Learning Safety Equipment Detection using Virtual Worlds

[1]Marco Di Benedetto, [2]Enrico Meloni, [1]Giuseppe Amato, [1]Fabrizio Falchi, [1]Claudio Gennaro
[1] *Institute of Information Science and Technologies, National Research Council*, Italy, {name.surname}@isti.cnr.it
[2] *Department of Information Engineering, Università di Pisa*, Italy, enrico.meloni@outlook.it

***Abstract*—Nowadays, the possibilities offered by state-of-the-art deep neural networks allow the creation of systems capable of recognizing and indexing visual content with very high accuracy. Performance of these systems relies on the availability of high quality training sets, containing a large number of examples (e.g. million), in addition to the the machine learning tools themselves.**

For several applications, very good training sets can be obtained, for example, crawling (noisily) annotated images from the internet, or by analyzing user interaction (e.g.: on social networks). However, there are several applications for which high quality training sets are not easy to be obtained/created. Consider, as an example, a security scenario where one wants to automatically detect rarely occurring threatening events.

In this respect, recently, researchers investigated the possibility of using a visual virtual environment, capable of artificially generating controllable and photo-realistic contents, to create training sets for applications with little available training images.

We explored this idea to generate synthetic photo-realistic training sets to train classifiers to recognize the proper use of individual safety equipment (e.g.: worker protection helmets, high-visibility vests, ear protection devices) during risky human activities. Then, we performed domain adaptation to real images by using a very small image data set of real-world photographs.

We show that training with the generated synthetic training set and using the domain adaptation step is an effective solution to address applications for which no training sets exist.

***Index Terms*—Deep Learning, Virtual Dataset, Transfer Learning, Domain Adaptation, Safety Equipment Detection**

I. Introduction

In the new spring of artificial intelligence, and in particular in its sub-field known as machine learning, a significant series of important results have shifted focus of industrial and research communities toward the generation of valuable data from which learning algorithms can be trained. For several applications, in the era of big data, the availability of real input examples, to train machine learning algorithms, is not considered an issue. However, for several other applications there is not such an abundance of training data. Sometimes, even if data is available it must be manually revised to make it usable as training data (e.g., by adding annotations, class labels, or visual masks), with a considerable cost.

In fact, although a series of annotated datasets are available and successfully used to produce important academic results and commercially fruitful products, there is still a huge amount of scenarios where laborious human intervention is needed to produce high quality training sets.

For example, such cases include, but are not limited to, safety equipment detection, weapon wielding detection, and autonomous driven cars.

To overcome these limitations and to provide useful examples in a variety of scenarios, the research community has recently started to leverage on the use of programmable virtual scenarios to generate visual datasets and the neede associated annotations. For example, in an image-based machine learning technique, using a modern rendering engine (i.e., capable of producing photo-realistic imagery) has been proven a valid companion to automatically generate adequate datasets (see Section II).

In this work we demonstrate the effectiveness of a virtual rendering engine to address the problem of detection and recognition in scenarios where little-to-no real images exist, and apply it in the context of safety equipment visual detection (see Figure 1), for which, to the best of our knowledge, no public dataset exists. In particular, we show how the *transfer learning* approach on a known deep neural network can reach state-of-the-art results in automatic visual media indexing, after being trained with virtually generated images containing people equipped with safety items, like high-visibility jackets and helmets, and domain adaptation using a few real image training examples. More in detail, we contribute in this field with the following results:

- creation of a virtual training set for personal safety equipment recognition, with different scene conditions,
- creation of an annotated real-world image test set, and
- creation of state-of-the-art classifiers for such scenario.

We will see that, in case of very few real available examples, the accuracy boost given by virtual images dramatically increases the system performance. The dataset that we created is made publicly available to the research community [1].

This work is organized as follows: Section II gives an overview of existing methods based on virtual environments; Section III describes how we used an existing rendering engine and the policy to create the dataset and the test set; Section IV discusses our detection method; Section V shows our experimental results; finally Section VI concludes.

