

# How Useful Is Photo-Realistic Rendering for Visual Learning?

Yair Movshovitz-Attias<sup>1</sup>(✉), Takeo Kanade<sup>2</sup>, and Yaser Sheikh<sup>2</sup>

<sup>1</sup> Computer Science Department, Carnegie Mellon University, Pittsburgh, USA  
yair@cs.cmu.edu

<sup>2</sup> The Robotics Institute, Carnegie Mellon University, Pittsburgh, USA  
{Takeo.Kanade,yaser}@cs.cmu.edu

**Abstract.** Data seems cheap to get, and in many ways it is, but the process of creating a high quality labeled dataset from a mass of data is time-consuming and expensive.

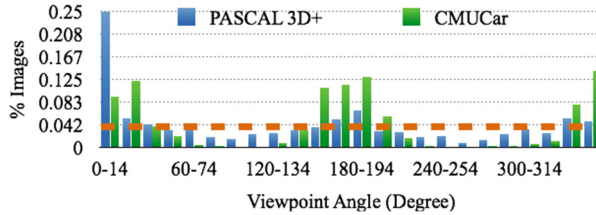
With the advent of rich 3D repositories, photo-realistic rendering systems offer the opportunity to provide nearly limitless data. Yet, their primary value for visual learning may be the quality of the data they can provide rather than the quantity. Rendering engines offer the promise of perfect labels in addition to the data: what the precise camera pose is; what the precise lighting location, temperature, and distribution is; what the geometry of the object is.

In this work we focus on semi-automating dataset creation through use of synthetic data and apply this method to an important task – object viewpoint estimation. Using state-of-the-art rendering software we generate a large labeled dataset of cars rendered densely in viewpoint space. We investigate the effect of rendering parameters on estimation performance and show realism is important. We show that generalizing from synthetic data is not harder than the domain adaptation required between two real-image datasets and that combining synthetic images with a small amount of real data improves estimation accuracy.

## 1 Introduction

The computer vision community has been building datasets for decades, and as long as we have been building them, we have been fighting their biases. From the early days of COIL-100 [16] the Corel Stock Photos and 15 Scenes datasets [17] up to and including newer datasets such as PASCAL VOC [5] and Imagenet [4], we have experienced bias: every sample of the world is biased in some way – viewpoint, lighting, etc. Our task has been to build algorithms that perform well on these datasets. In effect, we have “hacked” each new dataset - exploring it, identifying weaknesses, and in sometimes flawlessly fitting to it.

For an in depth analysis of the evolution of datasets (and an enjoyable read) we refer the reader to [24]. In short, there are two main ways in which our community has addressed bias: making new datasets, and building bigger ones. By making new datasets we continuously get new samples of the visual world, and



**Fig. 1.** Photographers tend to capture objects in *canonical* viewpoints. When real images are used as training data these viewpoints are oversampled. Here we show viewpoint distributions for two real image datasets. Note the oversampling of certain views, e.g.  $0^\circ$ ,  $180^\circ$ . In comparison, an advantage of a synthetic dataset is that it is created to specification. A natural distribution to create is uniform (dashed line).

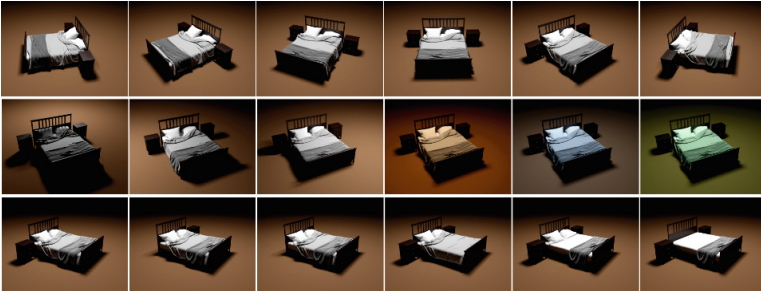
make sure our techniques handle more of its variability. By making our datasets larger we make it harder to over-fit to the dataset’s individual idiosyncrasies.

This approach has been remarkably successful. It requires, however, a great amount of effort to generate new datasets and label them with ground truth annotations. Even when great care has been taken to minimize the sampling bias it has a way of creeping in. As an example, for the task of object viewpoint estimation, we can observe clear bias in the distribution viewpoint angles when exploring real image datasets. Figure 1 shows the distribution of azimuth angles for the training sets of the car class of PASCAL VOC, and the CMUCar dataset. There is clear oversampling of some angles, mainly around  $0^\circ$  and  $180^\circ$ .

In this work we explore the benefits of synthetically generated data for viewpoint estimation. 3D viewpoint estimation is an ideal task for the use of renders, as it requires high of accuracy in labeling. We utilize a large database of accurate, highly detailed, 3D models to create a large number of synthetic images. To diversify the generated data we vary many of the rendering parameters. We use the generated dataset to train a deep convolutional network using a loss function that is optimized for viewpoint estimation. Our experiments show that models trained on rendered data are as accurate as those trained on real images. We further show that synthetic data can be also be used successfully during validation, opening up opportunities for large scale evaluation of trained models.

