# Using Images Rendered by PBRT to Train Faster R-CNN for UAV Detection

Junkai Peng[1], Changwen Zheng[2], Pin Lv[3], Tianyu Cui[4], Ye Cheng[5] and Lingyu Si[6]
Institute of Software, University of Chinese Academy of Sciences
4# South Fourth Street, Zhong Guan Cun, 100190, Beijing, P.R. China
pengjunkai15@mails.ucas.ac.cn[1], {changwen[2], lvpin[3], cuitianyu[4]}@iscas.ac.cn
chengye2010@aliyun.com[5], ls16200@my.bristol.ac.uk[6]

## ABSTRACT

Deep neural networks, such as Faster R-CNN, have been widely used in object detection. However, deep neural networks usually require a large-scale dataset to achieve desirable performance. For the specific application, UAV detection, training data is extremely limited in practice. Since annotating plenty of UAV images manually can be very resource intensive and time consuming, instead, we use PBRT to render a large number of photorealistic UAV images of high variation within a reasonable time. Using PBRT ensures the realism of rendered images, which means they are indistinguishable from real photographs to some extent. Trained with our rendered images, the Faster R-CNN has an AP of 80.69% on manually annotated UAV images test set, much higher than the one only trained with COCO 2014 dataset and PASCAL VOC 2012 dataset (43.36%). Moreover, our rendered image dataset contains not only bounding boxes of all UAVs, but also locations of some important parts of UAVs and locations of all pixels covered by UAVs, which can be used for more complicated application, such as mask detection or keypoint detection.

## Keywords

Object detection, deep learning, Faster R-CNN, PBRT, UAV

## 1 INTRODUCTION

As the rapid development of the powerful technologies of UAVs, more and more individuals can use UAVs to do creative works. Nevertheless, UAVs must be regulated in public area or no-fly zone, otherwise UAVs can become potential threats to public security and privacy.

A crucial step in the regulation of UAVs is detecting UAVs in videos rapidly. UAV detection means finding the location of each UAV for every frame of a video. More specifically, it draws a smallest rectangle that can cover all the pixels of a target UAV. Object detection in computer vision provides lots of methods to address this problem. With the development of deep learning in recent years, deeper and more complex convolutional neural networks (CNN) [1-5] have been designed to detect objects in videos efficiently, and have reached state-of-the-art performance.

However, training these CNNs requires a huge amount of data. Fortunately, several large-scale datasets are publicly available on the Internet, such as PASCAL VOC dataset [6] and COCO dataset [7], whose images are collected from the Internet and annotated through crowdsourcing. In addition, COCO dataset [7] includes not only objects' categories and positions, but also each individual's mask.

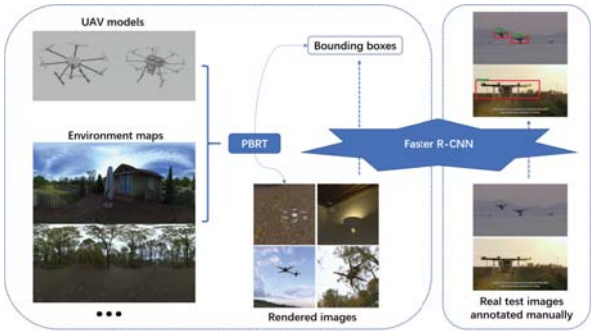


Figure 1: System overview. In contrast to previous methods, we use PBRT with environment maps as light source to render more photorealistic UAV images in a reasonable time.

Although these datasets contain a number of airplane (or aeroplane) images, more annotated UAV images are needed to promote the accuracy of UAV detection. Undoubtedly, annotating lots of UAV images through crowdsourcing is time consuming. In this manuscript, we propose to use Physically Based Rendering Toolkit

Figure 2: Some of our rendered images for training Faster R-CNN.

