Realistic Artificial Datasets for Object Detection and Instance Segmentation

Eduin Hernandez
NYCU, Taiwan
Email: eduin.ee08@nycu.edu.tw
Student ID: 22104094

Matin Moradi
Sharif University of Technology, Iran
Email: matinmoradi110@gmail.com
Student ID: 22104108

Vahid Haratian
Bilkent University, Turkey
Email: vahid.haratian@bilkent.edu.tr
Student ID: 22101349

***Abstract*—In this paper, realistic annotated images are created from a simulator that replicates the real-world environment. This dataset are then used to train a benchmark architecture for object detection and instance segmentation task and evaluated on real annotated images. The performance of this model is then compared to a counterpart that has been trained on the real annotated images to prove whether this training method is sufficient for inference on real images. This methodology could circumvent the problem of not having the real annotated dataset as well as augmenting them.**

I. Introduction

We are currently in an age where video games and simulators have a very realistic rendering of the textures present in objects and similar, if not identical, physics reactions to interaction between objects and their environment. Games engines and simulators such as Unity [1], Unreal Engine [2], NVIDIA ISAAC [3], and CoppeliaSim [4] make it possible to simulate and improve upon reality. Many Artificial Intelligent (AI) researchers and developers take advantage of this concept: they train and test the models inside these engines and simulators to verify concepts, performance, and possible issues. Reinforcement Learning (RL), a sub-branch of AI, uses this concept to train the models that are later deployed on robots. To avoid breaking or damaging them, the robots and their environments are replicated inside the simulation and then trained. Once a satisfactory level of performance has been achieved, the model is then deployed and tested on the real robot.

However, how about using images and data from the simulator to train models that are tested on real life images? That is the goal of this paper where we generate and annotate images inside CoppeliaSim [4], our simulator of choice, and evaluate the trained model on their real images counter-part. If the performance does well, then we can remove the meticulous and slow task of having humans annotate the images by hand and speed-up the training pipeline. This is especially useful when the dataset is not readily available or insufficient.

The paper is divided as follows: In Sec. II, we provide the Deep Neural Network (DNN) model architecture, description, and task it will be performing; Sec. III, the dataset we will be using, modifying, and generating; Sec. IV, results and interpretations; Sec. V some discussion and limitations we faced; and finally, Sec. VI, our finally thoughts on this work.

II. Architecture

A. Mask R-CNN

Mask R-CNN [5] was first introduced in 2017 as a state-of-the-art instance segmentation framework. It is an extension of Faster R-CNN [6], adding a branch for predicting segmentation mask in the Regions of Interests (RoI). Fig. 1 shows an sample output of Mask-RCNN.

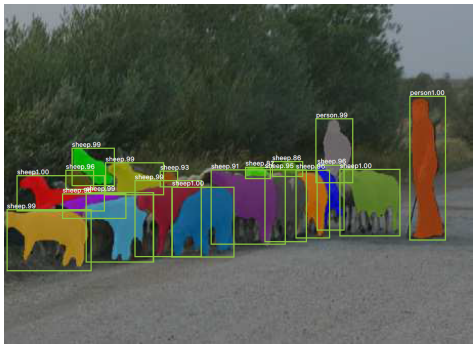


Fig. 1: Mask RCNN Output Example. Image from [5].

Mask-RCNN uses a backchain CNN to find the RoI and two front chains, one for labeling the object inside and one for computing a mask to find the object's exact location within the RoI. The backchain uses ResNet [7], a version of Residual CNN with skip passages between layers. It allows to construct a very deep neural networks even deeper than normal neural networks, then provide some shortcuts or skip passages to jump over one or more layers. In the backchain, Mask-RCNN also administers Feature Pyramid Network (FPN) [8], which is a CNN with a lateral chain to achieve better accuracy. FPN uses a classical bottom-up method to extract features with different scales. However, a top-down approach is administered to upsample features from higher stages on the lateral chain to achieve more accurate features. Results from the lateral chain are enhanced by using results from the bottom-up pathway with the same scale. Fig. 2 shows the structure of FPN stack.

