Balancing 3D-model fidelity for training a vehicle detector on simulated data

Thijs A. Eker[a], Ella P. Fokkinga[a], Friso G. Heslinga[a], and Klamer Schutte[a]

[a]TNO - Intelligent Imaging, Oude Waalsdorperweg 63, the Hague, the Netherlands

ABSTRACT

The use of simulated data for training deep learning models has shown to be a promising strategy for automated situational awareness, particularly when real data is scarce. Such simulated datasets are important in fields where access to environments or objects of interest is limited, including space, security, and defense. When simulating a dataset for training of a vehicle detector using 3D models, one ideally has access to high-fidelity models for each class of interest. In practice, 3D model quality can vary significantly across classes, often due to different data source or limited detail available for certain objects. In this study, we investigate the impact of this 3D model variation on the performance of a fine-grained military vehicle detector, that distinguishes 15 classes and is trained on simulated data. Our research is driven by the observation that variations in polygon count among 3D models significantly influence class-specific accuracies, leading to imbalances in overall model performance. To address this, we implemented four decimation strategies aimed at standardizing the polygon count across different models. While these approaches resulted in a reduction of overall accuracy, measured in average precision (AP) and AP@50, they also contributed to a more balanced confusion matrix, reducing class prediction bias. Our findings suggest that rather than uniformly lowering the detail level of all models, future work should focus on enhancing the detail in low-polygon models to achieve a more effective and balanced detection performance.

Keywords: Simulated data; 3D models; Object detection; Model fidelity

1. INTRODUCTION

Object detection models in high-stakes fields, such as the military domain, are required to have exceptional accuracy and reliability. Deep learning-based[1] detection methods are known to depend on vast amounts of real-world training data, which is costly and challenging to collect.

An effective alternative for model development is the use of simulated data. Recent work has shown the potential for several object detection applications, including military vehicles,[2] maritime vessels,[3] and persons.[4] The benefits of simulated training data have been studies for several imaging modalities, including natural images,[5] infrared,[6] radar,[7] and sonar imagery.[8]

A key method for creating synthetic training data is to simulate scenarios where 3D models of the objects are placed in a scenario of interest. This offers a controlled, scalable, and secure means of dataset creation, assuming representative 3D models can be produced. However, synthetic data has its drawbacks, primarily the 'synthetic-to-real gap'. This gap pertains to the differences between real and synthetic images (mainly in content, texture, and environmental conditions). As a result, models trained on synthetic datasets often fail to transfer effectively to real-world scenarios.[9]

Our research focuses on the development and enhancement of a military vehicle detection model using a simulated dataset derived from 3D models of 15 military vehicles. Previous iterations of our research demonstrated that while simulated data facilitates effective model training,[10] the performance across different vehicle classes was inconsistent. Some classes achieved as high as 85% classification accuracy, whereas others were never identified correctly. This disparity prompted a detailed analysis of the factors influencing model performance,

Artificial Intelligence for Security and Defence Applications II, edited by Henri Bouma, Radhakrishna Prabhu, Yitzhak Yitzhaky, Hugo J. Kuijf, Proc. of SPIE Vol. 13206, 132060O · © 2024 SPIE
0277-786X · doi: 10.1117/12.3031691

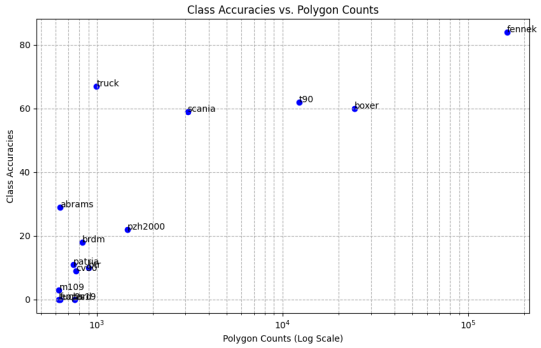


Figure 1: Average class accuracy over 6 runs relative to 3D model polygon count.

revealing a strong correlation between the polygon count of the 3D models and the corresponding class accuracies, as shown in Figure 1. For an example of a low and high polygon count 3D model see the no decimation column in Figure 3.