With rendered data, we control for viewpoint bias, and can create a uniform distribution. We can also adequately sample lighting conditions and occlusions by other objects. Figure 2 shows renders created for a single object, an IKEA bed. Note how we can sample the different angles, lighting conditions, and occlusions. We will, of course, have other types of bias, and this a combined approach – augment real image datasets with rendered data. We explore this idea below.

We assert that a factor limiting the success of many computer vision algorithms is the *scarcity* of labeled data and the *precision* of the provided labels. For viewpoint estimation in particular the space of possible angles is immense, and collecting enough samples of every angle is hard. Furthermore, accurately labeling each image with the ground truth angle proves difficult for human



**Fig. 2.** Generating synthetic data allows us to control image properties, for example, by: sampling viewpoint angles (top row), sampling lighting conditions (middle), and sampling occlusions by controlling the placement of night stands and linens (bottom).

annotators. As a result, most current 3D viewpoint datasets resort to one of two methods for labeling data: (1) Provide coarse viewpoint information in the form of viewpoint classes (usually 8 to 16). (2) Use a two step process of labeling. First, ask annotators to locate about a dozen keypoints on the object (e.g., front-left most point on a car’s bumper), then manually locate those same points in 3D space on a preselected model. Finally, perform PnP optimization [13] to learn a projection matrix from 3D points to 2D image coordinates, from which angle labels are calculated. Both methods are unsatisfying. For many downstream applications a coarse pose classification is not enough, and the complex point correspondence based process expensive to crowdsource. By generating synthetic images one can create large scale datasets, with desired label granularity level.

## 2 Related Work

The price of computational power and storage has decreased dramatically over the last decade. This decrease ushered in a new era in computer vision, one that makes use of highly distributed inference methods [3, 23] and massive amounts of labeled data [7, 15, 19]. This shift was accompanied by a need for efficient ways to quickly and accurately label these large datasets. Older and smaller datasets were normally collected and labeled by researchers themselves. This ensured high quality labels but was not scalable. As computer vision entered the age of “Big Data” researchers began looking for better ways to annotate large datasets.

Online labor markets such as Amazon’s Mechanical Turk have been used in the computer vision community to crowdsource simple tasks such as image level labeling, or bounding box annotation [12, 19, 26]. However, labor markets often lack expert knowledge, making some classes of tasks impossible to complete. Experts are rare in a population and when are not properly identified, their answers will be ignored when not consistent with other workers – exactly on the instances where their knowledge is most crucial [10].

The issues detailed above make for a compelling argument for automating the collection and labeling of datasets. The large increase in availability of 3D

CAD models, and the drop in the costs of obtaining them present an appealing avenue of exploration: Rendered images can be tailored for many computer vision applications. Sun and Saenko [22] used rendered images as a basis for creating object detectors, followed by an adaption approach based on decorrelated features. Stark et al. [20] used 3D models of cars as a source for labeled data. They learned spatial part layouts which were used for detection. However, their approach requires manual labeling of semantic part locations and it is not clear how easy this can scale up to the large number of objects now addresses in most systems. A set of highly detailed renders was used to train ensembles of exemplar detectors for vehicle viewpoint estimation in [14]. Their approach required no manual labeling but the joint discriminative optimization of the ensembles has a large computational footprint and will be hard to scale as well. Pepik et al. [18] showed that deep networks are not invariant to certain appearance changes, and use rendered data to augment the training data, and in [9, 25] rendered data is used to train pedestrian detectors.

Here, we show that detailed renders from a large set of high quality 3D models can be a key part of scaling up labeled data set curation. This was unfeasible just 10 years ago due to computational costs, but a single GPU today has three orders of magnitude more compute power than the server farm used by Pixar for their 1995 movie Toy Story [6]. The time is ripe for re-examining synthetic image generation as a tool for computer vision. Perhaps most similar to our work is [21] in which rendered images from a set of 3D models were employed for the task of viewpoint estimation. However, while they focus on creating an end-to-end system, our goal is to systematically examine the benefits of rendered data.










### 3 Data Generation Process

To highlight the benefits of synthetic data we opt to focus on the car object class. We use a database of 91 highly detailed 3D CAD models obtained from [doschdesign.com](http://doschdesign.com) and [turbosquid.com](http://turbosquid.com).

For each model we perform the following procedure. We define a sphere of radius  $R$  centered at the model. We create virtual cameras evenly spaced on the sphere in one degree increments over rings at 5 elevations:  $-5^\circ, 0^\circ, 10^\circ, 20^\circ, 30^\circ$ . Each camera is used to create a render of one viewpoint of the object. We explore the following rendering parameters:

**Lighting Position.** We uniformly sample the location of a directed light on a sphere, with elevation in  $[10^\circ, 80^\circ]$ .

**Light Intensity.** We uniformly sample the Luminous power (total emitted visible light power measured in lumens) between 1,400 and 10,000. A typical 100 W incandescent light bulb emits about 1500 lumens of light, and normal day light is between 5,000 and 10,000 lumens. The amount of power needed by the light source also depends on the size of the object, and the distance of the light source from it. As models were built at varying scales, the sampling of this parameter might require some adjustments between models.

| Light Type  | Candle  | 40W Tungsten  | 100W Tungsten   | Halogen   | Carbon  | Noon Sun  | Direct Sunlight   | Overcast Sky  | Clear Blue Sky  |
|-------------|---|---|---|---|---|---|---|---|---|
| Temperature | 1900  | 2600  | 2850  | 3200  | 5200  | 5400  | 6000  | 7000  | 20000   |
| Color       |  |  |  |  |  |  |  |  |  |

**Fig. 3.** Set of light source temperatures used in rendering process. We create each render with a randomly selected temperature.

