Odd-One-Out: Anomaly Detection by Comparing with Neighbors

Supplementary Material

This supplementary is structured as follows: we present additional details of the proposed datasets in Appendix A, training details in Appendix B, and additional results in Appendix C.

A. Data Generation Details

Our proposed scene AD datasets, *ToysAD-8K* and *PartsAD-15K*, are built upon two publicly available 3D shape datasets: Toys4K [6] (Creative Commons and royalty-free licenses) and ABC [1] (MIT license). For the ToysAD-8K, we selected 1,050 shapes from the Toys4K dataset, focusing on the most common real-world objects across 51 categories. A complete list of these categories is provided in Fig. A2, and the split of seen/unseen categories is shown in Tab. A1. On the other hand, PartsAD-15K is a non-categorical dataset. For this dataset, we randomly selected a subset of 4,200 shapes from the large-scale ABC. The test set of PartsAD-15K is comprised of unseen shapes. Given a 3D mesh model of an object, we automatically create anomalies by applying geometric deformations as described below.

Different types of Anomalies. We consider the following anomaly types: cracks, fractures, geometric deformations (*e.g.*, bumps, bends, and twists), misaligned parts (*e.g.*, translation and rotation), material mismatch, and missing parts. Fig. A1 depicts the distribution of different anomaly types in our proposed dataset. Before applying deformations, we normalize the mesh vertices to $[-1, 1]$. Then we synthetically generate realistic cracks and fractures following [5] to mimic the shape's most geometrically natural breaking patterns. For geometric deformations, we use Blender's [7] Simple Deform and Hook modifier to create global (*e.g.*, bending and twisting) and local deformations (*e.g.*, surface bumps), respectively. For translation, we randomly translate the vertices of a random part by an offset. The translation offset for each axis is randomly sampled from a uniform distribution of range $[0.04, 0.08]$. For rotational anomalies, we apply a 3D rotational transformation to a random part. The rotation matrix is formed using a random rotation axis and a radian angle sampled from a uniform distribution of range $[0.2, 0.4]$. The center of rotation is set to a fixed point at one of the connecting points between the anomalous part and the main body of the object. We also randomly change the texture of a part or region to create a material mismatch anomaly or completely remove a part to simulate a missing parts anomaly. Additionally, for PartsAD-15K, we use a different but geometrically similar instance in the same scene as an anomaly (referred to as 'distinct' in Fig. A1). To retrieve a geometrically similar shape, we use feature-based KNN clustering. Specifically, we extract DINOv2 features from multi-view images (rendered from multiple fixed viewpoints) of the 3D shapes, concatenate these features, and use the resulting concatenated features to build the KNN cluster. Then, we retrieve a similar 3D shape by querying the KNN cluster. To ensure the retrieved shape is geometrically similar, we calculate the Chamfer distance between the shapes and only accept a shape if the distance is below a certain threshold.

Scene construction. For generating scenes, we use resulting anomalies along with their normal instances. A scene can have more than one anomaly of the same instance, or it may not have any anomaly at all. For randomized object placement, we first determine each object's resting pose by simulating a rigid body drop using Blender [7]. This simulation process is repeated for each object instance in the scene, including both normal and anomaly instances. Then, we place each object instance randomly using its obtained resting pose, while also ensuring that they do not collide with each other. We prevent collisions by maintaining a minimum distance between each pair of objects.

Quality checks. Our anomaly generation process is fully automated, with multiple quality checks to ensure reliable samples. For positional or rotational anomaly creation, we discard and regenerate any instance where a part detaches from the main body during deformation. Similarly, if removing a part results in an impractical shape, we try a different part instead. For fracture anomalies, we discard the sample where the fracture removes more than 90% or less than 10% of an object, regenerating a new fracture in these cases. Finally, we ensure the anomalous region of an object is visible from at least one viewpoint.

Assets. To generate a realistic scene environment, we use PBR materials [8] for floors and HDRI environment maps [11] for image-based lighting to illuminate scenes. We randomly select a pair of PBR material and HDRI environment maps from the assets to randomize the scene background. We employed Blender 2.93 [7] with Cycles ray-tracing renderer for photo-realistic rendering. Blender 2.93 is released under the GNU General Public License (GPL, or "free software"), and the PBR and HDRI maps are released under the CC0 license.