(PBRT) [8] (a slightly modified version) in computer graphics to render photorealistic images, and in the meantime, to calculate corresponding accurate bounding boxes. Figure 1 shows the overall process. By choosing different (1) positions and orientations of UAVs, (2) 3D models of UAVs, (3) appearance materials of UAVs, (4) camera intrinsics and extrinsics, (5) environment maps and (6) postprocessing methods of the rendered images, a large variety of images can be rendered. The greatest advantage of acquiring UAV images by rendering is that not only accurate bounding boxes of UAVs can be easily obtained, but also the positions of any parts of UAVs (e.g. lifting rotor), even all the pixels occupied by the UAVs, can be recorded. Actually, supervised learning assumes that training data and testing data should be independent and identically distributed, which means the rendered images for training must be photorealistic and diverse. This is the reason that we use PBRT to render UAV images. Figure 2 shows some of the rendered images, which are real enough and various.

Faster R-CNN trained with PASCAL VOC 2012 dataset mixed with our rendered images has an AP of 80.69% on manually annotated UAV images test set, while Faster R-CNN purely trained with PASCAL VOC 2012 dataset, which contains hundreds of airplane images, has an AP of 43.36% on the same test set. This sufficiently validate our rendered UAV images and our work lays a foundation for using images rendered by PBRT to train Faster R-CNN. Obviously, these rendered images can also be used in broader applications, for example, instance segmentation (mask R-CNN [9]) and keypoint detection.

## 2 RELATED WORK

**Object detection.** In this manuscript, we pay more attention to validating our rendered training data and talk about its possible usages in more complicated applications. Therefore, we directly use a newly released framework Detectron [10], which trained and tested a lot of state-of-the-art object detection models for our reference, to build our UAV detection system without modification.

From Fast R-CNN [1] to Faster R-CNN [2], from YOLOv1 [3] to YOLOv2 [4], more and more effective methods have been emerging constantly to make object detection faster and more accurate. Deeper and deeper CNNs need more training data to avoid overfitting. So, next we introduce some previous methods about how to use rendered images to provide extensive training data.

**Using synthetically generated images.** Large-scale data is of significant importance for training deep neural networks. A variety of datasets with different characteristics have promoted many fields in computer vision. For example, ImageNet dataset [11] is necessary for the breakthroughs in both object classification and detection; COCO dataset [7] plays an important role in scene understanding.

As for synthetically generated images, Dosovitskiy et al. [12] created a simple synthetic 2D dataset of flying chairs for training their network which was proved to be sufficient to predict accurate optical flow in general videos. Inspired by this, Mayer et al. [13] used a customed version of the open source 3D creation suite Blender to render three dataset for training and evaluating scene flow methods. Su et al. [14] proposed


ISBN 978-80-86943-41-1

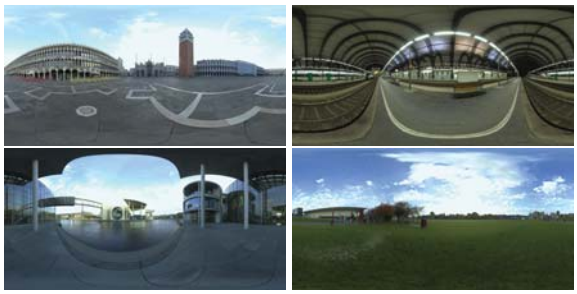

Figure 3: Some environment maps used in rendering.

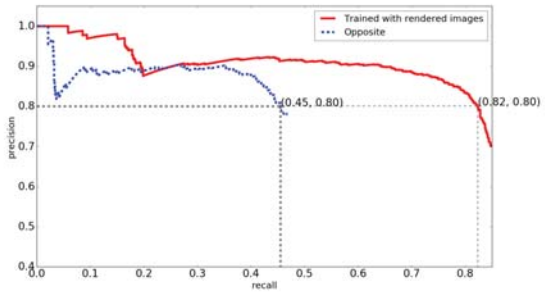


Figure 4: The precision-recall curves.