**Light Temperature.** We randomly pick one of  $K=9$  light temperature profiles. Each profile is designed to mimic a real world light scenario, such as midday sun, tungsten light bulb, overcast sky, halogen light, etc. (see Fig. 3).

**Camera F-stop.** We sample the camera aperture F-stop uniformly between 2.7 and 8.3. This parameter controls both the amount of light entering the camera, and the depth of field in which the camera retains focus.

**Camera Shutter Speed.** We uniformly sample shutter speeds between  $1/25$  and  $1/200$  of a second. This controls the amount of light entering the camera.

**Lens Vignetting.** This parameter simulates the optical vignetting effect of real world cameras. Vignetting is a reduction of image brightness and saturation at the periphery compared to the image center. For 25% of the images we add vignetting with a constant radius.

**Background.** Renders are layered with a natural image background patches that are randomly selected from PASCAL training images not from the “Car” class.

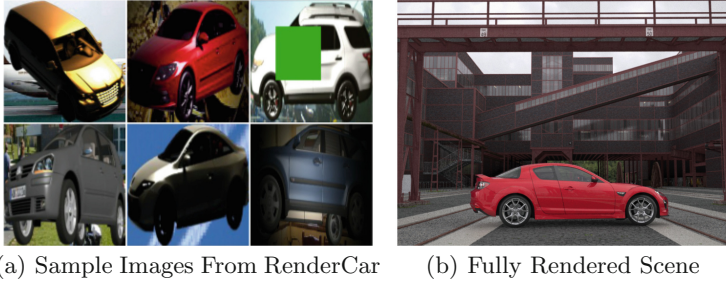
For rendering the images we use 3DS MAX, with the production quality VRAY rendering plug-in [8]. There has been considerable evidence that data augmentation methods can contribute to the accuracy of trained deep networks [27]. Therefore, we augment our rendered images by creating new images, as follows:

**Compression Effects.** Most of the images that the trained classifier will get to observe during test time are JPEG compressed images. Our renders are much cleaner, and are saved in lossless PNG format. In most cases JPEG compression is not destructive enough to be visually noticeable, but it was shown that it can influence classifier performance. We therefore JPEG compress all renders.

**Color Cast.** For each channel of the image, with probability  $\frac{1}{2}$  we add a value sampled uniformly in  $[-20, 20]$ .

**Channel Swap.** With probability 50% we randomly swap the color channels.

**Image Degradation.** Some ground truth bounding boxes are very small. The resulting object resolution is very different than our high resolution renders. In order for the model to learn to classify correctly lower resolution images, we estimate the bounding box area using the PASCAL training set, and 5 down-sample 25% of the renders to a size that falls in the lower 30% of the distribution.



**Fig. 4.** (a) Rendered training images with data augmentation. (b) To evaluate the use of renderers as test data we create scenes in which an object is placed in a fully modeled environment. This is challenging for models trained on natural images (Table 1).

**Occlusions.** To get robustness to occlusions we randomly place rectangular patches either from the PASCAL training set, or of uniform color, on the renders. The size of the rectangle is sampled between 0.2 and 0.6 of the render size.

Finally, we perform a train/test split of the data such that images from 90 models are used for training, and one model is held out for testing. The resulting datasets have 819,000 and 1,800 images respectively. Figure 4(a) shows a number of rendered images after application of the data augmentation methods listed above. We name this dataset RenderCar.

With the steady increase in computational power, the lowered cost of rendering software, and the availability of 3D CAD models, for the first time it is now becoming possible to fully model not just the object of interest, but also its surrounding environment. Figure 4(b) shows a fully rendered images from one such scene. Rendering such a fully realistic image takes considerably more time

**Table 1.** Median angular error of car viewpoint estimation on ground truth bounding boxes. Note the distinct effect of dataset bias: the best model on each dataset is the one trained on the corresponding training set. On average, the model trained on rendered images performs similarly to that trained on PASCAL, and better than one that is trained on CMUCar. Combining rendered data with natural images produces lower error than when combining two natural-image datasets. Combining all three datasets provides lowest error. Last column shows average error on columns 1, 2, 5.

| Training      | Validation |           |           |                   |           |             |                |
|---------------|------------|-----------|-----------|-------------------|-----------|-------------|----------------|
|               | PASCAL     | CMUCar    | RenderCar | Render full scene | P+C       | Avg         | Avg on natural |
| PASCAL (P)    | 16°        | 6°        | 18°       | 14°               | 8°        | 12.4°       | 10°            |
| CMUCar (C)    | 29.5°      | <b>2°</b> | 27°       | 13°               | 5°        | 15.3°       | 12.17°         |
| RenderCar (R) | 18°        | 6°        | 2°        | 8°                | 8°        | 8.4°        | 10.67°         |
| P+C           | 15°        | 3°        | 13°       | 9°                | 5°        | 9°          | 7.67°          |
| P+R           | <b>11°</b> | 6°        | 4°        | 6°                | 6°        | 6.6°        | 7.67°          |
| C+R           | 15°        | <b>2°</b> | 2°        | <b>5°</b>         | 4°        | 5.6°        | 7°             |
| P+C+R         | 12°        | <b>2°</b> | <b>1°</b> | 8°                | <b>3°</b> | <b>5.2°</b> | <b>5.67°</b>   |

and computational resources than just the model. Note the interaction between the model and the scene – shadows, reflections, etc. While we can not, at the moment, produce a large enough dataset of such renders to be used for training purposes, we can utilize a smaller set of rendered scene images for *validation*. We create a dataset of fully rendered scenes which we term RenderScene. In Table 1 we show that such a set is useful for evaluating models trained on real images.

