

Supplementary Materials

Adv3D: Generating Safety-Critical 3D Objects through Closed-Loop Simulation

Anonymous Author(s)

Affiliation

Address

email

Abstract: In this supplementary material, we provide additional details on our method, implementation and experimental setups, and then show additional quantitative / qualitative results. We first detail how we implement ADV3D including how to build digital twins for realistic LiDAR simulation (Sec A.1), how to create a low-dimensional representation space (Sec A.2) for adversarial shapes, the details of two modern autonomy models (Sec A.3), the adversarial optimization procedure (Sec A.4) and more experimental details. Finally then provide additional results and analysis in Sec B including full closed-loop and open-loop results for major tables in the main paper and additional experiments and additional qualitative examples. Additionally, we include a supplementary video, **supplementary_56.mp4**, providing an overview of our methodology, as well as video results on generated adversarial shapes and how it affect the autonomy performance in different scenarios.

A ADV3D Implementation Details

A.1 Realistic LiDAR Simulation

Following [1, 2], we leverage real-world LiDAR data and object annotations to build surfel meshes (textured by per-point intensity value) for the virtual world. For complete background coverage, we drove through the same scene to collect multiple sets of driving data and then unify multiple LiDAR sweeps to a standard map coordinate system. We then aggregate LiDAR points from all frames and apply a dynamic point removal algorithm [3] to keep only static points and reconstruct the background \mathcal{B} . For dynamic actors, we aggregate the LiDAR points within object-centric coordinate bounding boxes for each labeled driving snippet. We then symmetrize the aggregated points along the vehicle’s heading axis for a more complete shape. Given the aggregated points, we then estimate per-point normals from 200 nearest neighbors with a radius of 20cm and orient the normals upwards for flat ground reconstruction, outwards for more complete dynamic actors. We downsample the LiDAR points into $4cm^3$ voxels and create per-point triangle faces (radius $5cm$) according to the estimated normals. Due to sparse observations for most aggregated surfel meshes, we manually curated a set of actor meshes that have complete and clean geometry, together with CAD assets purchased from TurboSquid [4] for a larger asset variety.

A.2 Adversarial Shape Representation

To ensure realism and watertight manifolds, we use all the CAD cars (sedan, sports car, SUV, Van, and pickup trucks) to build the low-dimensional representation. We first rescale all actors to be in a unit cube with a pre-computed scaling factor ($1.1 \times$ largest dimension for all actors). Then we convert the input meshes to volumetric signed distance fields (SDF) with a resolution of 100 (*i.e.*, $|\mathbf{L}| = 100^3$) using an open-source library¹. Then we apply principal component analysis [5] on the flattened SDF values $\Phi \in \mathbb{R}^{|\mathbf{L}| \times 1}$ to obtain the latent representation. Specifically, we use $K = 3$

¹<https://github.com/wang-ps/mesh2sdf>

principle components for constructing the latent space. In practice, we find the larger number we use K , the high-frequency details can be captured but the interpolated shapes can be less realistic. This is because the non-major components usually capture the individual details instead of shared properties across all vehicles.

During optimization, we first obtain the minimum and maximum latent range $\mathbf{z}_{\min} \in \mathbb{R}^3, \mathbf{z}_{\max} \in \mathbb{R}^3$, where the minimum and maximum value for each latent dimension is recorded. Then we optimize a unit vector $\bar{\mathbf{z}} \in \mathbb{R}^5$, where the first three dimensions are for PCA reconstruction, and the last two dimension indicates the scale value for width and length (range from 0.8 to 1.3). Then we normalize this first three dimension to $[\mathbf{z}_{\min}, \mathbf{z}_{\max}]$. Given the optimized latent \mathbf{z} , we apply Equation (2) in the main paper to get the generated 3D SDF volumes and then extract the meshes using marching cubes [6] algorithm. The extracted meshes are then scaled to the real-world size and placed in the virtual world for simulation.

A.3 LiDAR-Based Autonomy Details

Both autonomy systems tested consist of two parts, where the first part uses different joint perception and prediction models, and the second part share the same rule-based planner [7].

Autonomy-A (Instance-Based): We implement a variant of joint P&P model [8] to perform instance-based joint detection and trajectory prediction. For the 3D object detection part, we use a modified two-stage PIXOR [9] model following [10] which takes voxelized LiDAR point clouds as input and outputs the BEV bounding box parameters for each object. For the trajectory prediction part, we use a model that takes lane graph and detection results as input and outputs the per-timestep endpoint prediction for each object. The prediction time horizon is set to 6 seconds.