II. Related Work

With the advent of deep learning, object detection technologies have achieved accuracies that were unimaginable only a

Final authenticated publication © 2019 IEEE:
https://doi.org/10.1109/CBMI.2019.8877466

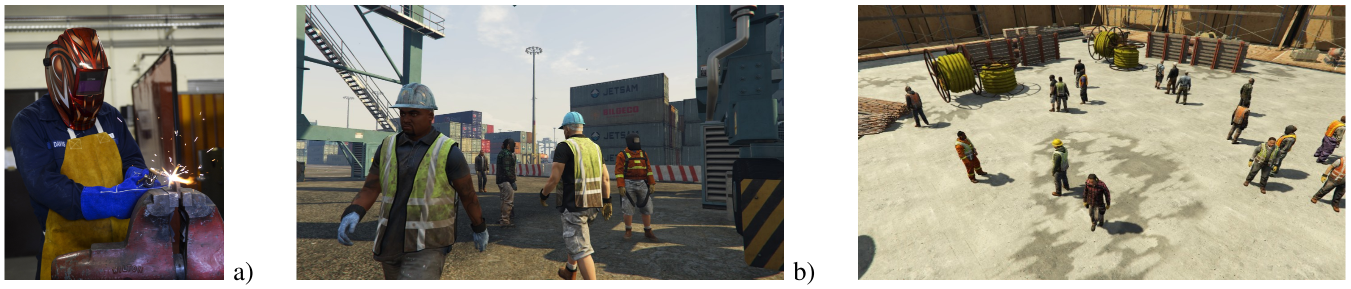

Fig. 1. Examples of safety equipment: a) real photograph of worker wearing a welding mask; b) and c) virtual renderings with people with helmets, high-visibility vests, and welding masks.

few years ago. YOLO architectures [2], [3] and Faster-RCNN [4] are today de facto-standard architectures for the object detection task. They are trained on huge generic annotated datasets, such as ImageNet [5], MS COCO [6], Pascal [7] or OpenImages v4 [8]. These datasets collect an enormous amount of pictures usually taken from the web and they are *manually* annotated.

With the need for huge amounts of labeled data, virtually generated datasets have recently gained great interest. The possibility of learning features from virtual data and validating them on real scenarios was explored in [9]. Unlike our work, however, they did not explore deep learning approaches. In [10], computer-generated imagery were used to study trained CNNs to qualitatively and quantitatively analyze deep features by varying the network stimuli according to factors of interest, such as to object style, viewpoint and color. The works [11], [12] exploit the popular Unreal Engine 4 (UE4) to build virtual worlds and use them to train and test deep learning algorithms.

The problem of transferring deep neural network models trained in simulated virtual worlds to the real world for vision-based robotic control was explored in [13]. In a similar scenario, [14] developed an end-to-end active tracker trained in virtual environment that can adapt to real world robot settings. To handle the variability in real-world data, [15] relied upon the technique of domain randomization, in which the parameters of the simulatorsuch as lighting, pose, object textures were randomized in non-realistic ways to force the neural network to learn the essential features of the object of interest. A deep learning model was trained in [16] to drive in a simulated environment and adapted it for the visual variation experienced in the real world.

[17], [18] focused their attention on the possibility to perform domain adaptation in order to map virtual features onto real ones. Richter et al. [19] explored the use of the video game *Grand Theft Auto V* (GTA-V) [20] for creating large-scale pixel-accurate ground truth data for training semantic segmentation systems. In [21], they used GTA-V for training a self-driving car and generated around 480,000 images for training. This work evidenced how GTA-V can indeed be used to automatically generate a large dataset. The use of GTA-V to train a self-driving car was explored also in [22], where images from the game were used to train a classifier for recognizing the presence of stop signs in an image and estimate their distance. In [23] a different game was used for training a self-driving car: TORCS, an open source racing simulator with a graphics engine less focused on realism than GTA-V.

Authors in [24] created a dataset taking images from GTA-V and demonstrated that it is possible to reach excellent results on tasks such as real people tracking and pose estimation.

[25] also used GTA-V as the virtual world but, unlike our method, they used Faster-RCNN and they concentrated on vehicle detection validating their results on the KITTI dataset. Instead, [26] used a synthetically generated virtual dataset to train a simple convolutional network to detect objects belonging to various classes in a video.

III. Training set from virtual worlds

In this paper we show that a low cost and off-the-shelf virtual rendering environment represents a viable solution for generating a high quality training set for scenarios lacking enough real training data. This methods allows generating a very large amount of annotated images, with the possibility of scenery changes like location, contents, and even weather conditions, with very little human intervention.