## 4 Network Architecture and Loss Function

Following the success of deep learning based approaches in object detection and classification, We base our network architecture on the widely used AlexNet model [11] with a number of modifications. The most important of those changes is our introduced loss function. Most previous work on viewpoint estimation task have simply reduced it to a regular classification problem. The continuous viewpoint space is discretized into a set of class labels, and a classification loss, most commonly SoftMax, is used to train the network. This approach has a number of appealing properties: (1) SoftMax is a well understood loss and one that has been used successfully in many computer vision tasks. (2) The predictions of a SoftMax are associated with probability values that indicate the model’s confidence in the prediction. (3) The classification task is easier than a full regression to a real value angle which can reduce over-fitting.

There is one glaring problem with this reduction - it does not take into account the circular nature of angle values. The discretized angles are just treated as class labels, and any mistake the model makes is penalized in the same way. There is much information that is lost if the distance between the predicted angle and the ground truth is not taken into account when computing the error gradients. We use a generalization of the SoftMax loss function that allows us to take into account distance-on-a-ring between the angle class labels:

$$E = -\frac{1}{N} \sum_{n=1}^N \sum_{k=1}^K w_{l_n, k} \log(p_{n, k}), \quad (1)$$

where  $N$  is the number of images,  $K$  the number of classes,  $l_n$  the ground truth label of instance  $n$ , and  $p_{n, k}$  the probability of the class  $k$  in example  $n$ .  $w_{l_n, k}$  is a Von Mises kernel centered at  $l_n$ , with the width of the kernel controlled by  $\sigma$ :

$$w_{l_n, k} = \exp\left(-\frac{\min(|l_n - k|, K - |l_n - k|)}{\sigma^2}\right). \quad (2)$$

The Von Mises kernel implements a circular normal distribution centered at  $0^\circ$ . This formulation penalizes predictions that are far, in angle space, more than smaller mistakes. By acknowledging the different types of mistakes, more information flows back through the gradients. An intuitive way to understand this loss, is as a matrix of class-to-class weights where the values indicate the weights  $w$ . A standard Softmax loss would have a weight of 1 on the diagonal of the matrix, and 0 elsewhere. In the angle-aware version, some weight is given to nearby classes, and there is a wrap-around such that weight is also given when the distance between the predicted and true class crosses the  $0^\circ$  boundary.



## 5 Evaluation

Our objective is to evaluate the usefulness of rendered data for training. First, we compare the result of the deep network architecture described in Sect. 4 on two fine-grained viewpoint estimation datasets:

**CMU-Car.** The MIT street scene data set [1] was augmented by Boddeti et al. [2] with landmark annotations for 3,433 cars. To allow for evaluation of precise viewpoint estimation Movshovitz-Attias et al. [14] further augmented this data set by providing camera matrices for 3,240 cars. They manually annotated a 3D CAD car model with the same landmark locations as the images and used the POSIT algorithm to align the model to the images.

**PASCAL3D+.** The dataset built by [28] augments 12 rigid categories of the PASCAL VOC 2012 with 3D annotations. Similar to above, the annotations were collected by manually matching correspondence points from CAD models to images. On top of those, more images are added for each category from the ImageNet dataset. PASCAL3D+ images exhibit much more variability compared to the existing 3D datasets, and on average there are more than 3,000 object instances per category. Most prior work however do not use the added ImageNet data and restrict themselves to only the images from PASCAL VOC 2012.

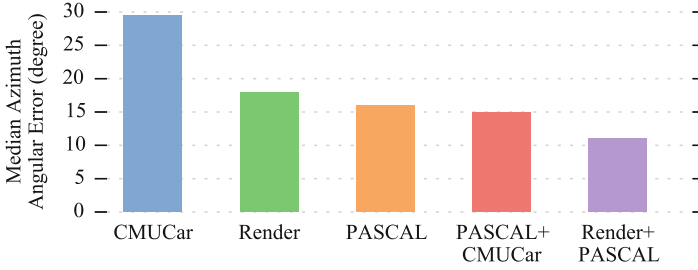
We also define two synthetic datasets, which consist the two types of rendered images described in Sect. 3 – **RenderCar** and **RenderScene**. The RenderCar dataset includes images from the entire set of 3D CAD models of vehicles. We use the various augmentation methods described above. For RenderScene we use a single car model placed in a fully modeled environment which depicts a fully realistic industrial scene as shown in Fig. 4(b), rendered over all 1800 angles as described in Sect. 3.

We focus our evaluation on the process of viewpoint prediction, and work directly on ground truth bounding boxes. Most recent work on detecting objects and estimating their viewpoint, first employ an RCNN style bounding box selection process [21] so we feel our approach is reasonable.