Autonomy-B (Instance-Free): We also verify our method on an instance-free autonomy system [12] for joint detection and motion forecasting to show its generalizability. Specifically, we replace the P&P model used in Autonomy-A with the occupancy-based model, which performs non-parametric binary occupancy prediction as perception results and flow prediction as motion forecasting results for each query point on a query point set. The occupancy and flow prediction can serve as the input for the sampling-based planner to perform motion planning afterwards.

A.4 Adversarial Optimization Details

Adversarial Objectives: The adversarial objective is given in Eqn. (3-5) in the main paper, where we set $\lambda_{\text{pred}} = 0.1$ and $\lambda_{\text{plan}} = 0.5$ for Autonomy-A. The adversarial objective for Autonomy-B also includes three terms: $\ell_{\text{det}}, \ell_{\text{pred}}$ and ℓ_{plan} , and we keep ℓ_{plan} as is since the PLT model we used is the same as Autonomy-A. However, the ImplicitO model in Autonomy-B does not have instance-level bounding box results, and the confidence score as well as IoU terms are no longer applicable. We thus follow [12, 13] and use the Soft-IoU metric to assess occupancy predictions. Similarly, we use the foreground mean end-point error (EPE) to measure the average L2 flow error at each occupied query point as done in [12], as the instance-based trajectory prediction is not available. Formally, the adversarial objective for Autonomy-B is defined as:

$$\mathcal{C}_t = \ell_{\text{det}}^t + \lambda_{\text{pred}} \ell_{\text{pred}}^t + \lambda_{\text{plan}} c_{\text{plan}}^t, \quad (1)$$

$$\ell_{\text{det}}^t = - \frac{\sum_{\mathbf{q} \in \mathcal{Q}} o(\mathbf{q}) \hat{o}(\mathbf{q})}{\sum_{\mathbf{q} \in \mathcal{Q}} (o(\mathbf{q}) + \hat{o}(\mathbf{q}) - o(\mathbf{q}) \hat{o}(\mathbf{q}))}, \quad (2)$$

$$\ell_{\text{pred}}^t = \frac{1}{\sum_{\mathbf{q} \in \mathcal{Q}} o(\mathbf{q})} \sum_{\mathbf{q} \in \mathcal{Q}} o(\mathbf{q}) \|\mathbf{f}(\mathbf{q}) - \hat{\mathbf{f}}(\mathbf{q})\|_2, \quad c_{\text{plan}}^t = c_{\text{jerk}}^t + c_{\text{lat}}^t, \quad (3)$$

where \mathcal{Q} is the query point set, $o(\mathbf{q})$ and $\hat{o}(\mathbf{q})$ are ground truth and predicted binary occupancy value $\in [0, 1]$ on the query point \mathbf{q} , respectively. Flow vector $\mathbf{f} : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ and the corresponding prediction $\hat{\mathbf{f}}$ specifies the BEV motion of any agent that occupies that location. We set $\lambda_{\text{pred}} = 1.0$ and $\lambda_{\text{plan}} = 0.5$ for Autonomy-B.

Black-box Optimization Details: To handle different modern autonomy systems and the non-differentiable LiDAR simulator, we adopt the black-box optimization in ADV3D. Inspired by existing

works [14, 15, 16], we adopt the Bayesian Optimization [17, 18] (BO) as the search algorithm, which maintains a surrogate model and select the next query candidate based on historical observations and acquisition function. Specifically, we use a standard Gaussian process (GP) model with Upper Confidence Bound [19, 20] (UCB) as the acquisition function. We set the exploration multiplier $\beta = 1.0$ to balance exploitation and exploration. Since the adversarial landscape is not locally smooth, we use the Matérn 3/2 kernel (product over each dimension with a length scale of 0.1) for the GP model. Unless stated otherwise, we set the total query budget as 100 and the first 11 queries are used for the initialization.

We also benchmark the other popular black-box algorithms including grid search [21] (GS), random search [22, 23, 24] (RS) and blend search (BS) [25]. For GS, we set 3 search points per dimension thus in total $3^5 = 243$ queries. For a random search, we set the query budget as 500 to achieve better performance. BS is an economical hyperparameter optimization algorithm that combines local search and global search. We adopt the official implementation². We also compare a baseline that conducts brute-forcing (BF) over the curated asset library with 746 vehicles and find the worst-case actor shape. Our optimization pipeline is built on the Ray Tune framework [26].