The front chain uses the same chain proposed by FasterCNN for labeling the objects and separating CNN with 3*3 kernels for convolution and 2*2 for deconvolution to calculate masks for objects inside regions of interest. Fig. 3 shows an overview of the front chain architecture.

In the original paper, they compared their stack with C4 [7] with depths of 50 and 101, and they have shown that they

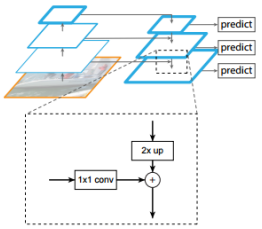


Fig. 2: Feature Pyramid Network. Image from [9].

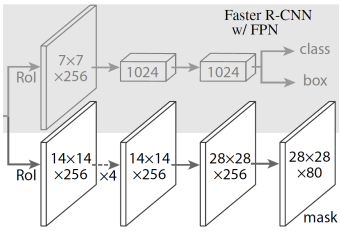


Fig. 3: Mask RCNN Front Chain. Image from [5].

can achieve a better result with Feature Pyramid Network. Also, they compared results for ResNet against ResNetXt [10], which yielded no significant improvement. This is why we will choose to use ResNet and FPN as the backchain/backbone in the experiments.

As for the loss, Mask R-CNN minimizes the following function for the sample RoI:

$$L = L_{cls} + L_{box} + L_{mask} \tag{1}$$

where L_{cls} is the log classification loss, L_{box} the true bounding box regression loss, and L_{mask} the average binary cross-entropy loss of the mask.

Note: Although the authors of Mask R-CNN provide their own official implementation, we chose to use the implementation provided by authors in [11] since it contains an object detection toolbox that already has the architectures and configurations of [5] along with many others. The purpose of this toolbox is to benchmark the many frameworks.

III. Dataset

In this section we will introduce the dataset we will be using for the simulations along with the modifications made for them along the way.

A. YCB

The YCB dataset [12] is a benchmark dataset for robotic manipulation. It provides a wide variety of common objects which can be categorized as follows: food, kitchen, tool, shape, and task items. In total, there are 77 different objects and 5 categories. Fig. 4 provides some visual example of these items on the food and tool categories.

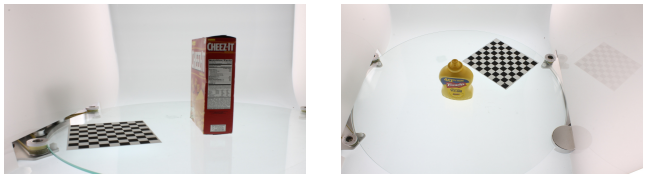
The authors from [12] provide the following for each object:

- 600 RGB Images of 4K resolution (JPG format)
- 600 RGB-D Images of 4K resolution with Depth Maps (JPG + H5 format)
- Segmentation Masks for each Image (PBM format)
- Turntable Pose Information for each Image (H5 format)
- Calibration Information for each RGB-D sensor (H5 format)
- Textured-map 3D mesh models (STL format)

We will be using the RGB images and their respective segmentation mask in Sec. III-B and the mesh models in Sec. III-C. Fig. 5 provides some examples of these RGB images. Note that these images are pictures of the same object rotated along a turntable with 5 cameras looking at it from various angles, providing 120 images each.

(a) Food Items (b) Tool Items

Fig. 4: Items from the YCB dataset. Images taken from [12]



(a) Cracker Box (b) Mustard

Fig. 5: Samples of RGB images provided in the YCB Dataset

B. YCB Modified (YCB-MOD)

We will be limiting the scope of our simulations to the following categories: food and tool items. In particular, we will be experimenting on three different sets, each containing the following items:

- Boxed and Canned Goods (B&C-MOD): Coffee Can, Cracker Box, Sugar Box, Soup Can, Mustard Bottle, Fish Can, Pudding Box, Gelatin Box, and Meat Can.
- Fruits (Fruit-MOD): Banana, Strawberry, Apple, Lemon, Peach, Pear, Orange, and Plum.
- Tools (Tool-MOD): Power Drill, Wood Block, Scissors, Large Marker, Flat Screwdriver, Hammer, and Extra Large Clamp.