To address these inconsistencies, we hypothesized that normalizing the level of detail across all classes by standardizing the polygon count of the 3D models would mitigate the skew in class accuracies. Consequently, we implemented a decimation process to adjust the polygon count of each model to approximately 1,000 polygons. This paper presents our methodology, the adjustments made to the simulated dataset, and the impact on the performance of the object detection model. By balancing the details represented in each vehicle class, we ensure a more uniform accuracy across different classes and we gain insight into the effect of variation in 3D-model fidelity. Although we do not improve overall detection performance, a more uniform accuracy can improve robustness and reliability of the system in operational settings.

2. METHODS

In this section, we describe how we standardize the level of detail across a set of 3D military vehicle models used for generating a dataset. Each 3D model undergoes two main stages: initially, we apply a silhouette-based refinement step, followed by our decimation strategy. Furthermore we demonstrate the steps involved in training the model and generating the synthetic dataset.

2.1 Silhouette-based 3D model refinement

To begin, we remove fine details, such as antennas and smaller mirrors, that are not consistently present across all 3D models. This process involves generating three silhouettes of the 3D model, one from each principal axis perspective (X, Y, and X). Next, we refine these silhouettes using a morphological opening operation, as illustrated on the left side in Fig. 2. The resulting silhouettes are shown on the right side in Fig. 2, projected around the original 3D model. Finally, any vertices from the original 3D model that lie outside these processed silhouettes are eliminated. Removing these vertices creates small holes in the 3D model's surface, which are in the next step automatically repaired as a by-product of our decimation strategy

2.2 Decimation strategy

After refining the silhouette to remove small details, we apply a decimation pipeline to reduce the complexity of our 3D models to approximately 1,000 polygons. The threshold was chosen based on the observed correlation between class accuracy and polygon count of the 3D models, as visualized in Fig. 1. We hypothesize that reducing the polygon count to 10^3 will create a better balance in model complexity, potentially improving overall accuracy or at least reducing the class prediction imbalance. Unlike standard decimation methods that often retain fine details like mirrors or antennas, our pipeline is designed to preserve the primary shape of the object.

Send correspondence to T.A. Eker: thijs.eker@gmail.com

Figure 2: Left: The silhouette over the X-axis of the 3D model before and after the opening operation. Right: The complete model with refined silhouettes, vertices outside the silhouette boundaries will be removed.

The decimation process begins with two initial steps that streamline the later steps. First, we selectively remove polygons that form angles below a 5-degree threshold, simplifying the model while maintaining its overall form. Second, we enhance the structure by adding a minimal thickness to the geometry. This step prevents the formation of holes and preserves the integrity of the model's surface during the remeshing.

Next, the model is remeshed to ensure a uniform level of detail across all models. This process recalculates the mesh based on a voxel grid that intersects with the original geometry. The voxel size controls the level of detail: larger voxel sizes result in retention of less detail. In our study, we use two different remeshing settings, referred to as *light'* and *heavy' decimation*. For *light decimation*, we use a voxel size of 0.05 with a minimal thickness of 0.07, while for *heavy decimation*, we use a voxel size of 0.1 with a minimal thickness of 0.13.

Finally, we proceed with decimation of the model by progressively removing polygons that form smaller angles, using an increasingly strict threshold beyond the initial 5-degree threshold, until the polygon count is reduced to approximately 1,000.

In addition to the *light* and *heavy decimation* strategies, we apply combined decimation experiments: *combined (light)* and *combined (heavy)*. In this hybrid approach, we only decimate models that initially had more than 10^3 polygons (Fig. 1). Those with a lower polygon count remain unaltered. We hypothesize that this combined approach maintains the balance between outlier reduction in polygon count, while preserving accuracy, as some models are more sensitive to decimation, due to their initial polygon count.