We split each dataset into train/validation sets and train a separate deep model on each one of the training sets. We then apply every model to the validation sets and report the results. We perform viewpoint estimation on ground truth bounding boxes for images of the car class. For PASCAL3D+ and CMU-Car these bounding boxes were obtained using annotators, and for the rendered images these were automatically created as the tightest rectangle that contains all pixels that belong to the rendered car.

Figure 5 shows the median angular azimuth error of 4 models when evaluated on the PASCAL validation set. While the model trained on PASCAL performs better than the one trained on rendered data it has an unfair advantage - the rendered model needs to overcome domain adaptation as well as the challenging task of viewpoint estimation. Note that the model trained on rendered data performs much better than the one trained on CMUCar. To us this indicates that some of the past concerns about generality of models trained from synthetic data may be unfounded. It seems that the adaptation task from synthetic data is not



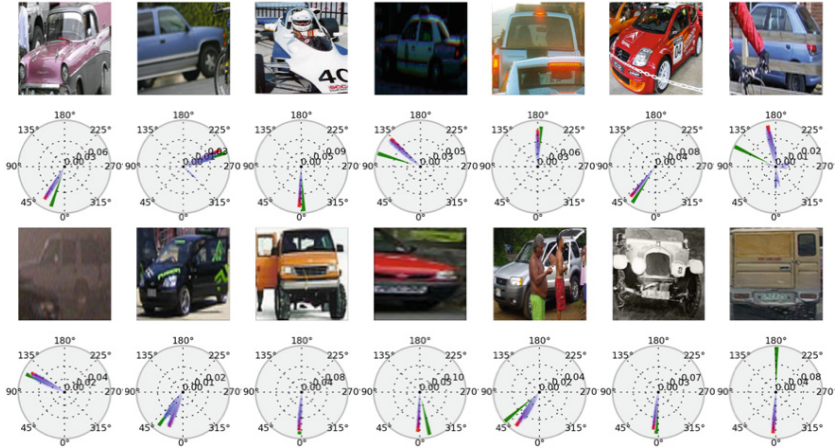


**Fig. 5.** Median azimuth angular error on the PASCAL test set for a number of models. The model trained on PASCAL is the best model trained on a single dataset, but it has the advantage of having access to PASCAL images during training. Combining PASCAL images with rendered training data performs better than combining them with additional natural images from CMUCar.

harder than from one real image set to another. Lastly note that best performance is achieved when combining real data with synthetic data. This model achieves better performance than when combining PASCAL data with images from CMUCar. CMUCar images are all street scene images taken from standing height. They have a strong bias, and add little to a model’s generalization.

Figure 6 shows example results on the ground truth bounding boxes from the PASCAL3d+ test set. Successful predictions are shown in the top two rows, and failure cases in the bottom row. The test set is characterized by many cropped and occluded images, some of very poor image quality. Mostly the model is robust to these, but when it mostly fails due to these issues, or the 180° ambiguity.

Table 1 shows model error for azimuth estimation for all train/validation combinations. First, it is easy to spot dataset bias - the best performing model on each dataset is the one that was trained on a training set from that dataset. This is consistent with the findings of [24] that show strong dataset bias effects. It is also interesting to see how some datasets are better for generalizing. The two right most columns average prediction error across multiple datasets. The *Avg* column averages across all datasets, and *Avg On Natural* averages the results on the 3 columns that use natural images for testing. Notice that the model trained on PASCAL data performs better overall than the one trained on CMUCar. Also notice that the model trained on RenderCar performs almost as well as the one trained on PASCAL. When combining data from multiple datasets the overall error is reduced. Unsurprisingly, combining all datasets produces the best results. It is interesting to note, however, that adding rendered data to one of the natural image datasets produces better results than when combining the two real-image ones. We conclude that this is because the rendered data adds two forms of variation: (1) It provides access to regions of the label space that were not present in the small natural-image datasets. (2) The image statistics of rendered images are different than those of natural images and the learner is forced to generalize better in order to classify both types of images. We further examine the effect of combining real and synthetic data below.



**Fig. 6.** Sample results of our proposed method from the test set of PASCAL3D+. Below each image we show viewpoint probabilities assigned by the model (blue), predictions (red), and ground truth (green). Right column shows failure cases. Strongly directional occluders and  $180^\circ$  ambiguity are the most common failures. (Color figure online)

*Render Quality:* Renders can be created in varying degrees of quality and realism. Simple, almost cartoon-like renders are fast to generate, while ones that realistically model the interplay of lighting and material can be quite computationally expensive. Are these higher quality renders worth the added cost?