Objects from the food categories were split into "Boxed and Canned Goods" and "Fruits" for a better training. Also, some items from the original categories were omitted due to either not having a proper mesh (i.e, key lock from the tool categories was distorted) or for having multiple instances of the same object with slight variation (i.e, medium clamp and large clamp are the same object in different sizes).

For the purpose of training, a total of N images are selected from a given category. From this N images, a N/K are taken for each item corresponding to that categories where K is the

number of classes belonging to that category, i.e., $K = 9$ for "Boxed and Canned Goods", $K = 8$ for "Fruits", and $K = 7$ for "Tools". Note that $N \leq K * 600$ and $N/K \leq 600$ since the maximum number of images provided per item is 600.

With the intention to create a COCO annotated format as in [13] from these RGB images to train a Mask R-CNN models, their corresponding binary segmentation masks are converted into a vector of the contour, bounding box, and area as required by the COCO format and stored as the annotation. Both the sampled images and annotation data are stored together in separate folder.

C. YCB Artificial (YCB-ART)

For the artificial dataset, we borrow the meshes of the objects used in the new categories chosen in Sec. III-B and pre-process them inside CoppeliaSim [4], a robotics and physics simulator, such that it can be manipulated by the physics engine and respond to other objects inside the simulation. The resultant coppeliaSim-compatible meshes with textures are shown in Figs. 6, 7, 8.

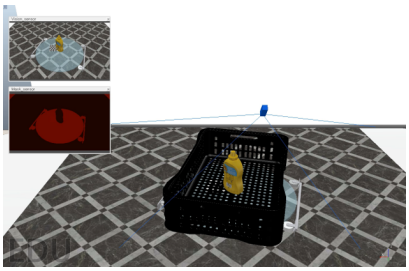
For generating diverse images and annotations, we borrow the technique used in [14] for domain randomization: random object postures, object quantity, spotlight posture, camera posture, environment light intensity, spotlight light intensity, and spotlight color. Additionally, for reasons we will further explain in Sec. IV, we will also employ random background textures. The original scene was replicated as much as possible. For each generated and annotated image, domain randomization is applied and the objects are dropped on top of the table and into the crate, the last of which used is to prevent it from going outside of camera's range, however the crate is set to not be rendered by the camera as shown in Fig. 9b. After the physics reaction have settled, two image are taken: one containing the RGB of the scene and one containing the mask segmentation of each object according to their simulation id. The images are then annotated according to the desired parameters: class labels and occlusion labels.

With the intention to imitate the original YCB images, we will only use a single item per image annotation, so unlike [14], object quantity is not randomized unless stated otherwise. Also, we will not use the occlusion labels as there is only one object present in the scene. A visualization of the scene generating the artificial dataset can be observed in Fig. 9 and the resultant artificial RGB images in Fig. 10.

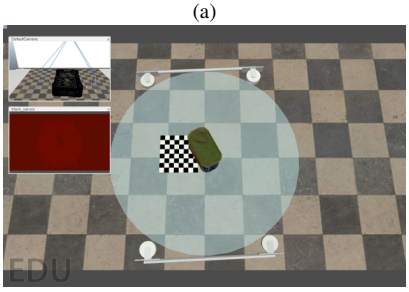
Fig. 6: Mesh models of boxed and canned goods objects inside CoppeliaSim.

Fig. 7: Mesh Models of fruit objects inside CoppeliaSim.

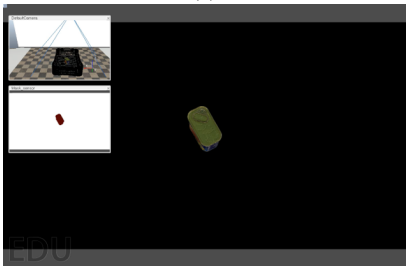
Fig. 8: Mesh Models of tool objects inside CoppeliaSim.