2.3 Dataset generation and model training

Following the described decimation strategy, we generate a dataset following the procedure described in previous work.[5] Using the Blender 3D graphics software, we place the 3D model at the center of a High Dynamic Range Imaging (HDRI) scene. Realistic camera positions and model orientations are sampled randomly within predefined ranges and the images are rendered, with corresponding object masks automatically generated as annotations. In this case, we used 100 unique HDRI scenes and generated 400 images per class, for a total of 6000 images per dataset.

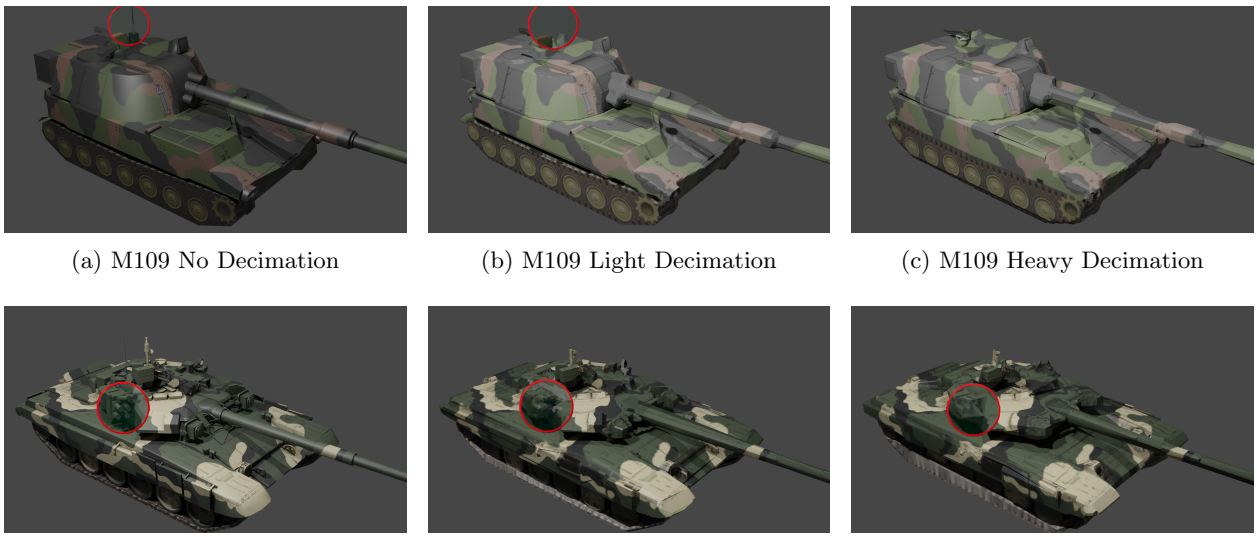
We adopted the model architectures and training strategies outlined in.[10] We utilized object detection model implementations found in the MMdetection framework.[11] The following data augmentations were applied: photometric distortions, horizontal flipping, cropping, and scaling. The models used for fine-tuning on our simulated datasets were a Mask R-CNN[12] with a SWIN-T transformer backbone,[13] pretrained on the COCO dataset,[14] and MM-Grounding-DINO,[15] pre-trained on various datasets.

To evaluate model performance, the mean average precision (mAP) and the AP at an intersection over union (IoU) threshold of 0.5 (AP@50) are computed. Both are popular metrics for object detection tasks and combine

classification accuracy with localisation accuracy. The latter, AP@50, is more focused on classification accuracy than on localisation accuracy. To gain insights into the class prediction balance, the normalized confusion matrices are computed for each experiment. Every experiment is repeated six times to obtain a standard deviation of the performance metrics and to reduce variance in the confusion matrices.