A.5 Additional Experimental Details

Realism Evaluation for Generated Shapes: We evaluate the realism of ADV3D using Jensen–Shannon divergence [27] (JSD) between generated shapes by ADV3D and vertex deformation (VD). Specifically, we calculate JSD by uniformly sampling point clouds of 1000 points from the optimized shapes with the birds-eye-view 2D histogram of all CAD models in our asset sets (resolution of 100×100).

B Additional Results and Analysis

Attacking Full Autonomy Stack: To take into account the full autonomy stack, we find it is important to use an adversarial objective that takes a combination of submodule costs. In Tab. A1, we provide the full results for Tab. 3 in the main paper, including missing combinations \mathcal{M}_4 and \mathcal{M}_5 . Moreover, we also compare with the results in closed loop using shapes generated by open-loop attack.

#ID	Opt. Settings	Perception $\sum_t \ell_{\text{det}}^t$	Prediction $\sum_t \ell_{\text{pred}}^t$	Planning $\sum_t c_{\text{plan}}^t$	AP \uparrow (%, @0.5)	Recall \uparrow (%, @0.5)	minADE \downarrow L_2 error	meanADE \downarrow L_2 error	Lat. \downarrow (m/s^2)	Jerk \downarrow (m/s^3)
Original					88.7	89.4	2.51	4.99	0.261	0.294
\mathcal{M}_1	Open-Loop	✓			80.4	87.6	2.01	4.95	0.256	0.301
	Closed-Loop				69.6	71.4	1.97	5.02	0.239	0.310
\mathcal{M}_2	Open-Loop		✓		83.5	88.5	2.52	5.39	0.223	0.341
	Closed-Loop				83.1	89.1	2.92	6.34	0.254	0.412
\mathcal{M}_3	Open-Loop			✓	87.2	88.8	2.57	5.38	0.305	0.352
	Closed-Loop				86.7	88.3	2.94	6.03	0.324	0.434
\mathcal{M}_4	Open-Loop	✓	✓		79.9	85.9	2.57	5.35	0.231	0.353
	Closed-Loop				70.1	78.8	2.90	5.98	0.223	0.401
\mathcal{M}_5	Open-Loop	✓		✓	81.2	84.3	2.57	5.60	0.333	0.253
	Closed-Loop				72.3	75.0	2.95	6.04	0.342	0.401
\mathcal{M}_0	Open-Loop	✓	✓	✓	85.5	87.7	2.73	5.99	0.262	0.372
	Closed-Loop				75.4	76.4	2.82	6.21	0.411	0.410

Table A1: **Full table of adversarial optimization for the full autonomy stack.** Unlike existing works that consider sub-modules, ADV3D generates actor shapes that are challenging to all downstream modules.

Latent Asset Representation: We repeat the experiment from Tab. 3 but now using a lower density 100 triangle mesh which has a lower dimension thus can be optimized with BO. Results in Tab. A2 show that large vertex deformation is required to achieve similar attack strength as Adv3D. Moreover, the vertex deformed actors are overly simplified and unrealistic with noticeable artifacts.

²<https://github.com/microsoft/FLAML>

Algorithms	AP \uparrow (%, @0.5)	Recall \uparrow (%, @0.5)	minADE \downarrow L_2 error	Jerk \downarrow (m/s^3)	JSD
Original	98.7	99.6	4.70	0.090	—
VD: 0.05m	98.7	99.6	5.04	0.090	0.057
VD: 0.1m	98.7	99.6	5.10	0.090	0.137
VD: 0.2m	98.7	99.6	5.16	0.090	0.253
VD: 0.5m	78.3	81.2	5.77	0.103	0.745
VD: 1.0m	45.5	48.5	5.79	0.155	0.758
ADV3D (ours)	50.3	55.8	7.87	0.111	0.175

Table A2: Compare with vertex deformation.

Adv3D VD (5 cm) VD (10 cm) VD (20 cm) VD (50 cm) VD (100 cm)

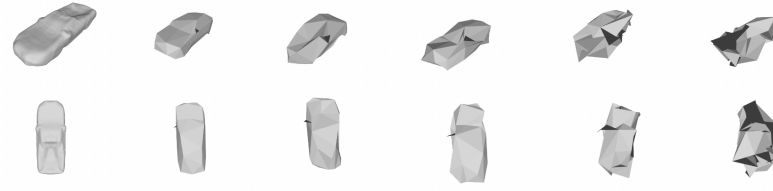


Figure A1: Qualitative comparisons with Vertex-Deformation (VD). Top and bottom show side and top-down views respectively.