In this work, we used the generated training set to train a *You Only Look Once* (YOLO) neural system [2], [3] for its efficiency and high detection accuracy. However, the applied methodology can be used to other machine learning tools.

We used the *Rockstar Advanced Game Engine* (RAGE) from the GTA-V computer game, and its scripting ability to deploy a series of pedestrians with and without safety equipment in different locations of the game map. The *RAGE Plugin Hook* [27] allowed us to create and inject our C# scripts into the game.

Our scripts uses the plugin API to add pedestrians with chosen equipment in various locations of the game map, place cameras in places where we want to take pictures, check that objects are in the field of view and not occluded, recover 3D meshes bounding boxes from the rendering engine, and save game screenshots (i.e., our dataset images) and their associated annotations (bounding boxes and classes).

Personal safety equipment that we consider are, for example, high-visibility vests, helmets, welding masks, and others. In addition to persons wearing these equipement, we also generate pedestrians without protections, where we annotate, person, bare head, bare chest (see Figure 2 as example).

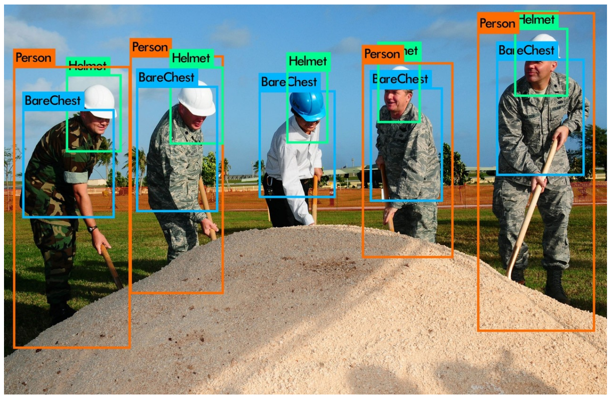

Fig. 2. Detected objects in a working scenario.

The generation of the virtual dataset required first to configure the RAGE engine to create various types of scenarios (Section III-A). Then, the RAGE engine was used to capture images along with annotations. For every image, the annotations (coordinates of the bounding boxes and identities of relevant elements) were retrieved from the RAGE engine through our script (Section III-B). We used this approach both for creating the virtual world training set and the virtual world validation set. The dataset was eventually completed by adding a real world test set, composed of real world images, to test the accuracy of the trained neural network on real scenes (Section III-C).

A. Scenario Creation

To generate the training scenario we used the plugin API to customize the following game features:

- **Camera**: used to set up the viewpoints from which the scenario must be recorded.
- **Pedestrians**: used to set up the number of people in the scene and their behavior, chosen from the set offered by the game engine, such as wandering around an area, chatting between themselves, fighting, and so on.
- **Place**: used to set up the place where the pedestrians will be generated; there is a series of game map preset places, plus user-defined locations identified by map coordinates.
- **Time**: used to set up the time of day during which the scene takes place.
- **Weather**: used to set up the weather conditions during the animation.

We used 9 different game map locations with 3 different weather conditions each to create the virtual training set. from these we acquired a total of 126900 images with an average of 12 persons per shot. The virtual validation set spans 1 location with 3 weather conditions, and consists of 350 images with an average of 12 persons each. Therefore, in the end, we have 30 different scenarios where virtual world images were extracted from.

B. Dataset Annotation

Dataset annotation is the process which creates the annotated images for the dataset. In our case, we annotate the

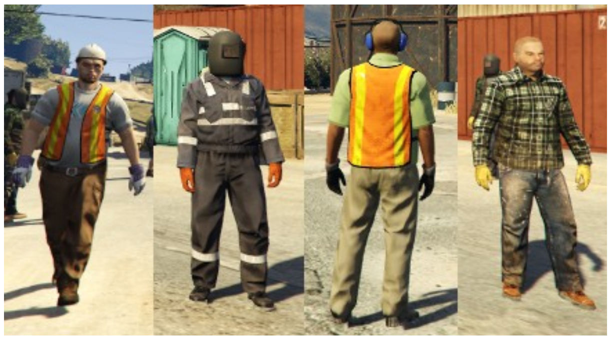

Fig. 3. Examples of safety equipment objects detected by our system.

