Table 2: Experiment 2: Search and match the best rendered view among a collection of 3D shapes. Per super-category, we go over all N 3D shapes per sub-category, namely $N \cdot 2,542$ 3D shape hypotheses. Matching is feasible even when the sub-categories are unknown (compare with Table 1).

aeroplane		boat		car	
Subtype	p@5	Subtype	p@5	Subtype	p@5
aeroplane01	0.35	boat01	0.15	car01	0.46
aeroplane02	-	boat02	0.00	car02	0.27
aeroplane03	0.27	boat03	0.22	car03	0.23
aeroplane04	0.31	boat04	0.44	car04	0.08
aeroplane05	0.24	boat05	0.09	car05	0.35
aeroplane06	0.34	boat06	0.42	car06	0.28
aeroplane07	0.33			car07	0.40
aeroplane08	-			car08	0.25
				car09	0.69
				car10	0.16
Query Mean	0.32	Query Mean	0.18	Query Mean	0.37

we observe a good matching accuracy. We conclude that if new 3D

Table 3: Experiment 3: Search and match the best rendered view from unseen 3D shapes. Intra-category knowledge transfer can be performed when new 3D shapes added at test time are somewhat similar to the ones seen during training (marked in gray).

aeroplane		boat		car	
Subtype	p@5	Subtype	p@5	Subtype	p@5
aeroplane01	0.67	boat01	0.43	car01	0.85
aeroplane02	-	boat02	0.29	car02	0.84
aeroplane03	0.45	boat03	0.37	car03	0.57
aeroplane04	0.62			car04	0.62
				car05	0.71
aeroplane05	0.40	boat04	0.06	car06	0.61
aeroplane06	0.14	boat05	0.55	car07	1.00
aeroplane07	0.33	boat06	0.00	car08	0.25
aeroplane08	-			car09	0.88
				car10	0.44

shapes added at test time are somewhat similar to the seen ones, our network can match them reasonably well.

4.4 Search and match under noisy conditions

Noisy object location. We first look at the effect of noise on the object location (u, v), see Fig. 6 (A). For a limited amount of 5% noise the impact on the network is modest, when averaged over categories the loss is 0.59 for aeroplane, 0.41 for boat and 0.73 for car. When we add more noise performance starts to suffer, with 10% noise the numbers drop to 0.52, 0.32 and 0.59 respectively. Unsurprisingly the performance drops as more and more noise is added, be it that the drop is less pronounced after 20%.

Noisy object scale. Next, we investigate the sensitivity when adding noise to the camera distance d, which relates to the detected object scale (smaller $d \rightarrow$ larger object scale), see Fig. 6 (B). As before, the larger the deviation the larger the drop in performance and any noise in the range ±10% leads to an acceptable matching accuracy. Interestingly, when scaling d down to 90% it leads to a slight increase in performance for boats. The reason is that scaling down the camera distance d results in a bigger amodal bounding box that often is a better fit to the actual object.

Noisy object location and scale. Last, we assess the impact of having noise in both the location and the scale. Results are shown in Fig. 6 (C). We observe the accuracy remains stable for all categories, indicating the two different types of noise do not reinforce each other too much.

4.5 Search engine comparison

Results for given ground truth object location are shown in Table 4. In terms of the mean over all queries, we obtain better $ACC@\theta$ than [5] for all 3 categories. The boat category has relatively low performance for both methods, because of the low AIOU rate (see Table 1) that indicates large shape discrepancies. Looking into the sub-category comparisons, Choy *et al.* is better in some cases (e.g. aeroplane03, aeroplane04, boat02-boat06), mostly when less training examples are provided which results in our two-stream network being underfitting. Table 5 reports results of joint object detection and 3D-to-2D matching. In terms of AVP we outperform Choy *et al.* also in this setting. As expected, performance for both search engines suffers when azimuth angle discretization becomes finer. While aeroplanes and cars are again matched better, now aeroplane are easier to match than cars, presumably due to their easier detection of typically simpler backgrounds. We conclude that our system with a learned matching function is beneficial over hand-engineered feature matching approaches when sufficient (>80) training samples are available. At the same time there is still a lot of work needed before we arrive at generic and precise searching and matching of 3D shapes in images.

Table 4: Experiment 5: Search engine comparison with Choy *et al.* [5] using ground truth object location, where we run the software of Choy *et al.* on our texture-free setting. In terms of $ACC@\theta$ our approach is especially beneficial for 3D shapes that have sufficient training samples, where Choy *et al.* profit from limited example regimes (see Table 1). For boats both approaches perform modestly.

aeroplane			boat			car		
Subtype	Choy *et al.*	This paper	Subtype	Choy *et al.*	This paper	Subtype	Choy *et al.*	This paper
aero01	0.48	0.59	boat01	0.32	0.36	car01	0.39	0.62
aero02	-	-	boat02	0.48	0.21	car02	0.62	0.60
aero03	0.55	0.46	boat03	0.33	0.26	car03	0.54	0.57
aero04	0.39	0.31	boat04	0.19	0.19	car04	0.42	0.50
aero05	0.40	0.56	boat05	0.50	0.09	car05	0.35	0.71
aero06	0.39	0.57	boat06	0.42	0.25	car06	0.83	0.61
aero07	0.33	0.52				car07	0.40	0.80
aero08	-	-				car08	0.63	0.50
						car09	0.76	0.74
						car10	0.60	0.48
Query Mean	0.35	0.54	Query Mean	0.22	0.29	Query Mean	0.29	0.61