Figure 5: Left pair: using red-specular-plastic. Right pair: using aluminium. For each pair, the left image is rendered with default method and the right image is rendered with fitted NPF model and using its fitted D factor for importance sampling. Pay more attention to the white noise on the UAVs

a synthesis pipeline that generated millions of images with accurate viewpoint labels for viewpoint estimation. But rather than pursuing realistic effect, they put more effort to generate images of high diversity. Therefore they used alpha-composition to blend a rendered 3D model as foreground and a scene image as background. In contrast to them, we used environment maps as light sources to make rendered images more realistic. Peng et al. [15] also used synthetic images to train deep object detectors. They explored the complex invariance encoded in the features learned by CNN. One of their major conclusions was, when learning a detection model for a new category with no or limited labeled real data available, it was advantageous to simulate texture, color and pose in the synthetic data. Using PBRT, we can take all these factors into account easily. Aker et al. [16] simply combined background-subtracted real images to create an extensive artificial UAV dataset for training a UAV detection network. This method is only suitable for small UAVs because it does not take lighting condition into account. Cutting a medium or large UAV into another image makes the UAV look abrupt. Moreover, they need to segment some UAVs from background. In contrast to this method, we use PBRT [8] to render photorealistic UAV images, which can not only use detailed 3D models and modify their appearance materials, but also set the position of the UAV relative to the camera arbitrarily. PBRT [8] is a modern photorealistic rendering system that can even render vivid natural scenes, although it may take a considerably long time. In next section we introduce how to use PBRT with environment maps and measured materials to render real enough UAV images within a reasonable time.

## 3 REALISTIC IMAGE SYNTHESIS

Modeling all the objects, including UAVs, trees, houses, roads and so on, to form a natural scene will take PBRT or other renderers an unbearably long time to render an image. Previous methods simply combined real background images with rendered 3D models, which lost realism to some extent. Here we propose to use environment maps, which are images of the distant environment surrounding the rendered object. As light sources in the scene, environment maps provide illuminations shining on the UAV from all 360° angles. Usually, an environment map is synthesized by photos of a same scene taken under several specific angles. They can also be rendered by PBRT through careful design. All the 90 environment maps used to render our training set are downloaded from HDRI Haven[1]. Besides in Figure 1, more environment maps are shown in Figure 3.

Detailed UAV 3D models are another key part in rendering photorealistic images. In this manuscript we totally only use five detailed UAV models, two of which are shown in the top left corner of Figure 1. Peng et al. [15] showed a significant boost from adding more shape variation to the training data for Fast R-CNN. Therefore, it is convinced that using more 3D models to render more UAV images for training can further improve the performance of UAV detection. Unfortunately, there is still a lack of freely available and detailed UAV 3D models.

In addition, in order to increase the diversities of the training set, the MERL database [17], which includes 100 different accurately measured materials, are used in rendering. However, directly using these measured data with default importance sampling method of PBRT requires a large number of samples per pixel for rendering, which takes PBRT a relatively long time to render an image. Therefore, a BRDF model named non-parametric factor microfacet model (a modified version), which was first designed by Bagher et al. [18] and was much more accurate than other microfacet

[1] https://hdrihaven.com/


ISBN 978-80-86943-41-1

| Faster R-CNN | AP |
|---|---|
| Pretrained with COCO 2014 dataset | 43.03% |
| Finetuned with only PASCAL VOC 2012 dataset | 43.36% |
| Finetuned with only our rendered training set (without occluded images) | 56.28% |
| Finetuned with PASCAL VOC 2012 dataset and our rendered training set (without occluded images) | 79.51% |
| Finetuned with PASCAL VOC 2012 dataset and our rendered training set (with occluded images) | **80.69%** |

Table 1: Testing results.

Figure 6: Some of our test images. UAVs in these images can not be detected by the network only trained with COCO dataset and PASCAL VOC dataset but can be detected by the network trained with them mixed with our rendered images.