3. RESULTS

Figure 3 shows examples of the decimation effects on two different models: the M109 and the T90, for which the initial models consist of 625 and 12261 polygons respectively (see also Fig. 1). The M109 model, being simpler, is less affected by the decimation process. It primarily removes elements such as antennas, highlighted by red circles in the images. In contrast, the T90 model, which is more complex, exhibits a more significant reduction in detail. In Fig. 3, the smoke grenade launchers are highlighted across the decimation settings. Initially, these launchers appear as distinct components, but they gradually transform into a simple cube with only a textured surface as decimation increases, similar to the simpler details observed in the initial M109 model.



(a) M109 No Decimation (b) M109 Light Decimation (c) M109 Heavy Decimation

(d) T90 No Decimation (e) T90 Light Decimation (f) T90 Heavy Decimation

Figure 3: Comparative visualization of M109 and T90 models under different decimation settings. The top row shows the M109 model from no decimation to light and heavy decimation settings, while the bottom row shows the T90 model with similar variations. The red circles in the top row show that antennas are removed in the decimation step of the M109. In the bottom row, the smoke grenade launchers of the T90 are highlighted, which are gradually simplified in the decimation steps until only a cube with a textured surface remains.

In Table 1 the results for the experiments conducted using the Mask R-CNN and Grounding DINO models are presented. The table reports the mAP and AP@50 for each model under different decimation settings: no decimation (the initial 3D models), *light decimation*, *heavy decimation*, and combined decimation approaches (*light* and *heavy*).

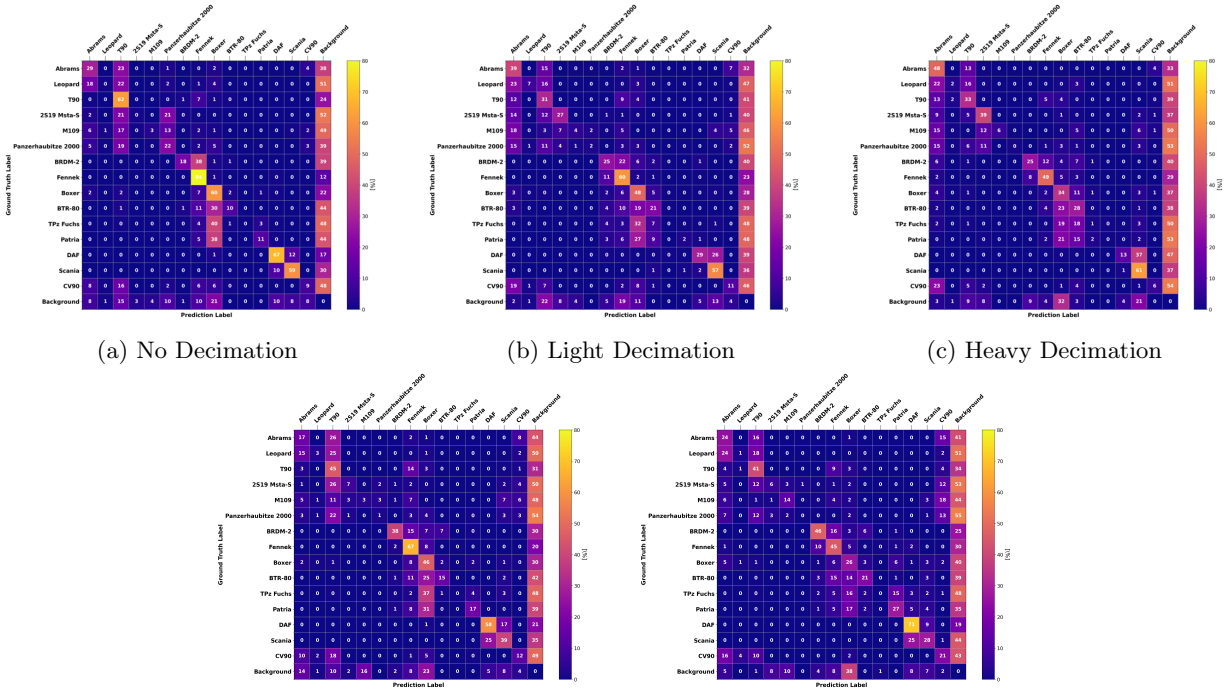
We observe that for four decimation approaches, the mAP and AP50 decrease with respect to the models fine-tuned on datasets generated based on the initial 3D models. For the grounding DINO model, the least performance loss is observed for the *combined (light)* method, where only models with a high polygon count ($> 10^3$) were altered.