(a)



(b)



(c)

Fig. 9: Example of dataset generation process using CoppeliaSim from (a) the user's viewpoint, (b) the camera's viewpoint, (c) the camera's viewpoint rendering only the object to segment.

IV. Results

For our sets of results, we have used the Mask R-CNN architecture with ResNet-50 as backbone with FPN for the feature extraction. One of these models was trained on the YCB-MOD dataset and a separate one on the YCB-ART dataset. For the optimizer, SGD was employed with a learning

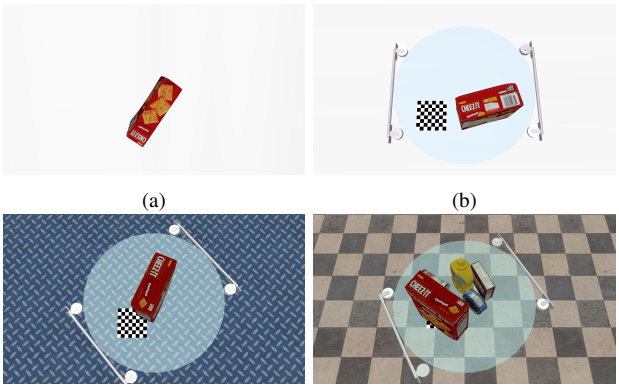


Fig. 10: Samples of RGB images generated in the simulation for B&C-ART and its various variations: (a) B&C-ART-BLAND, (b) B&C-ART-WHITE, (c) B&C-ART, and (d) B&C-ART-5s

rate of 0.02, momentum of 0.9, and weight decay of 10^{-4}. Each model was trained for a total of 12 epochs and evaluated by its log class loss, bounding box average precision (AP), and segmentation mask average precision (AP).

For YCB-MOD, a total of 1k images and annotations are sampled for the traning and 100 for the validation from the "Box and Canned Goods", "Fruits", and "Tools" and labeled "B&C-MOD", "Fruit-MOD", and "Tool-MOD" respectively. Note that this validation dataset will be shared for evaluation of the respective YCB-ART trained model. For YCB-ART, a total of 1k images and annotations are generated for the training and 100 for the validation from the "Box and Canned Goods", "Fruits", and "Tools" and labeled "B&C-ART", "Fruit-ART", "Tool-ART" respectively. We also provide a hybrid dataset from using 900 images and annotations from B&C-ART and 100 images and annotations from B&C-MOD which we label "B&C-HYB".

Additionally, we create variations for B&C-ART, denoting them with the following suffix labels:

- 5s: Randomizing the number of object that appears in the image from 1 to 5 samples, as in Fig. 10d.
- 5k: 5k images are generated and annotated.
- BLAND: No background object nor textures are used in the image, as in 10a.
- WHITE: No texture background are used in the image, as in Fig. 10b.

And also provide tunning in the hyper-parameters of the training and architecture of the model, denoting them with the following suffix labels:

- r50: ResNet-50 used as backbone architecture (Default backbone).
- r101: ResNet-101 used as backbone architecture.
- ADAM: Adam optimizer with learning rate 0.003 and weight decay of 10^{-4}.

When modifying the RGB YCB images in Sec. III-B and generating the artificial Dataset in Sec. III-C, we scale down the image resolution to 1080p to avoid overloading the memory and speeding up the training of Mask R-CNN.

Dataset	Bounding Box			Segmentation Box		
Train Dataset	AP	AP_{50}	AP_{75}	AP	AP_{50}	AP_{75}
B&C-MOD	0.951	0.995	0.995	0.955	0.955	0.995
B&C-HYB	0.862	0.979	0.973	0.885	0.979	0.978
B&C-ART-5s	0.633	0.919	0.733	0.642	0.922	0.757
B&C-ART	0.579	0.867	0.658	0.613	0.874	0.874
B&C-ART-WHITE	0.433	0.650	0.518	0.475	0.642	0.561
B&C-ART-BLAND	0.431	0.634	0.534	0.475	0.642	0.561
B&C-ART-5K	0.324	0.490	0.376	0.369	0.491	0.456
Fruit-MOD	0.957	0.996	0.996	0.956	0.996	0.996
Fruit-Art	0.551	0.976	0.556	0.562	0.976	0.589
Tool-MOD	0.859	0.967	0.949	0.822	0.965	0.940
Tool-ART	0.200	0.416	0.196	0.241	0.443	0.248

Table I: Test Average Precision (AP) for various datasets and their modifications on the same architecture model. Evaluation is carried out on B&C-MOD-VAL, Fruit-MOD-VAL, and Tool-MOD-VAL accordingly. Ranked in descending order according to their Bounding Box AP.