Rendering and view selection. We render each scene from various viewpoints sampled from a hemisphere around the scene origin. The viewpoint is parameterized in spherical coordinates where azimuth values are sampled uniformly over $[0, 2\pi)$ and elevation values are uniformly sampled in $[\pi/9, 2\pi/9]$. We adjust the radius in the range [1.5, 2.5] to ensure that all the objects in the scene are visible (even if partially) in the rendered view.

Table A1. Dataset composition of ToysAD-8K

	Categories
Seen	dinosaur, fish, frog, monkey, light, lizard, orange, boat, dog, lion, pig, cookie, panda, chicken, orange, ice, horse, car, airplane, cake, shark, donut, hat, cow, apple, bowl, hamburger, octopus, giraffe, chess, bread, butterfly, cupcake, bunny, elephant, fox, deer, bus, bottle
Unseen	mug, plate, robot, glass, sheep, shoe, train, banana, cup, key, penguin, hammer

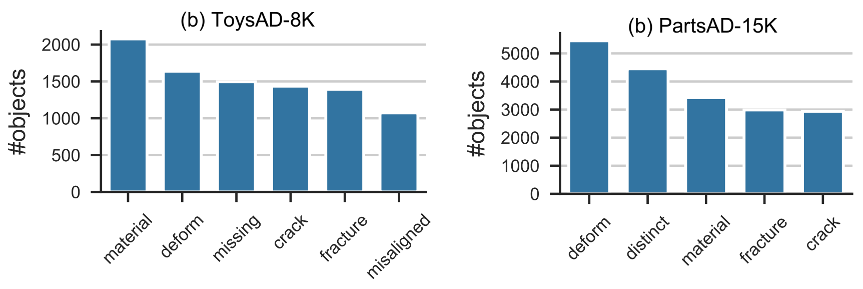

Figure A1. Distribution of anomaly types in the proposed ToysAD-8K and PartsAD-15 datasets.

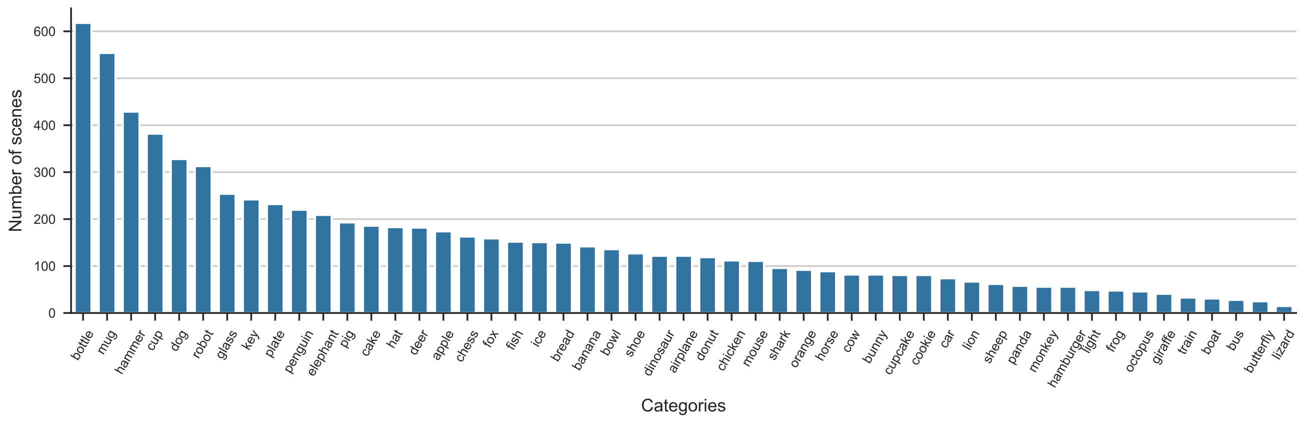

Figure A2. Distribution of object categories in ToysAD-8K dataset.