111 **Additional Qualitative Examples:** We provide more qualitative examples in Figure A2, A3 and A4
112 to show that ADV3D is able to generate safety-critical actor shapes for autonomy testing with
113 appearance coverage.

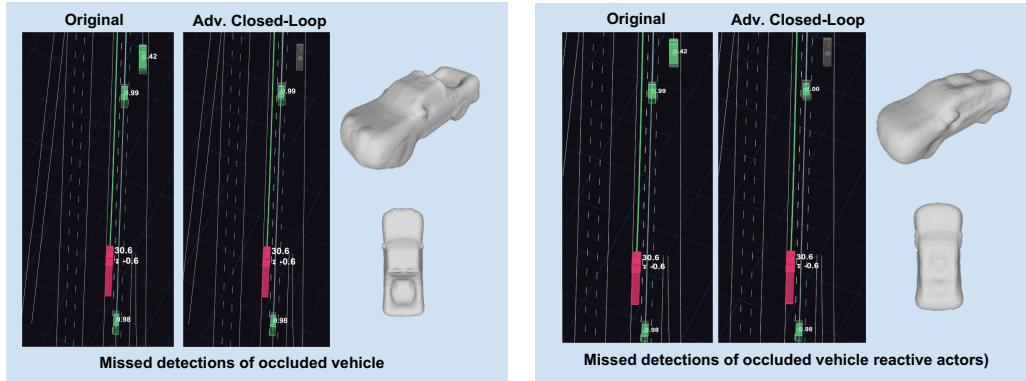


Figure A2: Qualitative examples of adversarial shape generation (non-reactive actors vs reactive actors). ADV3D is able to generate adversarial actors that cause detection failures due to occlusion.

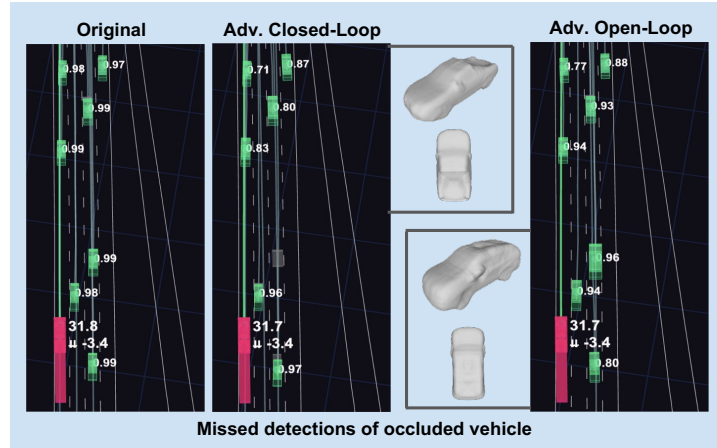


Figure A3: Qualitative examples of adversarial shape generation in closed loop vs open loop. ADV3D is able to generate adversarial actors that cause detection failures due to occlusion but the open-loop counterpart fails.

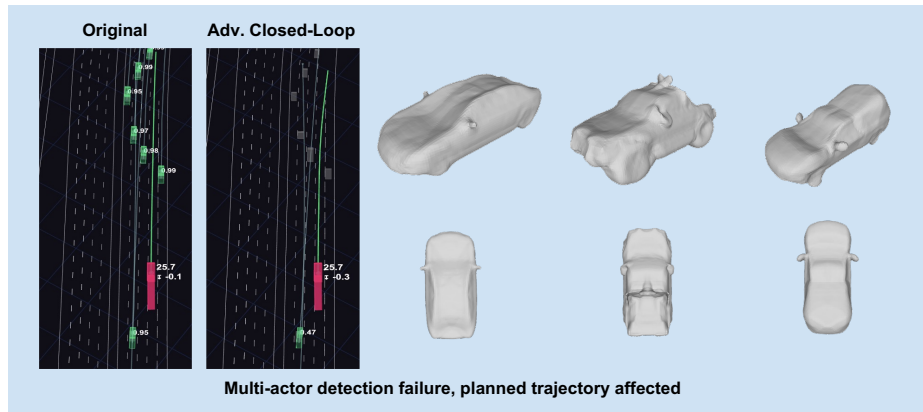


Figure A4: Qualitative examples of adversarial shape generation with multi-actor attacks. ADV3D creates three safety-critical shapes that cause detection failures for all front actors.