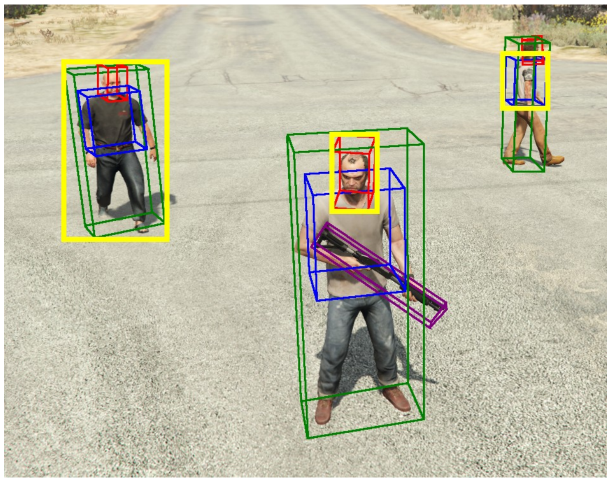

Fig. 4. Bounding box estimation: oversized approximation with respect to on-screen projections. With the available API, the hooked virtual engine is able to provide the bounding boxes of individual 3D meshes, overestimated due to collision proxy expansion for animations. Not being able to access the original 3D geometry and the current animation frame, our best strategy is to project on-screen the eight corners of the 3D bounding box, and then take their containing minimum rectangle as our best annotation.

following elements (see Figure 3):

- **Helmet**: a head wearing an helmet
- **Welding Mask**: a head wearing a welding mask
- **Ear Protection**: a head wearing hearing protection
- **High-Visibility Vest (HVV)**: person chest wearing a high visibility vest
- **Person**: a full-body person
- **Chest**: the bare chest (without HVV)
- **Head**: the bare head (without helmet or ear protection)

For each viewpoint setup in the scenario, we process every object to extract its position on the 2D image. This is done by first calculating the geometry of its transformed 3D bounding box, then approximately testing the box visibility, and finally extracting the image 2D bounding box by contouring the 3D box vertices. The visibility is checked by testing the occlusion of line-of-sight rays from the camera to a certain fixed amount of point in the box volume, and the object is considered visible if at least one ray is not occluded.

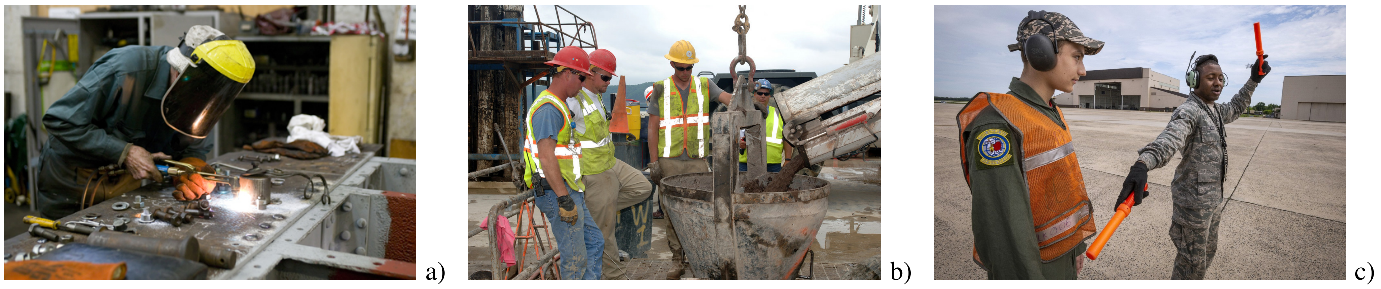

Fig. 5. Real-World Validation Set: composed of 180 copyright-free images, our validation set is available at the project website.

C. *Real world Test Set*

The motivation of this work is to prove that it is possible to train a system with a virtual world even when it supposed to be used in the real world. To test the performance of the trained neural network in the real world, we created a real world test set using copyright-free photographs of people wearing safety equipments. The set is composed of 180 images (see Figure 5) showing persons with and without the items listed in Section III-B, each associated with manually created annotations of bounding boxes and element identities.