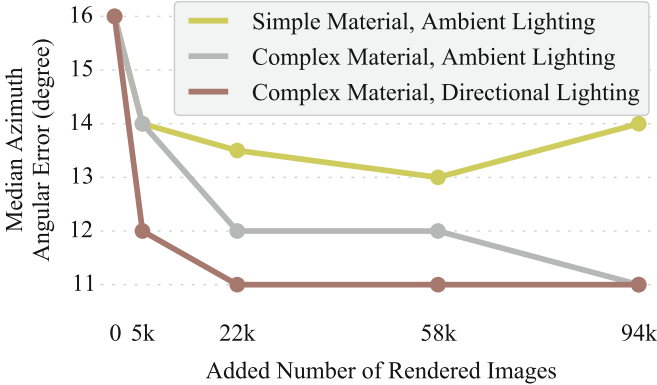
Figure 7 shows 3 render conditions we use to evaluate the effect of render quality on system performance. The top row shows the basic condition, renders created using simplified model materials, and uniform ambient lighting. In the middle row are images created using a more complex rendering procedure – the materials used are complex. They more closely resemble the metallic surface of real vehicles. We still use ambient lighting when creating them. The bottom row



**Fig. 7.** We evaluate the effect of render quality under 3 conditions: simple object material, and ambient lighting (top row); complex material and ambient lighting (middle); and a sophisticated case using complex material, and directional lighting whose location, color, and strength are randomly selected (bottom).

shows images that were generated using complex materials and directional lighting. The location, color, and strength of the light source are randomly selected.

Figure 8 shows median angular error as a function of dataset size for the 3 render quality conditions. We see that when the rendering process becomes more sophisticated the error decreases. Interestingly, when using low quality renders the error increases once the amount of renders dominate the train set. We do not see this phenomena with higher quality renders. We conclude that using complex materials and lighting is an important aspect of synthetic datasets.

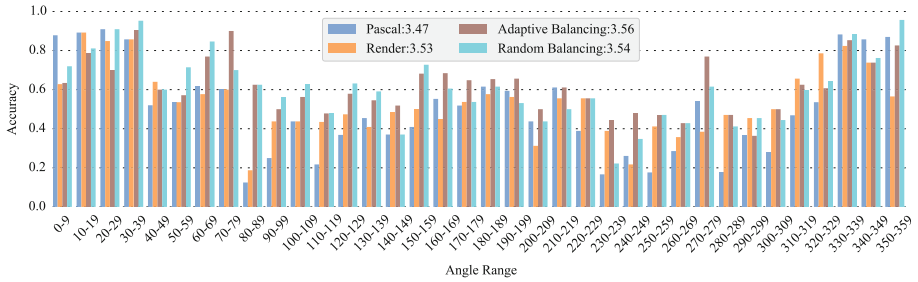


**Fig. 8.** Error as a function of dataset size for the 3 render quality conditions described in Fig. 7. All methods add rendered images to the PASCAL training set. There is a decrease in error when complex model materials are used, and a further decrease when we add random lighting conditions.

*Balancing a Training Set Using Renders:* So far we have seen that combining real images with synthetically generated ones improves performance. Intuitively this makes sense, the model is given more data, and thus learns a better representation. But is there something more than that going on? Consider a trained model. What are the properties we would want it to have? We would naturally want it to be as accurate as possible. We would also want it to be unbiased – to be accurate in all locations in feature/label space. But bias in the training set makes this goal hard to achieve. Renders can be useful for bias reduction.

In Fig. 9 we quantify this requirement. It shows the accuracy of 3 models as a function of the angle range in which they are applied. For example, a model trained on 5k PASCAL images (top row), has about 0.8 accuracy when it is applied on test images with a ground truth label in  $[0,9]$ .

We want the shape of the accuracy distribution to be as close to uniform as possible. Instead, we see that the model performs much better on some angles, such as  $[0,30]$ , than others,  $[80,100]$ . If the shape of the distribution seems familiar it is because it closely mirrors the training set distribution shown in Fig. 1. We calculate the models’ entropy on the accuracy distribution entropy as a tool for



**Fig. 9.** Model accuracy as a function of ground truth angle. A model should perform uniformly well across all angles. The entropy of each distribution (legend) shows deviation from uniform. For the model trained on PASCAL images (blue), the accuracy mirrors the train set distribution (Fig. 1). The entropy of a model trained on rendered images is higher, but still not uniform (orange). This is due to biases in the test set. Brown and light blue bars show accuracy distribution of models for which rendered images were used to balance the training set. We see they have higher entropy with adaptive balancing having the highest. Models were tested on PASCAL test images. (Color figure online)

comparison. Higher entropy indicates a more uniform distribution. When using a fully balanced training set of rendered images (2nd row), the test entropy increases. This is encouraging, as the model contains less angle bias.

Rendered images can be used as a way to balance the training set to reduce bias while still getting the benefits of natural images. We experiment with two methods for balancing which we call adaptive balancing (3rd row), and random balancing (bottom row). In adaptive balancing we sample each angle reversely proportional to its frequency in the training set. This method transforms the training set to uniform using the least number of images. In random balancing the added synthetic images are sampled uniformly. As the ratio of rendered images in the training set increases, the set becomes more uniform. For this experiment both methods added 2,000 images to the PASCAL training set. Interestingly, the two balancing methods have a similar prediction entropy. However, no method’s distribution is completely uniform. From that we can conclude that there are biases in the test data other than angle distribution, or that some angles are just naturally harder to predict than others.

These results raise an interesting question: how much of the performance gap we have seen between models trained on real PASCAL images, and those trained on rendered data stems from the angle bias of the test set? Table 2 shows the median errors of these models on a sample of the PASCAL test set in which all angles are equally represented. Once the test set is uniform we see that models based on a balanced training set perform best, and the model trained solely on PASCAL images has the worst accuracy.