References

- [1] S. Manivasagam, S. Wang, K. Wong, W. Zeng, M. Sazanovich, S. Tan, B. Yang, W.-C. Ma, and R. Urtasun. Lidarsim: Realistic lidar simulation by leveraging the real world. In *CVPR*, 2020.
- [2] Z. Yang, Y. Chai, D. Anguelov, Y. Zhou, P. Sun, D. Erhan, S. Rafferty, and H. Kretzschmar. Surfelgan: Synthesizing realistic sensor data for autonomous driving. *CVPR*, 2020.
- [3] H. Thomas, B. Agro, M. Gridseth, J. Zhang, and T. D. Barfoot. Self-supervised learning of lidar segmentation for autonomous indoor navigation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14047–14053. IEEE, 2021.
- [4] TurboSquid. <https://www.turbosquid.com>, Access date: 2023-05-17.
- [5] K. Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.
- [6] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.
- [7] A. Sadat, M. Ren, A. Pokrovsky, Y.-C. Lin, E. Yumer, and R. Urtasun. Jointly learnable behavior and trajectory planning for self-driving vehicles. *IROS*, 2019.
- [8] M. Liang, B. Yang, W. Zeng, Y. Chen, R. Hu, S. Casas, and R. Urtasun. Pnpnet: End-to-end perception and prediction with tracking in the loop. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11553–11562, 2020.
- [9] B. Yang, W. Luo, and R. Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018.
- [10] Y. Xiong, W.-C. Ma, J. Wang, and R. Urtasun. Learning compact representations for lidar completion and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1074–1083, 2023.
- [11] A. Cui, S. Casas, K. Wong, S. Suo, and R. Urtasun. Gorela: Go relative for viewpoint-invariant motion forecasting. *arXiv preprint arXiv:2211.02545*, 2022.
- [12] B. Agro, Q. Sykora, S. Casas, and R. Urtasun. Implicit occupancy flow fields for perception and prediction in self-driving. In *CVPR*, 2023.
- [13] J. Kim, R. Mahjourian, S. Ettinger, M. Bansal, B. White, B. Sapp, and D. Anguelov. Stop-net: Scalable trajectory and occupancy prediction for urban autonomous driving. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8957–8963. IEEE, 2022.
- [14] Y. Abeyirigoonawardena, F. Shkurti, and G. Dudek. Generating adversarial driving scenarios in high-fidelity simulators. In *ICRA*, 2019.
- [15] J. Wang, A. Pun, J. Tu, S. Manivasagam, A. Sadat, S. Casas, M. Ren, and R. Urtasun. Advsim: Generating safety-critical scenarios for self-driving vehicles. In *CVPR*, 2021.
- [16] A. Boloor, K. Garimella, X. He, C. Gill, Y. Vorobeychik, and X. Zhang. Attacking vision-based perception in end-to-end autonomous driving models. *Journal of Systems Architecture*, 110: 101766, 2020.
- [17] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.
- [18] B. Ru, A. Cobb, A. Blaas, and Y. Gal. Bayesopt adversarial attack. In *International conference on learning representations*, 2020.
- [19] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.
- [20] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

- 160 [21] P. Liashchynskiy and P. Liashchynskiy. Grid search, random search, genetic algorithm: a big
161 comparison for nas. *arXiv preprint arXiv:1912.06059*, 2019.
- 162 [22] C. Guo, J. Gardner, Y. You, A. G. Wilson, and K. Weinberger. Simple black-box adversarial
163 attacks. In *International Conference on Machine Learning*, pages 2484–2493. PMLR, 2019.
- 164 [23] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein. Square attack: a query-efficient
165 black-box adversarial attack via random search. In *Computer Vision–ECCV 2020: 16th
166 European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII*, pages
167 484–501. Springer, 2020.
- 168 [24] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of machine
169 learning research*, 13(2), 2012.
- 170 [25] C. Wang, Q. Wu, S. Huang, and A. Saied. Economic hyperparameter optimization with blended
171 search strategy. In *International Conference on Learning Representations*, 2021.
- 172 [26] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica. Tune: A research
173 platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.
- 174 [27] V. Zyrianov, X. Zhu, and S. Wang. Learning to generate realistic lidar point clouds. In
175 *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27,
176 2022, Proceedings, Part XXIII*, pages 17–35. Springer, 2022.