IV. Method

The backbone of our detection algorithm is the *You Only Look Once* (YOLO) system [2], an efficient neural network able to detect, in a single image, the objects which it has been trained for. The detection ends with a list of 2D bounding boxes, each associated with a class label referring to the recognized object. In our implementation, we used the YOLO *v3* [3] (hereafter abbreviated with YOLO) network with the *Darknet-53* in its its core. We trained it to recognize personal safety equipment components, as shown in the following.

A. *Transfer Learning*

We already said that we use the generated virtual world training set to train YOLO to detect and recognize our elements of interests in images. In particular, we adapt YOLO to our scenario using *transfer learning*. Our hypothesis is that a pre-trained network already embeds enough knowledge that allow us to specialize it for a new scenario, leveraging on the transfer learning capability of deep neural networks and on training sets generated from a virtual world.

The purpose of transfer learning is to exploit the first already trained layers (i.e., the ones identifying low-level features) and to extend the detection capability to the new set of objects by updating the last layers of the network.

With a trained deep convolutional neural network, its first layers have learned to identify features that are more and more complex according to layer depth; for example, the first layer will be able to detect straight borders, the second layer smooth contours, the third some kind of color gradients, and so on while arriving to last layers capable of identify entire objects.

In our case, we used a YOLO netowrk pre-trained on the COCO dataset. We retrained it by blocking the learning parameters update of the first part of the network, and allowing updates only in the last sections. Specifically, we kept the first 81 (i.e., the feature extractors) of the total 106 layers, and froze the weights of the first 74. The network was trained for 24000 iterations, that is 11 epochs with the following parameters: batch size 64, decay 0.0005, learning rate 0.001, momentum 0.9, IoU threshold: 0.5, Confidence threshold: 0.25. As explained in III, our virtual dataset is composed of 30 scenarios, 27 of which were used as the training set and 3 were left for validation. The 3 scenarios of the validation set contain 13500 images. From these, 350 images were randomly selected to form the virtual validation dataset.

In this way, a new set of objects are recognized by the network.

B. *Evaluation Metrics*

To evaluate the performance of our implementation, we applied the standard measures used in the object detection literature, i.e., *Intersection over Union* (IoU) based on the area of the detected (D) and real (V) bounding boxes, and *Precision* (Pr) and *Recall* (Rc) based on true (T) / false (F) positive (P) / negative (N) detections:

- $IoU = (D \cap V)/(D \cup V)$
- $Pr = TP/(TP + FP)$
- $Rc = TP/(TP + FN)$

Detected bounding boxes are associated with a confidence score, ranging from 0 to 1, and are considered in the output if and only if their confidence score is greater than a configurable threshold. Given the above definitions, we calculate the *mean Average Precision* (mAP) as the average of the maximum precision at different recall values.

V. Experiments and Results

A. *Experimental Setups*

We trained and evaluated three variations of the network on both virtual and real images: YCV, which is YOLO base trained on COCO and retrained with Virtual data; YCVR, which is YCV fine-tuned with Real data; YCR, which is YOLO base trained on COCO and retrained with Real data.

To obtain YCVR, we split the real world dataset in two parts with 100 images each: a training part and a testing part. We used the training part to apply domain adaptation from virtual to real on the YCV network. To choose the set of weights from which to start, we validated each of them on the training part,

TABLE I
mAP OF YCV TESTED ON OUR VIRTUAL WORLD DATASET.

Iter.	**Epochs**	**Head**	**Helmet**	**Weld. Mask**	**Ear Prot.**	**Chest**	**HVV**	**Person**	**mAP**
18k	8	89.56%	86.76%	73.23%	87.97%	90.36%	90.04%	89.32%	86.75%
19k	9	89.42%	81.65%	73.39%	88.68%	89.72%	90.43%	89.87%	86.17%
20k	9	87.02%	81.76%	72.61%	88.87%	89.45%	90.33%	89.54%	85.66%
21k	10	89.75%	86.74%	75.53%	89.05%	89.75%	90.09%	89.79%	**87.24%**
22k	10	89.83%	87.49%	74.17%	88.27%	89.41%	89.95%	89.83%	86.99%
23k	10	89.75%	86.02%	73.39%	88.47%	89.89%	90.03%	89.76%	86.76%
24k	11	89.03%	88.46%	73.44%	89.05%	90.08%	90.04%	89.95%	87,15%

TABLE II
mAP OF YCV TESTED ON COPYRIGHT-FREE REAL WORLD IMAGES.