Figure 4 shows the confusion matrices from the Grounding DINO experiments. In the original confusion matrix, several classes, particularly the *T90*, *Fennek*, *Boxer*, *Scania*, and the *DAF* truck, have a very high prediction rate. Four of these classes correspond to those with higher polygon counts, as indicated in the accuracy versus polygon count plot in 4. Although all decimation strategies result in a decrease in overall accuracy, the confusion matrices appear to become more balanced. This is especially true for the *combined (light)* decimation

Table 1: Results for Grounding DINO and Mask RCNN. All experiments were run six times to compute the standard deviation.

Experiment	Grounding DINO		Mask R-CNN	
	mAP	AP@50	mAP	AP@50
No decimation	35.1 ± 2.9	38.9 ± 3.1	34.9 ± 0.8	44.0 ± 0.9
Light decimation	30.8 ± 2.6	34.7 ± 2.8	31.8 ± 0.4	40.1 ± 0.4
Combined (light)	32.2 ± 3.2	35.7 ± 3.6	31.3 ± 1.0	38.4 ± 1.1
Heavy decimation	29.6 ± 1.7	33.1 ± 1.9	29.6 ± 0.2	37.8 ± 0.2
Combined (heavy)	31.0 ± 2.7	34.7 ± 2.8	28.6 ± 1.4	35.9 ± 1.6

(Fig. 4d), which also shows the smallest decrease in AP. The *heavy* decimation strategies further appear to balance the confusion matrix, but also come with a greater reduction in AP. The overall decrease in performance can be explained by the increase in number of objects predicted as background and the fact that although there are less false positives predicted for the highly detailed 3D models, the number of true positives also increases.



(a) No Decimation (b) Light Decimation (c) Heavy Decimation

(d) Combined Light Decimation (e) Combined Heavy Decimation

Figure 4: Confusion matrices for the grounding DINO experiments, averaged over the six runs.

In Figure 5, we visualize how the class accuracy relative to polygon count changes after *combined (light)* decimation. The models with initially $> 10^3$ polygons decrease in both accuracy and naturally, polygon count. However, for the models in the lower left corner, so, those with initially a low accuracy and a low polygon count, we observe an increase in accuracy.

4. DISCUSSION

In this study, we investigated the impact of 3D model fidelity, specifically in the level of detail, on the performance of DL-based object detection models trained on simulated data. This investigation was prompted by the observation that variations in polygon count among the 3D models used for the simulation correlate strongly with the class-specific accuracies, and thus influence the overall accuracy of the model. To address this, we

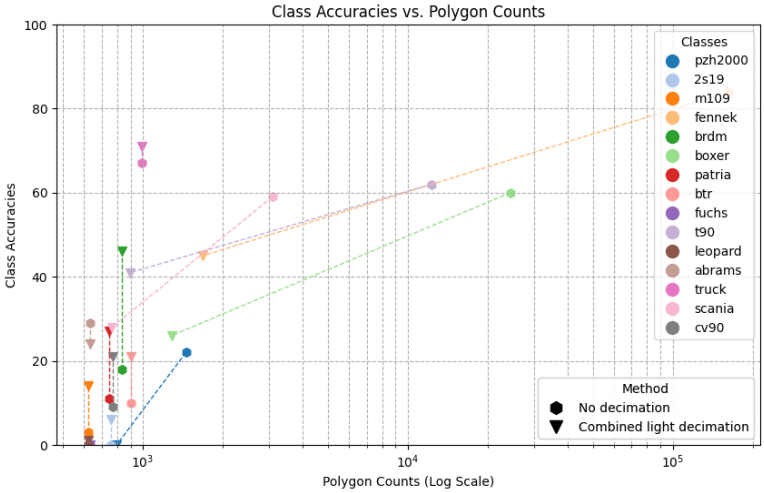


Figure 5: Change in average class accuracy versus polygon count, from no decimation (initial 3D models) to after *combined (light) decimation.* for the Grounding DINO model. Although in the upper right corner (high polygon count and accuracy), the accuracy decreases after decimation, in the lower left corner (low polygon count and accuracy) the accuracy increases after decimation.