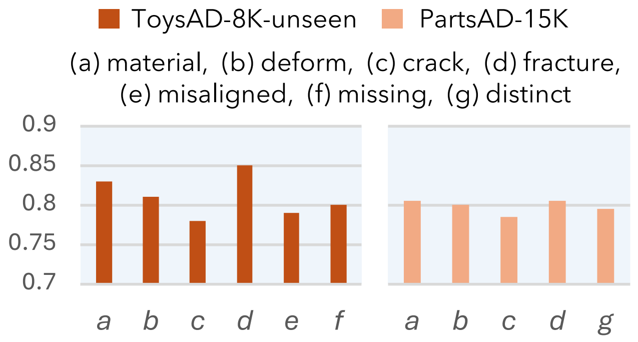

Figure A3. Anomaly detection performance (Accuracy) by anomaly type on the ToysAD-8K and PartsAD-15 datasets.

Our framework can easily be trained with real-world manufacturing scene environments. Our model relies solely on 2D supervision, making the data collection process much easier. We also do not need precise annotation of 3D bounding boxes as they are not used during training. For annotating instance-wise anomaly labels, we can use 2D bounding boxes, which can be projected in 3D using Visual Hull [2], then used for coarse localization.

B. Training Details

We train our model in two stages. In the first stage, we train it with just image and feature reconstruction losses. In the second stage, we train the model end-to-end with both reconstruction and binary classification losses. All the experiments are performed on a single NVIDIA A40 GPU with a batch size of 4, utilizing 28GB of GPU memory. The first stage takes 36 hours to complete, followed by an additional 24 hours for the second stage.

The ablation experiments are conducted on the same workstation with the same GPU by removing one or a few core components from the full method. Specifically, variant methods *A* and *B* take 36 hours to train the first stage, while only taking 14 hours to train the second stage. Regarding variant method C, it takes similar 36 and 24 hours for the two stages as in the full method.

We compare our method with Recons-Recog [4], ImVoxelNet [3], and DETR3D [10]. For the Recons-Recog approach, we use DGCNN [9] as a feature extractor. In terms of training time of these baseline methods, Recons-Recog takes 10 hours to train, ImVoxelNet takes 24 hours to train, and DETR3D takes 2 days to converge.

C. Additional Results

Fig. A4 shows a qualitative comparison with two baseline methods: ImVoxelNet and DETR3D. Our method accurately predicts the anomalies, while the baselines perform poorly. In the first three examples, the baselines fail due to their inability to compare with other objects in the scene, which is necessary for correct predictions. In the last example, both DETR3D and our method predict correctly, while ImVoxelNet fails. In Fig. A5 and Fig. A6, we present additional qualitative results on ToysAD-8K and PartsAD-15K, respectively. We show a breakdown of accuracy across different anomaly types in Fig. A3 to interpret the model's performance in various such scenarios.

References

[1] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *CVPR*, 2019. 1

[2] Kiriakos N Kutulakos and Steven M Seitz. A theory of shape by space carving. In *IJCV*, 2000. 2

[3] Danila Rukhovich, Anna Vorontsova, and Anton Konushin. Imvoxelnet: Image to voxels projection for monocular and multi-view general-purpose 3d object detection. In *WACV*, 2022. 3

[4] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, 2016. 3

[5] Silvia Sellán, Jack Luong, Leticia Mattos Da Silva, Aravind Ramakrishnan, Yuchuan Yang, and Alec Jacobson. Breaking good: Fracture modes for realtime destruction. In *ACM Transactions on Graphics*, 2023. 1

[6] Stefan Stojanov, Anh Thai, and James M Rehg. Using shape to categorize: Low-shot learning with an explicit shape bias. In *CVPR*, 2021. 1

[7] Blender Development Team. Blender (version 3.1.0) [computer software]. `https://blender.org/`, 2022. 1

[8] Anh Thai, Ahmad Humayun, Stefan Stojanov, Zixuan Huang, Bikram Boote, and James M Rehg. Low-shot object learning with mutual exclusivity bias. In *NeurIPS*, 2024. 1

[9] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. In *ACM Transactions on Graphics*, 2019. 3

[10] Yue Wang, Vitor Campagnolo Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, and Justin Solomon. Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In *CoRL*, 2022. 3

[11] Greg Zaal, Rob Tuytel, Rico Cilliers, James Ray Cock, Andreas Mischok, Sergej Majboroda, Dimitrios Savva, and Jurita Burger. Polyhaven: a curated public asset library for visual effects artists and game designers, 2021. 1