Iter.	**Epochs**	**Head**	**Helmet**	**Weld. Mask**	**Ear Prot.**	**Chest**	**HVV**	**Person**	**mAP**
18k	8	29.99%	69.72%	22.68%	48.10%	49.98%	67.01%	77.51%	51.00%
19k	9	25.57%	61.07%	19.65%	34.58%	36.89%	66.60%	72.13%	45.21%
20k	9	36.25%	74.13%	27.31%	55.60%	45.66%	69.89%	76.91%	**55.11**%
21k	10	35.84%	69.18%	26.92%	47.41%	37.90%	66.70%	76.85%	51.54%
22k	10	26.31%	64.48%	22.65%	48.23%	43.54%	63.62%	74.93%	49.11%
23k	10	35.02%	68.48%	11.76%	56.57%	42.41%	65.02%	74.25%	50.51%
24k	11	33.71%	65.55%	27.11%	43.17%	37.78%	62.65%	73.31%	49.04%

choosing the one with highest mAP. We chose the weights after 20,000 iterations, with 59.05 mAP. The fine-tuning was done for 1000 iterations.

To better evaluate the benefit contributed by the virtual world training set, we also fine-tuned YOLO base, pre-trained on COCO, with the same 100 real images used for obtaining YVCR. As we said before we call YCR this network.

B. Results

YCV obtains 87.24 mAP on when tested on virtual images (see Table I). When testd on real world images it obtains 55.11 mAP (see Table III). Most of the AP loss is caused by the classes Head, Welding Mask, Ear Protection, and Chest. We believe that this is due to the fact that in real life there are many more variations of these object classes than those the game can render.

We want to note that on virtual world testing, YCV obtains its best mAP after 21000 iterations. On real world testing, best performance is reached after 20000 iterations. This implies that the best performing set of weights for the virtual world test is not the best also for real world validation.

YCVR obtains a significant boost and reaches 76.1 mAP. This means that fine-tuning with only 100 real images is very effective on a network which was previously fine-tuned with several similar virtual images. We also note that testing YCVR on the virtual world yields a lower mAP with respect to YCV. The main drop of AP in this case is seen on Head and Welding Mask, which are the classes with most differences between real and virtual.

YCR obtain 57.3 mAP when tested on the real world test set. This result is just slightly better than that obtained by YCV, and by far worse than YCVR. This means that the contribution given by the virtual world training set, to train the network for the new scenario is very relevant, and just a fine-tuning with a few images is enough to adapt the network back to the real world domain.

VI. Conclusions

Training deep neural networks in virtual environments has been recently proven to be of help when the number of available training examples for the specific task is low. In this work, we considered the task of learning to detect proper equipment in risky human activity scenarios.

We created and made available two datasets: the first one has been generated using a virtual reality engine (RAGE from GTA-V); the second one is composed of real photos.

In our experiments, we trained YOLO on the virtual dataset and tested on the real images as well as using just a small number of real photos to fine-tune the deep neural network we trained in the virtual environment. The experiments we conducted demonstrated that training on virtual world images, and executing a step of domain adaptation with a limited number of real images, is very effective. Obtained performance when training with virtual world images and adapting to the

TABLE III
mAP COMPARISON OF OUR NETWORKS ON VIRTUAL VALIDATION OR REAL TESTING

Network	Test	Head	Helmet	Weld. Mask	Ear Prot.	Chest	HVV	Person	mAP
YCV	V	89.7%	86.7%	75.5%	89.0%	89.7%	90.0%	89.7%	87.2%
YCV	R	36.3%	74.1%	27.3%	55.6%	45.7%	69.9%	76.9%	55.1%
YCR	R	44.1%	52.2%	42.3%	62.0%	59.1%	60.7%	80.6%	57.3%
YCVR	R	78.8%	73.3%	66.3%	74.0%	74.7%	78.6%	87.1%	76.1%

domain with a few real images is much higher than just fine tuning an existing network with a few real images for the scenario at hand. We plan to use the same virtual environment to train to detect people using weapons (see Figure 4).

ACKNOWLEDGEMENT

This work was partially supported by "Automatic Data and documents Analysis to enhance human-based processes" (ADA), funded by CUP CIPE D55F17000290009, and by the AI4EU project, funded by EC (H2020 - Contract n. 825619). We gratefully acknowledge the support of NVIDIA Corporation with the donation of a Jetson TX2 board used for this research.