The inference results of these trained models can be observed in Figs. 11, 12 and Tabs. I, II. As results in Tab. I show, the best model for inferencing real images are, unsurprisingly, those trained with real images. However, B&C-HYB shows that even with only 10% of real images and 90% of artificial, we can still achieve a sufficient enough performance, having only a 10% drop. Contrary to the hybrid case, there is a significant loss in the inference performance of the artificial case: As shown in Fig. 11b, the model creates multiple detections on the same object and misclassifications.

If background noise is not used, such as the turntable, chessmat, or background texture, the model is more likely to mis-detect background objects, hence the difference in the performance of B&C-ART-BLAND, B&C-ART-WHITE, and B&C-ART. Including more object samples per images slightly improves the performance as shown by B&C-ART-5s, however, including more images to train on significantly overfits the model on the artificial data, i.e, B&C-ART-5k. Increasing the model depth, like with B&C-ART-r101, slightly improve the inference performance, but it is debatable whether that slight increase is worth using a larger network. As for using different optimizers, like Adam, there was no particular benefit in changing optimizer, rather, it turned out to be harmful for the inference on real images.

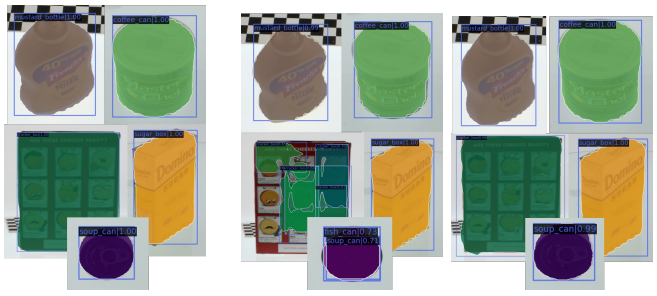
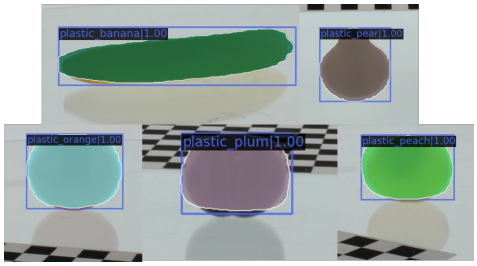
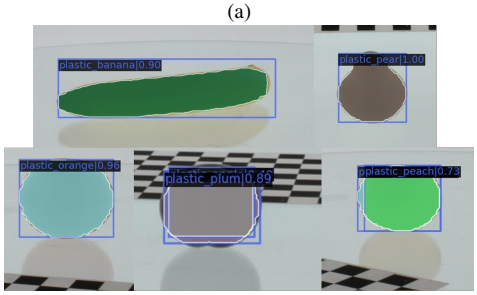


Fig. 11: Inference results of Mask R-CNN trained on (a) B&C-MOD, (b) B&C-ART, (b) B&C-HYB images and tested on a B&C-MOD-VAL image.



(a)



(b)

Fig. 12: Inference results of Mask R-CNN trained on (a) Fruit-MOD and (b) Fruit-ART images and tested on a Fruit-MOD-VAL image.

Dataset	Bounding Box			Segmentation Box		
Train Dataset	AP	AP_{50}	AP_{75}	AP	AP_{50}	AP_{75}
B&C-ART-r101	0.584	0.877	0.654	0.627	0.885	0.744
B&C-ART-r50	0.579	0.867	0.658	0.613	0.874	0.874
B&C-ART-ADAM	0.046	0.104	0.032	0.049	0.111	0.040

Table II: Test Average Precision (AP) for B&C on different architectures and training parameters. Evaluation is carried out on B&C-MOD-VAL. Ranked in descending order according to their Bounding Box AP.