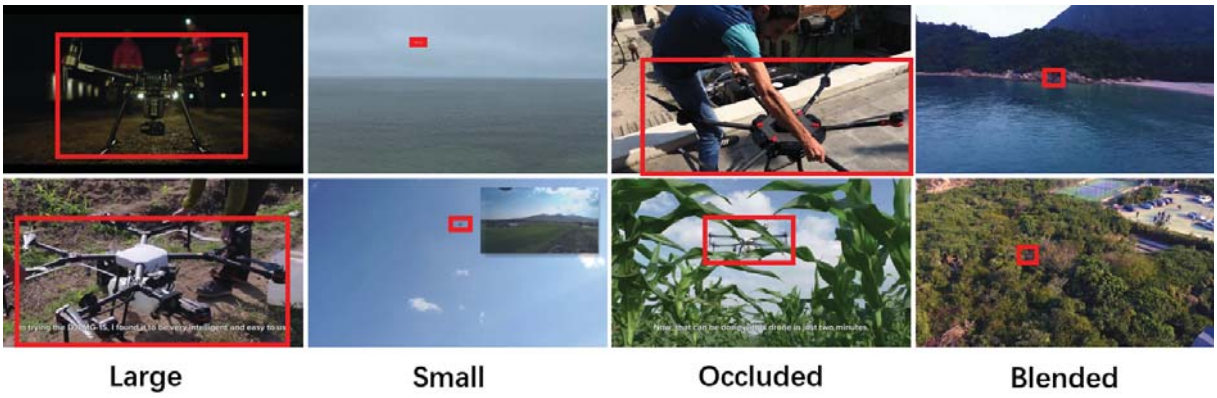


Figure 7: Some of our test images. UAVs in these images can not be detected by both the networks trained and not trained with our rendered images. It is worth noting that not all large, small, occluded or blended UAVs can not be detected.

models for fitting, is used to fit these measured data and the corresponding fitted D factors are used for importance sampling. This method dramatically reduces the needed number of samples per pixel for rendering. In other words, within the same time, this method can render a higher quality image. For example, Figure 5 shows two pairs of UAV images rendered with measured data, red-specular-plastic and aluminium respectively. For each pair of images, the left one is rendered with default importance sampling method while the right one is rendered with aforementioned method. All the images are rendered with 64 samples per pixel. Obviously, for each pair, the right image has a higher quality, especially the aluminium pair (pay more attention to the white noise on the UAVs).

Moreover, the positions and orientations of camera and UAV model can also be used to increase the diversities of the training set. Rotating around the UAV, the camera takes several photos for every 60°. The UAV also rolls $[-45°, 45°]$, for which the interval is set to 15°. The distance between the UAV and the camera is set to 4m, 8m or 12m. The field of view of the camera is about 30°. This parameter setting is not immutable.

Motion blur is not present in UAV videos except for the rotor wings. But those static UAVs should also be detected. Therefore, both static UAVs and lifting UAVs are rendered. We use Blender to split the UAV model into several parts and set rotation speed of its rotor wings to be 50 revolutions per second in PBRT scene files. The shutter speed is about 0.005 second.

Totally, we rendered 60480 UAVs images for training. It took Intel i7-4710MQ approximately 3 seconds to render an image and record its corresponding bounding box. Consequently, it took about two day to render the whole training set. Compared with annotating manually, this time is negligible. The quality and efficiency


ISBN 978-80-86943-41-1

of rendering can be further improved by using the OptiX API with the AI-accelerated denoiser [19].

## 4 FASTER R-CNN TRAINING AND TESTING RESULT

We use Detectron [10] provided by Facebook AI Research to finetune Faster R-CNN with our rendered training set. Detectron model zoo provides some models pretrained with COCO dataset, which can be used to initialize our network. Finally ResNet-101 model is selected as the backbone model, which has closer performance to ResNeXt-101-32x8d model but much less inference time [10]. Initialized by this pretrained model, we first trained Faster R-CNN with PASCAL VOC 2012 dataset. Then PASCAL VOC 2012 dataset is mixed with our rendered dataset to train another Faster R-CNN. The net gain from our rendered dataset is checked by comparing the performance of these two trained network.