Lastly, we hold constant the number of images used for training and vary the proportion of rendered images. Table 3 shows the error of models trained with varying proportions of real-to-rendered data. Models are trained using 5,000

**Table 2.** Median angular error (MAE) on a viewpoint-uniform sample of the PASCAL test set. Performance with rendered based training set is best.

| Training set: | PASCAL | Render | Adaptive balancing | Random balancing |
|---------------|--------|--------|--------------------|------------------|
| MAE:          | 26.5°  | 26.0°  | <b>16.0°</b>       | 17.5°            |

**Table 3.** Median angular error (MAE) on PASCAL test set. Training set size is fixed at 5k images (the size of the PASCAL training set) and we modify the proportion of renders used. There is an improvement when replacing up to  $\frac{1}{2}$  of the images with renders. The renders help balance the angle distribution and reduce bias.

| Renders in train set: | 0%  | 25%        | 50%        | 75% | 100% |
|-----------------------|-----|------------|------------|-----|------|
| Median angular error: | 16° | <b>14°</b> | <b>14°</b> | 15° | 22°  |

images – the size of the PASCAL training set. Notice that there is an improvement when replacing up to 50% of the images with rendered data. This is likely due to the balancing effect of the renders on the angle distribution which reduces bias. When most of the data is rendered, performance drops. This is likely because the image variability in renders is smaller than real images. More synthetic data is needed for models based purely on it to achieve the lowest error – 18°.

*Size of Training Set:* We examine the role of raw number of renders in performance. We keep the number of CAD models constant at 90, and uniformly sample renders from our pool of renders. Many of the cars in PASCAL are only partially visible, and we want the models to learn this, so in each set half the rendered images show full cars, and half contain random 60% crops Table 4 summarizes the results of this experiment. Better performance is achieved when increasing the number of renders, but there are diminishing returns. There is a limit to the variability a model can learn from a fixed set of CAD models, and one would need to increase the size of the model set to overcome this. Obtaining a large amount of 3D models can be expensive and so this motivates creation of large open source model datasets.

*Loss Layer:* When using Von Mises kernel-based SoftMax loss function is that it is impossible to obtain zero loss. In a regular SoftMax, if the model assigns prob-

**Table 4.** Effect of training set size and loss layer. We see a trend of better performance with increased size of training set, but the effect quickly diminishes. Clearly, more than just training set size influences performance. The Von Mises weighted SoftMax (wSM) performs better than regular SoftMax (SM) for all train sizes.

| Train set size | 10k |       | 50k   |     | 330k |       | 820k |            |
|----------------|-----|-------|-------|-----|------|-------|------|------------|
| Loss layer     | SM  | wSM   | SM    | wSM | SM   | wSM   | SM   | wSM        |
| MAE            | 31° | 27.5° | 54.5° | 21° | 26°  | 21.5° | 23°  | <b>18°</b> |

**Table 5.** Effect of kernel width ( $\sigma$ ) of the Von Mises kernel. The loss function is not sensitive to the selection of kernel width.

| $\sigma$             | 2   | 3   | 4   | 10  | 15  |
|----------------------|-----|-----|-----|-----|-----|
| Effective width      | 6°  | 8°  | 12° | 30° | 50° |
| Median angular error | 13° | 14° | 14° | 13° | 15° |

**Table 6.** Effect of added occlusions to the training set. All models evaluated used datasets of 50,000 rendered images.

| % Occluded images | 0.0 | 0.1 | 0.35       | 0.4 | 0.5 | 1.0 |
|-------------------|-----|-----|------------|-----|-----|-----|
| MAE               | 25° | 25° | <b>21°</b> | 25° | 28° | 26° |

ability 1.0 to the correct class it can achieve a loss of zero. In the weighted case, when there is some weight assigned to more than one class, a perfect prediction will actually result in infinite loss, as all other classes have probability values of zero, and the log-loss assigned to them will be infinite. Minimum loss will be reached when prediction probabilities are spread out across nearby classes. This is both a downside of this loss function, but also its strength. It does not let the model make wild guesses at the correct class. Nearby views are required to have similar probabilities. It trades angle resolution with added prediction stability.

Table 4 compares the results of SoftMax based models and models trained using the weighted SoftMax layer over a number of rendered dataset sizes. The weighted loss layer performs better for all sizes. This supports our hypothesis that viewpoint estimation is not a standard classification task. Most experiments in this section were performed using  $\sigma = 2$  for the Von Mises kernel in Equation (2). This amounts to an effective width of 6°, meaning that predictions that are farther away from the ground truth will not be assigned any weight. Table 5 shows the effect of varying this value. Interestingly we see that the method is robust to this parameter, and performs well for a wide range of values. It appears that even a relatively weak correlation between angles provides benefits.

*Occlusion:* The PASCAL test set contains many instances of partially occluded cars. When augmenting synthetic data we add randomly sized occlusions to a subset of the rendered images. Table 6 shows that there is some benefit from generating occlusions, but when too many of the training images are occluded it becomes harder for the model to learn. What is the best way to generate occlusions? In our work we have experimented with simple, single color, rectangular occlusions, as well as occlusions based on random patches from PASCAL. We saw no difference in model performance. It would be interesting to examine the optimal spatial occlusion relationship. This is likely to be object class dependent.