implemented four decimation strategies aimed at standardizing the polygon count across different models. While all decimation approaches resulted in a reduction of overall accuracy - measured in mAP and AP@50 - they also contributed to a more balanced confusion matrix. This indicates a reduction in class prediction bias. The balance was most effectively achieved with the *combined (light)* decimation strategy, which maintained relatively the highest accuracy while reducing the class prediction imbalance caused by the classes with initially much higher polygon counts.

Our original hypothesis was that reducing the polygon count to a uniform level across all classes would mitigate the imbalance in class prediction and in this way, increase overall accuracy. However, our results indicate that while reducing the polygon count does decrease the prediction frequency for those classes, it also leads to a noticeable drop in average precision AP. These findings are supported by Figure 5, which shows that although class accuracies are converging, the overall accuracy decreases. This suggests that lowering the detail level to match that of less frequently predicted classes does not solve the problem; in fact, these models lack sufficient detail to effectively train a detector. Rather than attempting to reach a uniform level of detail by reducing the complexity of high-detail models, a more effective approach might be to enhance the low-detail models, bringing them up to the level of the high-detail models. This approach would ensure that all models possess the necessary detail to contribute meaningfully to the training process. Furthermore, the results implicate that polygon count might not be the decisive factor in determining model quality. Other aspects, such as texture quality or the accuracy of the 3D model, might play a more significant role in influencing prediction performance.

While equalizing the level of detail is necessary, the goal should be to elevate all models to a higher fidelity rather than reducing detail. Research consistently shows that models trained on highly detailed and realistic synthetic data are better equipped to generalize to real-world scenarios, reducing the domain gap between simulated and real environments.[16–19] Future work should focus on enhancing the level of detail of 3D models and thereby the quality of the simulated data. This could be achieved through the use of generative AI methods, such as inpainting and controllable diffusion, which have shown promising results for creating more realistic and detailed simulated data.[20–23]

REFERENCES

[1] Lecun, Y., Bengio, Y., and Hinton, G., "Deep learning," *Nature* **521**, 436–444 (2015).

[2] Ruis, F. A., Liezenga, A. M., Heslinga, F. G., Ballan, L., Eker, T. A., den Hollander, R. J., van Leeuwen, M. C., Dijk, J., and Huizinga, W., "Improving object detector training on synthetic data by starting with a strong baseline methodology," in [*Synthetic Data for Artificial Intelligence and Machine Learning: Tools, Techniques, and Applications II*], **13035**, 333–345, SPIE (2024).

[3] Westlake, S. T., Volonakis, T. N., Jackman, J., James, D. B., and Sherriff, A., "Deep learning for automatic target recognition with real and synthetic infrared maritime imagery," in [*Artificial intelligence and machine learning in defense applications II*], **11543**, 41–53, SPIE (2020).

[4] Spell, G. P., Tran, M., Torrione, P., Jeiran, M., Bahhur, B., and Manser, K., "Integration of synthetic data into real world computer vision pipelines," in [*Synthetic Data for Artificial Intelligence and Machine Learning: Tools, Techniques, and Applications II*], **13035**, 275–283, SPIE (2024).

[5] Eker, T. A., Heslinga, F. G., Ballan, L., den Hollander, R. J., and Schutte, K., "The effect of simulation variety on a deep learning-based military vehicle detector," in [*Artificial Intelligence for Security and Defence Applications*], **12742**, 183–196, SPIE Sensors + Imaging (2023).