V. Discussions and Limitations

The experiments faced many issues and limitations. In terms of the dataset, part of the multi-classification for a single instance was caused by the discrepancy of the Cracker box mesh model and Cracker box used for real images: the Cracker box that was scanned to create the mesh model had a design change, with the back surface texture no longer matching the one from the image. Since the texture does not match and only one mesh model is used per class label, this proves to be insufficient for the inference. The model could achieve a better performance if a higher intra-class variation on the texture could be provided, this could prove especially useful if the products go through constant design change.

In terms of training with YCB-ART, there were often issues with exploding and vanishing gradients when modifying the default training parameters such as the training batch size, epoch number, optimizer, and even the dataset size. A small batch size of 1 and, depending on the dataset, 2 often cause the gradient to explode, so training it with batch sizes of 3 or 4 provided a more stable learning. However, by increasing the batch size, more memory would be required without any guarantee of a better performance for the model. Increasing the training epoch did not improve the inference performance, rather, the gradients would often explode after reaching epoch 13 or 14 for reasons we have yet to understand. Changing the optimizer to Adam or dataset size to 5 times its original size often lead to issues in training, with the model not always been able to learn the train-set. The results for using 5x the original dataset size or Adam provided in Tab. I and II were for the successful training cases, despite their poor inference performance.

In terms of the DNN models, our original intent was to work with DetectoRS [9], a more recent and better performing architecture in comparison to Mask R-CNN which is also available in [11], with an additional requirement for the dataset: an image containing the segmentation masks for all the object and background in the corresponding RGB image. Although we were able to provide this requirements, the implementation of DetectoRS in [11] would not allow us to initiate the training since there were some errors with the "gpu" which we were unable to resolve.

VI. Conclusion

We have seen the results of training an object detection and image segmentation model on artificial images, however there are improvements to be made on these images so the training on these artifical dataset can generalize well enough to be evaluated on real images. It is not a robust method, as finetunning the simulator environment is necessary to improve the dataset used for training DNNs. There is also need to increase the intra-class texture variation for the mesh models as well as including more background variation so that the DNNs model can becomes invariant to slight texture change and background noise. However, if small a dataset is already available, the hybrid dataset generation seems to be a promising method for augmentation.

References

[1] J. K. Haas, "A history of the unity game engine," 2014.
[2] Epic Games, "Unreal engine." [Online]. Available: https://www.unrealengine.com
[3] J. Liang, V. Makoviychuk, A. Handa, N. Chentanez, M. Macklin, and D. Fox, "Gpu-accelerated robotic simulation for distributed reinforcement learning," in *Conference on Robot Learning*. PMLR, 2018, pp. 270–282.
[4] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1321–1326.
[5] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969. [Online]. Available: https://github.com/matterport/Mask_RCNN
[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.
[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[8] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.
[9] S. Qiao, L.-C. Chen, and A. Yuille, "Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 213–10 224. [Online]. Available: https://github.com/joe-siyuan-qiao/DetectoRS

[10] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 761–769.

[11] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "MMDetection: Open mmlab detection toolbox and benchmark," *arXiv preprint arXiv:1906.07155*, 2019. [Online]. Available: https://github.com/open-mmlab/mmdetection

[12] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The ycb object and model set: Towards common benchmarks for manipulation research," in *2015 international conference on advanced robotics (ICAR)*. IEEE, 2015, pp. 510–517. [Online]. Available: https://www.ycbbenchmarks.com/

[13] 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 *European conference on computer vision*. Springer, 2014, pp. 740–755.

[14] C.-H. L. Tsai, J.-Y. J. Chang, E. Hernandez, and X.-W. You, "A new approach to enhance artificial intelligence for robot picking system using realistic digital twins tool," *International Journal of iRobotics*, vol. 4, no. 3, pp. 1–6, 2021.