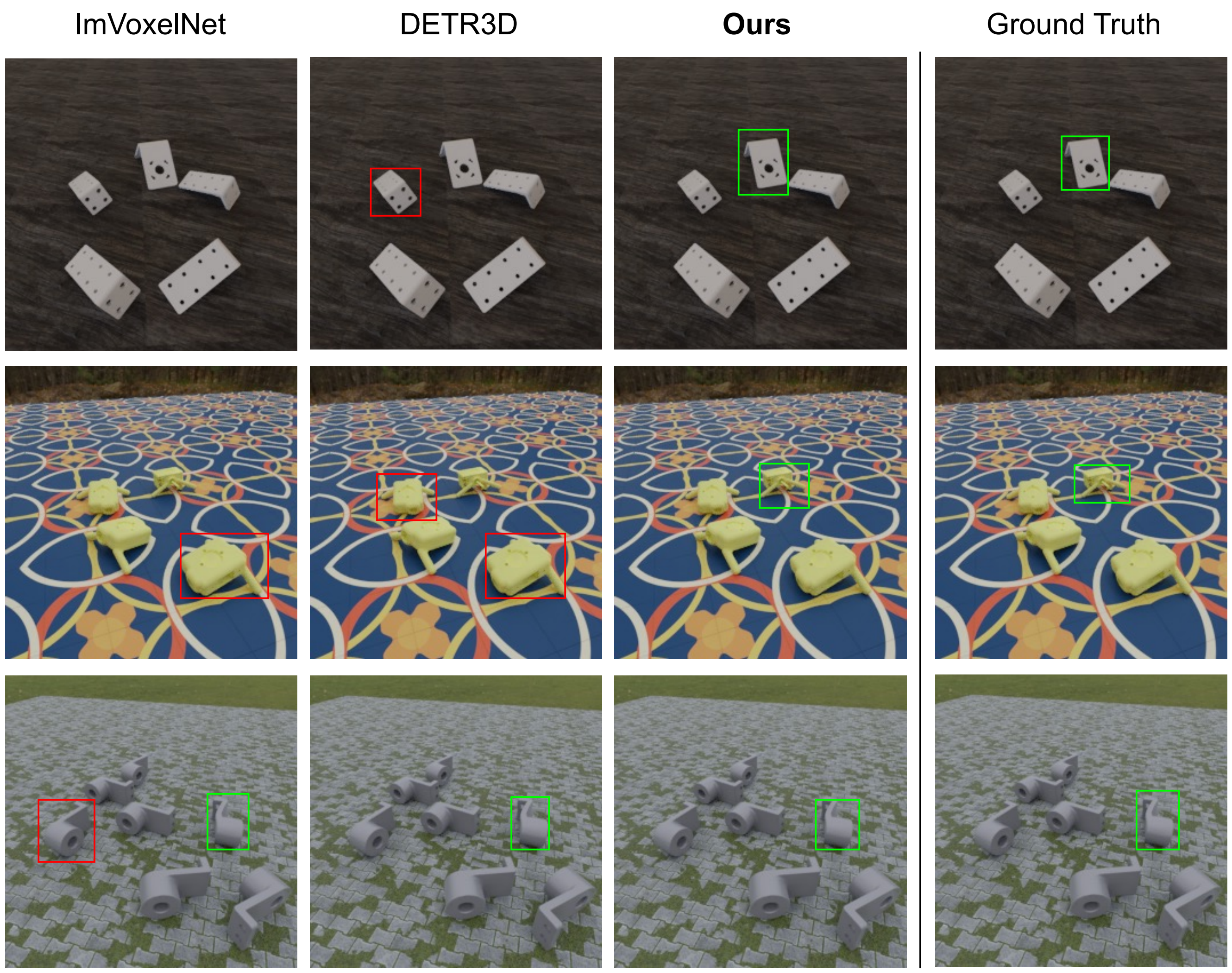

Figure A4. We qualitatively compare our method with two baselines on the PartsAD-15K dataset. ImVoxelNet failed to detect the anomalous object in the first example. We use the red box for wrong predictions and the green box for correct predictions.