[6] Vo, D. T., Duc, P. A., Thao, N. N., and Ninh, H., "An approach to synthesize thermal infrared ship images," in [*Synthetic Data for Computer Vision Workshop - CVPR*], (2024).

[7] Heslinga, F. G., Uysal, F., van Rooij, S. B., Berberich, S., and Caro Cuenca, M., "Few-shot learning for satellite characterisation from synthetic inverse synthetic aperture radar images," *IET Radar, Sonar & Navigation* **18**(4), 649–656 (2024).

[8] McKenzie, K., Jacobs, E., Ramirez, A., Conroy, J., and Watson, T., "Acoustic sensing on multi-rotor uav for target detection using a convolutional neural network," in [*Synthetic Data for Artificial Intelligence and Machine Learning: Tools, Techniques, and Applications II*], **13035**, 1303504, SPIE (2024).

[9] Man, K. and Chahl, J., "A review of synthetic image data and its use in computer vision," *Journal of Imaging* **8**(11), 310 (2022).

[10] Heslinga, F. G., Eker, T. A., Fokkinga, E. P., van Woerden, J. E., Ruis, F. A., den Hollander, R. J., and Schutte, K., "Combining simulated data, foundation models, and few real samples for training object detectors," in [*Synthetic Data for Artificial Intelligence and Machine Learning: Tools, Techniques, and Applications II*], **13035**, 44–55, SPIE (2024).

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

[12] He, K., Gkioxari, G., Dollár, P., and Girshick, R., "Mask r-cnn," in [*2017 IEEE International Conference on Computer Vision (ICCV)*], 2980–2988 (2017).

[13] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B., "Swin transformer: Hierarchical vision transformer using shifted windows," in [*Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*], (2021).

[14] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L., "Microsoft COCO: Common objects in context," in [*Computer Vision – ECCV 2014*], Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., eds., 740–755 (2014).

[15] Zhao, X., Chen, Y., Xu, S., Li, X., Wang, X., Li, Y., and Huang, H., "An open and comprehensive pipeline for unified object grounding and detection," *arXiv preprint arXiv:2401.02361* (2024).

[16] Chang, H.-Y., Chen, P.-Y., Chou, T.-H., Kao, C.-S., Yu, H.-Y., Lin, Y.-T., and Chen, Y.-N., "A survey of data synthesis approaches," *arXiv preprint arXiv:2407.03672* (2024).

[17] Schimanski, T., Ni, J., Kraus, M., Ash, E., and Leippold, M., "Towards faithful and robust llm specialists for evidence-based question-answering," *arXiv preprint arXiv:2402.08277* (2024).

[18] Hu, X., Li, S., Huang, T., Tang, B., Huai, R., and Chen, L., "How simulation helps autonomous driving: A survey of sim2real, digital twins, and parallel intelligence," *IEEE Transactions on Intelligent Vehicles* (2023).

[19] Alkhalifah, T., Wang, H., and Ovcharenko, O., "Mlreal: Bridging the gap between training on synthetic data and real data applications in machine learning," *Artificial Intelligence in Geosciences* **3**, 101–114 (2022).

[20] Voetman, R., Aghaei, M., and Dijkstra, K., "The big data myth: Using diffusion models for dataset generation to train deep detection models," *arXiv preprint arXiv:2306.09762* (2023).
[21] Yildirim, A. B., Baday, V., Erdem, E., Erdem, A., and Dundar, A., "Inst-inpaint: Instructing to remove objects with diffusion models," (2023).
[22] Fang, H., Han, B., Zhang, S., Zhou, S., Hu, C., and Ye, W.-M., "Data augmentation for object detection via controllable diffusion models," in [*Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*], 1257–1266 (2024).
[23] Zhang, C., Yang, W., Li, X., and Han, H., "Mmginpainting: Multi-modality guided image inpainting based on diffusion models," *IEEE Transactions on Multimedia* (2024).