We do not split the rendered images to evaluate the trained detection networks. Instead, many key frames cut from 10 real UAV introduction videos that are downloaded from Internet are annotated manually by us to form a test set (about 994 images including UAVs). Some of these test images are shown in Figure 6 and Figure 7. This test set is used to evaluate the trained detection network. All the testing results are shown in Table 1 and their corresponding precision-recall curves are shown in Figure 4. Actually, COCO 2014 dataset and PASCAL VOC 2012 dataset do not have any UAV images, although they have plenty of airplane (or aeroplane) images. However, Faster R-CNNs trained with them still have an AP of 43% on our test set. As shown in Figure 4, they have relatively low recall. That is, they can detect UAVs in some simple scenes but do not work for relatively complex scenes. In addition, Table 1 also shows that, finetuned with PASCAL VOC 2012 dataset, the performance of Faster R-CNN is neither increased nor decrease.

Before our rendered images are mixed into PASCAL VOC 2012 dataset, they are divided into two sets, first of which contains 38880 images that only have one UAV. The second set contains 21600 images that have two UAVs and most of them are occluded by each other. Some of these rendered images are shown in Figure 2. It is noted that Faster R-CNN trained with PASCAL VOC 2012 dataset mixed with the first set has an AP of 79.51% (Table 1), which is much higher than that only trained with PASCAL VOC 2012 dataset (43.36%). However, Faster R-CNN trained with PASCAL VOC 2012 dataset mixed with both two sets has an AP of 80.69%, which only slightly larger than 79.51%. It seems that UAVs occluded with each other in the second set do not provide more information. But in fact,

Figure 8: The UAV is occluded in both images but the right one can be detected.

there are only a few test images containing UAV occluded by people or plants (see Figure 7). In addition, Faster R-CNN purely trained with the first set has an AP of 56.28%. To some extend, the diversities of our rendered images are limited by the number of UAV models and environment maps used in rendering. Using more models, environment maps or even camera settings will definitely further promote the performance of Faster R-CNN.

Figure 6 shows some images from the test set, for which the network trained with our rendered images can detect the UAVs inside them but the other one not trained with our rendered images cannot. The threshold is set to 0.7. These images span from small UAVs to large UAVs. One of the UAV is even partly occluded. Therefore, the qualitative and quantitative results show that our rendered trained images make the trained Faster R-CNN more general.

There are still some images of complex scenes that the networks neither trained nor not trained with our rendered images can detect UAVs inside them. Some of these images are shown in Figure 7. It is worth noting that not all too large, too small, occluded or blended UAVs can not be detected. As shown in Figure 8, the UAV is occluded in both images but the UAV in the right image can be detected.

These undetected images also guide the direction for us. Firstly, we need to prepare more training images which include big and detailed UAVs, or small and blur UAVs. Next, we needed to add special lighting conditions in some images. Additionally, images of UAVs occluded by persons or plants are not easy to render but images of UAVs occluded by cars, boats or houses are relatively easy. Last but not least, as indicated by Su et al. [14], using more 3D models will also help resist overfitting of the R-CNN.

## 5 FUTURE WORK

Besides aforementioned work, we also try to use our rendered images to train mask-RCNN and do key point detection.

Moreover, it is completely possible to render videos of flying UAVs. But if the UAVs need to interact with other objects, the scene files will be more complex and the rendering time will be longer. But with carefully designed scene files and accelerated by the OptiX API

 ISBN 978-80-86943-41-1

[20], some relatively simple natural scenes are possible to be rendered in a reasonable time.