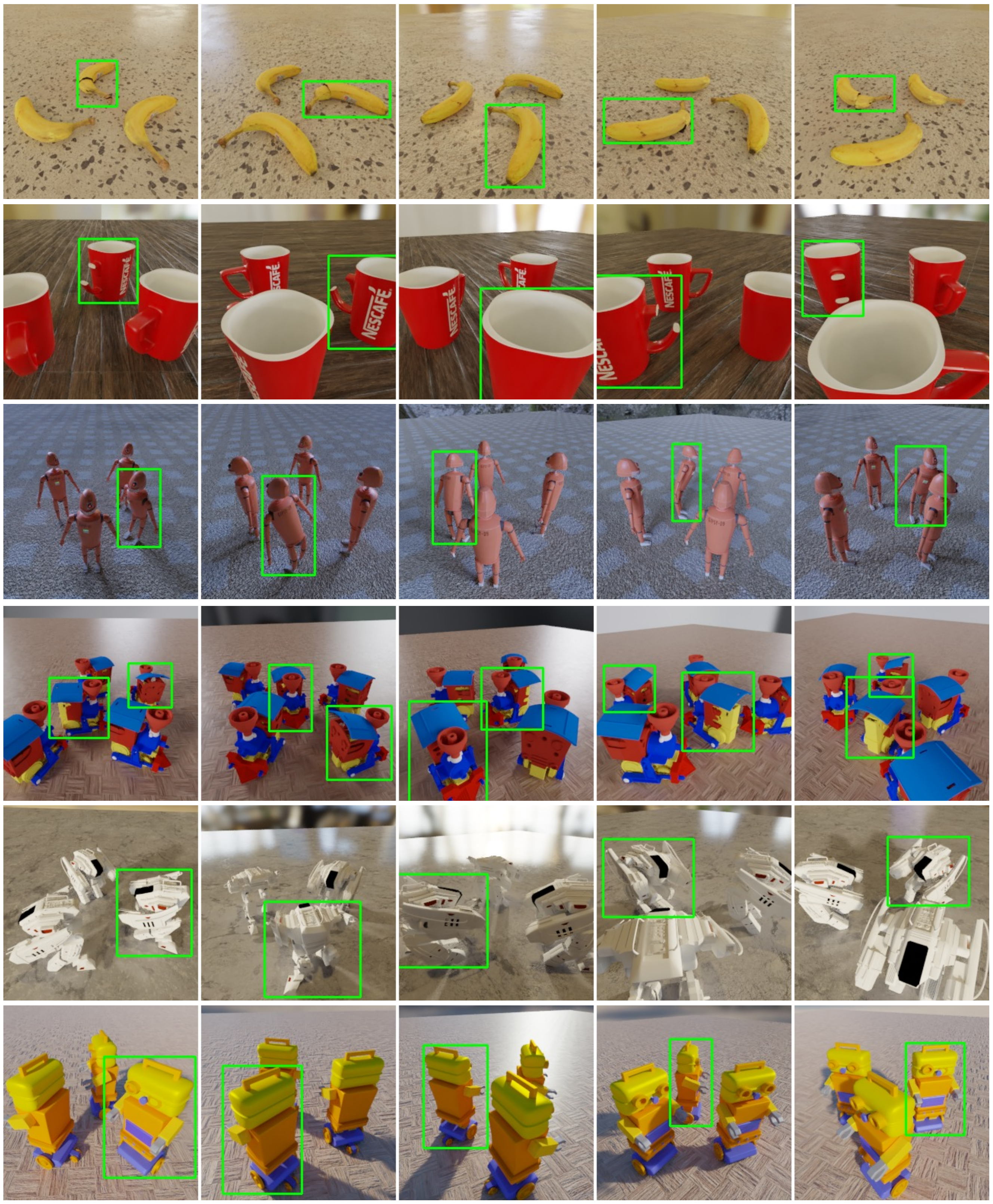

Figure A5. Additional results on the *unseen* set of ToysAD-8K dataset. Each row shows multiple views of the input scene and the green box denotes our model's prediction. For rows 1 to 3, the anomalies are easy to spot and self-explanatory. In row 4, there are two anomalies: one with broken outer parts at the back (see view 5), and another with a tilted roof (see views 1 and 2). For row 5, one leg is broken (see view 2). In row 6, the eye is missing (see view 5).