Table 5: Experiment 5: Search engine comparison with Choy *et al.* [5] with automatically detected object location, using their public software on our texture-free setting. The proposed system achieves better performance in terms of AVP for aeroplane and car and worse for boat.

	aeroplane				boat				car			
#Bins	4	8	16	24	4	8	16	24	4	8	16	24
Choy *et al.*	0.35	0.22	0.10	0.05	0.14	0.06	0.02	0.01	0.23	0.17	0.10	0.07
This paper	0.44	0.24	0.10	0.05	0.07	0.04	0.01	0.01	0.30	0.26	0.19	0.11

5 CONCLUSION

This paper focuses on searching and matching the best rendered view of a texture-free 3D shape to an object of interest in a 2D image. Matching rendered views of 3D shapes to RGB images is challenging because of imperfect 3D shapes and domain shift in appearance due to texture mismatch. We propose a deeply learned matching function that attacks these challenges and can be used as a search engine of 3D shapes to objects in 2D. We evaluate the proposed search engine on the most populated PASCAL3D+ vehicle categories, testing the capabilities of transferring the learnt function to unseen 3D shapes and its sensitivity to imperfect 3D shapes and localization. We also identify the need for accurate amodal bounding box detection in 2D images as an important 3D-to-2D matching topic for further investigation.

REFERENCES

[1] Jurgen Assfalg, Alberto Del Bimbo, and Pietro Pala. 2004. Retrieval of 3D Objects by Visual Similarity. In *MIR*.

[2] Mathieu Aubry, Daniel Maturana, Alexei Efros, Bryan Russell, and Josef Sivic. 2014. Seeing 3D chairs: exemplar part-based 2D-3D alignment using a large dataset of CAD models. In *CVPR*.

[3] Aayush Bansal, Bryan Russell, and Abhinav Gupta. 2016. Marr Revisited: 2D-3D Alignment via Surface Normal Prediction. In *CVPR*.

[4] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015. ShapeNet: An Information-Rich 3D Model Repository. *CoRR* (2015).

[5] Christopher Bongsoo Choy, Michael Stark, Sam Corbett-Davies, and Silvio Savarese. 2015. Enriching Object Detection with 2D-3D Registration and Continuous Viewpoint Estimation. In *CVPR*.

[6] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2015. The pascal visual object classes challenge: A retrospective. *IJCV* 111, 1 (2015), 98–136.

[7] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. 2016. Convolutional two-stream network fusion for video action recognition. In *CVPR*.

[8] Thomas Funkhouser, Patrick Min, Michael Kazhdan, Joyce Chen, Alex Halderman, David Dobkin, and David Jacobs. 2003. A Search Engine for 3D Models. *ACM Trans. Graph.* 22, 1 (2003), 83–105.

[9] Saurabh Gupta, Pablo Arbelaez, Ross Girshick, and Jitendra Malik. 2015. Aligning 3D models to RGB-D images of cluttered scenes. In *CVPR*.

[10] Richard Hartley and Andrew Zisserman. 2003. *Multiple View Geometry in Computer Vision* (2 ed.). Cambridge University Press, New York, NY, USA.

[11] Fabian Junkert, Markus Eberts, Adrian Ulges, and Ulrich Schwanecke. 2017. Cross-modal Image-Graphics Retrieval by Neural Transfer Learning. In *ICMR*. 330–337.

[12] Abhishek Kar, Shubham Tulsiani, Joao Carreira, and Jitendra Malik. 2015. Amodal completion and size constancy in natural scenes. In *ICCV*.

[13] Toshikazu Kato, Takio Kurita, Nobuyuki Otsu, and Kyoji Hirata. 1992. A sketch retrieval method for full color image database-query by visual example. In *ICPR*. IEEE, 530–533.

[14] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *ICLR*.

[15] Svetlana Kordumova, Thomas Mensink, and Cees G. M. Snoek. 2016. Pooling Objects for Recognizing Scenes without Examples. In *ICMR*.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*.

[17] Michael S Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. 2006. Content-based multimedia information retrieval: State of the art and challenges. *ACM TOMCCAP* 2, 1 (2006), 1–19.

[18] Joseph J Lim, Aditya Khosla, and Antonio Torralba. 2014. FPM: Fine pose parts-based model with 3d cad models. In *ECCV*.

[19] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. 2015. Bilinear cnn models for fine-grained visual recognition. In *ICCV*.

[20] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. Ssd: Single shot multibox detector. In *ECCV*.

[21] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 2016. 3D Bounding Box Estimation Using Deep Learning and Geometry. *arXiv preprint arXiv:1612.00496* (2016).

[22] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *CVPR*.

[23] Shaoqing Ren, Kaiming He, and Ross Girshick. 2015. Faster R-CNN_ Towards Real-Time Object Detection with Region Proposal Networks. In *NIPS*.

[24] Karen Simonyan and Andrew Zisserman. 2014. Two-stream Convolutional Networks for Action Recognition in Videos. In *NIPS*.

[25] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. 2015. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *ICCV*.

[26] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. 2015. Going deeper with convolutions. In *CVPR*.

[27] Johan W. H. Tangelder and Remco C. Veltkamp. 2007. A survey of content based 3D shape retrieval methods. *MTAP* 39, 3 (2007), 441.

[28] Shubham Tulsiani and Jitendra Malik. 2015. Viewpoints and Keypoints. In *CVPR*.

[29] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. 2014. Beyond PASCAL: A Benchmark for 3D Object Detection in the Wild. In *WACV*.