## 6 CONCLUSION

We present a synthetic UAV dataset with sufficient realism, variation and size. It is rendered by PBRT with measured reflection materials and environment maps that reduce the rendering time but still promise the realism. By comparing the AP of Faster R-CNN detection networks trained with and without our rendered images, we conclude that, UAV images rendered by the method mentioned above do promote the performance of the UAV detection network.

## 7 REFERENCES

[1] Girshick, R.: Fast R-CNN. In: 2015 IEEE International Conference on Computer Vision (ICCV) pp. 1440–1448 (2015)

[2] Ren, S.Q, He, K.M, Girshick, R., Sun, J.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence 36(6), 1137–1149 (2017)

[3] Redmon, J. and Divvala, S. and Girshick, R. and Farhadi, A.: You Only Look Once: Unified, Real-Time Object Detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 779–788 (2016)

[4] Redmon, J. and Farhadi, A.: YOLO9000: Better, Faster, Stronger. Towards Real-Time Object Detection with Region Proposal Networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 6517–6525 (2017)

[5] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: SSD: Single Shot MultiBox Detector. In: Computer Vision – ECCV 2016 pp. 21–37 (2016)

[6] Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The Pascal Visual Object Classes (VOC) Challenge. International Journal of Computer Vision 88(2), 303–338 (2010)

[7] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common Objects in Context. In: Computer Vision – ECCV 2014 pp. 740–755 (2014)

[8] Pharr, M., Jakob, W., Humphreys, G.: Physically Based Rendering: From Theory to Implementation. Morgan Kaufmann, San Francisco, 2016

[9] He, K.M., Gkioxari, G., DollÃ¡r, P., Girshick, R.: Mask R-CNN. In: 2017 IEEE International Conference on Computer Vision (ICCV) pp. 2980-2988 (2017)

[10] Ross, G., Ilija, R., Georgia, G., Piotr, D., He, K.M.: Detectron. `https://github.com/facebookresearch/detectron`, 2018

[11] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Li, F.F.: ImageNet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 248-255 (2009)

[12] Dosovitskiy, A., Fischery, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Smagt, P.V.D., Cremers, D., Brox, T.: FlowNet: Learning Optical Flow with Convolutional Networks. In: 2015 IEEE International Conference on Computer Vision (ICCV) pp. 2758-2766 (2015)

[13] Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T.: A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 4040-4048 (2016)

[14] 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: 2015 IEEE International Conference on Computer Vision (ICCV) pp. 2686-2694 (2015)

[15] Peng, X.C., Sun, B.C., Ali, K, Saenko, K.: Exploring Invariances in Deep Convolutional Neural Networks Using Synthetic Images. CoRR abs/1412.7122 (2014)

[16] Aker, C., Kalkan, S.: Using deep networks for drone detection. In: 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS) pp. 1-6 (2017)

[17] Matusik, W., Pfister, H., Brand, M., McMillan, L.: A Data-Driven Reflectance Model. ACM Transactions on Graphics 22(3), 759-769 (2003)

[18] Bagher, M.M., Snyder, J., Nowrouzezahrai, D.: A Non-Parametric Factor Microfacet Model for Isotropic BRDFs. ACM Transactions on Graphics 35(5), 159:1–159:16 (2016)

[19] Chaitanya, C.R.A., Kaplanyan, A.S., Schied, C., Salvi, M., Lefohn, A., Nowrouzezahrai, D., Aila, T.: Interactive Reconstruction of Monte Carlo Image Sequences Using a Recurrent Denoising Autoencoder. ACM Transactions on Graphics 36(4), 98:1–98:12 (2017)

[20] Parker, S.G., Bigler, J., Dietrich, A., Friedrich, H., Hoberock, J., Luebke, D., McAllister, D., McGuire, M., Morley, K., Robison, A., Stich, M.: OptiX: A General Purpose Ray Tracing Engine. ACM Transactions on Graphics 29(4), 66:1–66:13 (2010)

 ISBN 978-80-86943-41-1