REFERENCES

[1] Enrico Meloni, Marco Di Benedetto, Giuseppe Amato, Fabrizio Falchi, Claudio Gennaro, "Project Website," http://aimir.isti.cnr.it/vw-ppe, 2019.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[3] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018.

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 91–99.

[5] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, June 2009, pp. 248–255.

[6] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 740–755.

[7] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan 2015.

[8] A. Kuznetsova, H. Rom, N. Alldrin, J. R. R. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig, and V. Ferrari, "The open images dataset V4: unified image classification, object detection, and visual relationship detection at scale," *CoRR*, vol. abs/1811.00982, 2018.

[9] J. Marn, D. Vzquez, D. Gernimo, and A. M. Lpez, "Learning appearance in virtual scenarios for pedestrian detection," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2010, pp. 137–144.

[10] M. Aubry and B. C. Russell, "Understanding deep features with computer-generated imagery," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2875–2883.

[11] W. Qiu and A. Yuille, "Unrealcv: Connecting computer vision to unreal engine," in *European Conference on Computer Vision*. Springer, 2016, pp. 909–916.

[12] K.-T. Lai, C.-C. Lin, C.-Y. Kang, M.-E. Liao, and M.-S. Chen, "Vivid: Virtual environment for visual deep learning," in *Proceedings of the 26th ACM International Conference on Multimedia*, ser. MM '18. New York, NY, USA: ACM, 2018, pp. 1356–1359.

[13] Z.-W. Hong, C. Yu-Ming, S.-Y. Su, T.-Y. Shann, Y.-H. Chang, H.-K. Yang, B. H.-L. Ho, C.-C. Tu, Y.-C. Chang, T.-C. Hsiao *et al.*, "Virtual-to-real: Learning to control in visual semantic segmentation," *arXiv preprint arXiv:1802.00285*, 2018.

[14] W. Luo, P. Sun, F. Zhong, W. Liu, T. Zhang, and Y. Wang, "End-to-end active object tracking and its real-world deployment via reinforcement learning," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[15] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 969–977.

[16] A. Bewley, J. Rigley, Y. Liu, J. Hawke, R. Shen, V.-D. Lam, and A. Kendall, "Learning to drive from simulation without real world labels," *arXiv preprint arXiv:1812.03823*, 2018.

[17] D. Vzquez, A. M. Lopez, and D. Ponsa, "Unsupervised domain adaptation of virtual and real worlds for pedestrian detection," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, Nov 2012, pp. 3492–3495.

[18] D. Vzquez, A. M. Lpez, J. Marn, D. Ponsa, and D. Gernimo, "Virtual and real world adaptation for pedestrian detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 4, pp. 797–809, April 2014.

[19] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *European Conference on Computer Vision*. Springer, 2016, pp. 102–118.

[20] Rockstar Games, Inc., "Grand Theft Auto - V," https://www.rockstargames.com/V, 2013.

[21] M. Martinez, C. Sitawarin, K. Finch, L. Meincke, A. Yablonski, and A. Kornhauser, "Beyond grand theft auto v for training, testing and enhancing deep learning in self driving cars," *arXiv preprint arXiv:1712.01397*, 2017.

[22] A. Filipowicz, J. Liu, and A. Kornhauser, "Learning to recognize distance to stop signs using the virtual world of grand theft auto 5," Tech. Rep., 2017.

[23] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2722–2730.

[24] M. Fabbri, F. Lanzi, S. Calderara, A. Palazzi, R. Vezzani, and R. Cucchiara, "Learning to detect and track visible and occluded body joints in a virtual world," in *European Conference on Computer Vision (ECCV)*, 2018.

[25] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, and R. Vasudevan, "Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?" *CoRR*, vol. abs/1610.01983, 2016.

[26] E. Bochinski, V. Eiselein, and T. Sikora, "Training a convolutional neural network for multi-class object detection using solely virtual world data," in *Advanced Video and Signal Based Surveillance (AVSS), 2016 13th IEEE International Conference on*. IEEE, 2016, pp. 278–285.

[27] "RAGE Plugin Hook," https://ragepluginhook.net, 2013.