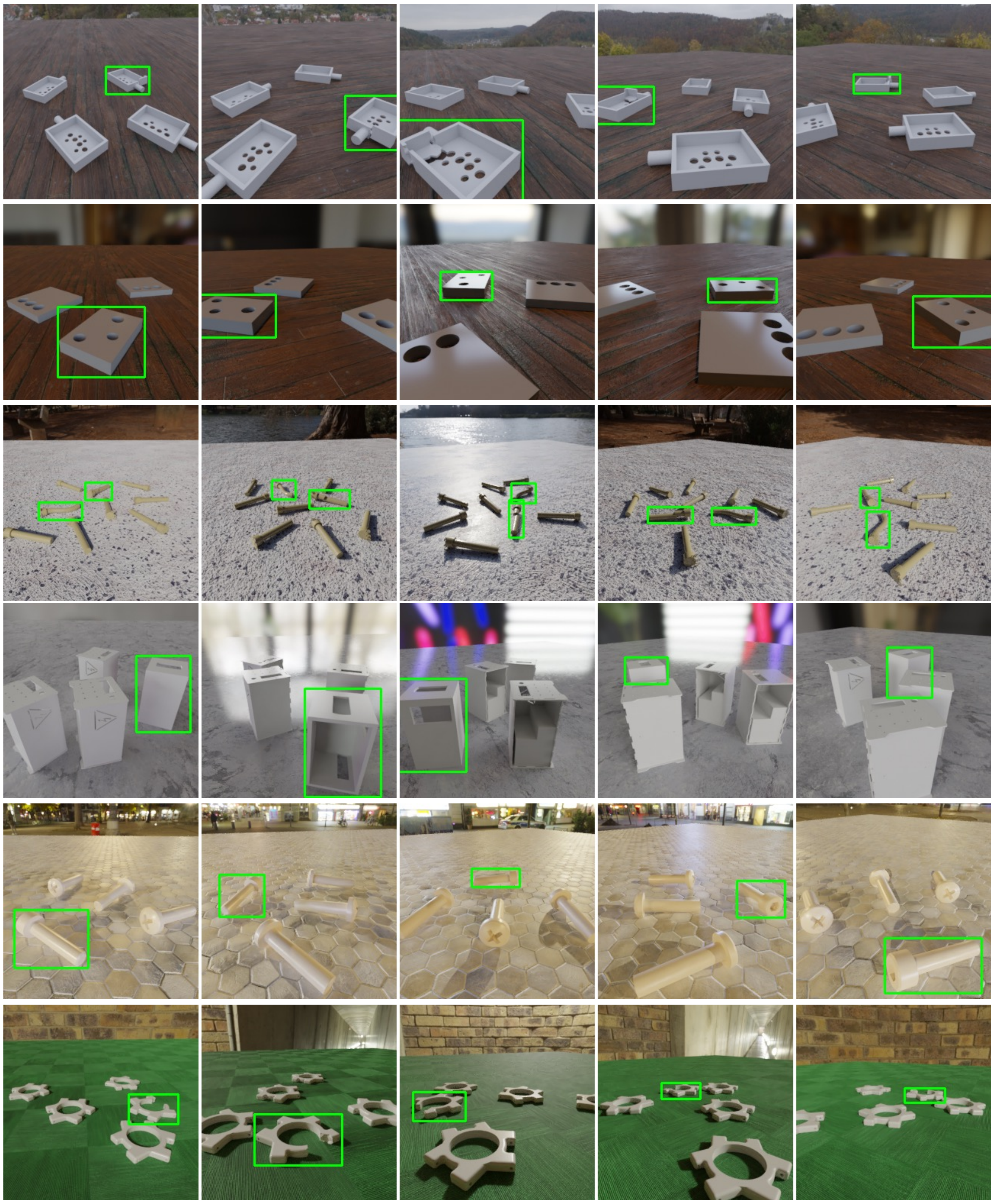

Figure A6. Additional results on the PartsAD-15K dataset. Each row shows multiple views of the input scene and the green box denotes our model's prediction.