## 6 Discussion

In this work we propose the use of rendered images as a way to automatically build datasets for viewpoint estimation. We show that models trained on renders are competitive with those trained on natural images – the gap in performance can be explained by domain adaptation. Moreover, models trained on a combination of synthetic/real data outperform ones trained on natural images.

The need for large scale datasets is growing with the increase in model size. Based on the results detailed here we believe that synthetic data should be an

important part of dataset creation strategies. We feel that a combination a small set of carefully annotated images, combined with a larger number of synthetic renders, with automatically assigned labels, has the best cost-to-benefit ratio.

This strategy is not limited to viewpoint estimation, and can be employed for a range of computer vision tasks. Specifically we feel that future research should focus on human pose estimation, depth prediction, wide-baseline correspondence learning, and structure from motion.

## References

1. Bileschi, S.M.: StreetScenes: towards scene understanding in still images. Ph.D. thesis, Massachusetts Institute of Technology (2006)
2. Boddeti, V.N., Kanade, T., Kumar, B.: Correlation filters for object alignment. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2013)
3. Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Ranzato, M., Senior, A., Tucker, P., Yang, K., Le, Q.V., Ng, A.Y.: Large scale distributed deep networks. In: Advances in Neural Information Processing Systems (NIPS), pp. 1223–1231 (2012)
4. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2009)
5. Everingham, M., Van Gool, L., Williams, C., Winn, J., Zisserman, A.: The pascal visual object classes challenge 2011 (VOC2011) results (2011)
6. Fatahalian, K.: Evolving the real-time graphics pipeline for micropolygon rendering. Ph.D. thesis, Stanford University (2011)
7. Goodfellow, I.J., Bulatov, Y., Ibarz, J., Arnoud, S., Shet, V.: Multi-digit number recognition from street view imagery using deep convolutional neural networks. arXiv preprint [arXiv:1312.6082](https://arxiv.org/abs/1312.6082) (2013)
8. Group, C: Vray rendering engine (2015). <http://www.chaosgroup.com>
9. Hattori, H., Naresh Boddeti, V., Kitani, K.M., Kanade, T.: Learning scene-specific pedestrian detectors without real data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
10. Heimerl, K., Gawalt, B., Chen, K., Parikh, T., Hartmann, B.: CommunitySourcing: engaging local crowds to perform expert work via physical kiosks. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM (2012)
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)
12. Law, E., Settles, B., Snook, A., Surana, H., Von Ahn, L., Mitchell, T.: Human computation for attribute and attribute value acquisition. In: Proceedings of the First Workshop on Fine-Grained Visual Categorization (FGVC) (2011)
13. Lepetit, V., Moreno-Noguer, F., Fua, P.: EPnP: an accurate  $O(n)$  solution to the PnP problem. *International Journal Computer Vision* **81**, 155–166 (2009)
14. Movshovitz-Attias, Y., Naresh Boddeti, V., Wei, Z., Sheikh, Y.: 3D pose-by-detection of vehicles via discriminatively reduced ensembles of correlation filters. In: Proceedings of the British Machine Vision Conference (BMVC), Nottingham, UK, September 2014



15. Movshovitz-Attias, Y., Yu, Q., Stumpe, M., Shet, V., Arnoud, S., Yatziv, L.: Ontological supervision for fine grained classification of street view storefronts. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
16. Nene, S.A., Nayar, S.K., Murase, H., et al.: Columbia object image library (coil-20). Technical report CUCS-005-96 (1996)
17. Oliva, A., Torralba, A.: Modeling the shape of the scene: a holistic representation of the spatial envelope. *Int. J. Comput. Vis.* **42**(3), 145–175 (2001)
18. Pepik, B., Benenson, R., Ritschel, T., Schiele, B.: What is holding back convnets for detection? *CoRR* (2015). <http://arxiv.org/abs/1508.02844>
19. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.* **115**, 211–252 (2014)
20. Stark, M., Goesele, M., Schiele, B.: Back to the future: learning shape models from 3D cad data. In: Proceedings of the British Machine Vision Conference (BMVC) (2010)
21. Su, H., Qi, C.R., Li, Y., Guibas, L.J.: Render for CNN: viewpoint estimation in images using CNNs trained with rendered 3D model views. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2015)
22. Sun, B., Saenko, K.: From virtual to reality: fast adaptation of virtual object detectors to real domains. In: Proceedings of the British Machine Vision Conference (BMVC) (2014)
23. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. [arXiv:1409.4842](https://arxiv.org/abs/1409.4842) [cs], September 2014
24. Torralba, A., Efros, A.: Unbiased look at dataset bias. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2011)
25. Vazquez, D., Lopez, A.M., Marin, J., Ponsa, D., Geronimo, D.: Virtual and real world adaptation for pedestrian detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **36**, 797–809 (2014)
26. Von Ahn, L., Dabbish, L.: Labeling images with a computer game. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM (2004)
27. Wu, R., Yan, S., Shan, Y., Dang, Q., Sun, G.: Deep image: scaling up image recognition. arXiv preprint [arXiv:1501.02876](https://arxiv.org/abs/1501.02876) (2015). <http://arxiv.org/abs/1501.02876>
28. Xiang, Y., Mottaghi, R., Savarese, S.: Beyond pascal: a benchmark for 3D object detection in the wild. In: Winter Conference on Applications of Computer Vision (WACV) (2014)