MultiTest: Physical-Aware Object Insertion for Testing Multi-sensor Fusion Perception Systems

Xinyu Gao*
State Key Laboratory for Novel Software Technology
Nanjing University, China

Zhijie Wang*
University of Alberta
Edmonton, Canada

Yang Feng†
State Key Laboratory for Novel Software Technology
Nanjing University, China

Lei Ma†
The University of Tokyo, Japan
University of Alberta, Canada

Zhenyu Chen†
State Key Laboratory for Novel Software Technology
Nanjing University, China

Baowen Xu
State Key Laboratory for Novel Software Technology
Nanjing University, China

ABSTRACT

Multi-sensor fusion stands as a pivotal technique in addressing numerous safety-critical tasks and applications, e.g., self-driving cars and automated robotic arms. With the continuous advancement in data-driven artificial intelligence (AI), MSF's potential for sensing and understanding intricate external environments has been further amplified, bringing a profound impact on intelligent systems and specifically on their perception systems. Similar to traditional software, adequate testing is also required for AI-enabled MSF systems. Yet, existing testing methods primarily concentrate on single-sensor perception systems (e.g., image-based and point cloud-based object detection systems). There remains a lack of emphasis on generating multi-modal test cases for MSF systems.

To address these limitations, we design and implement MultiTest, a fitness-guided metamorphic testing method for complex MSF perception systems. MultiTest employs a physical-aware approach to synthesize realistic multi-modal object instances and insert them into critical positions of background images and point clouds. A fitness metric is designed to guide and boost the test generation process. We conduct extensive experiments with five SOTA perception systems to evaluate MultiTest from the perspectives of: (1) generated test cases' realism, (2) fault detection capabilities, and (3) performance improvement. The results show that MultiTest can generate realistic and modality-consistent test data and effectively detect hundreds of diverse faults of an MSF system under test. Moreover, retraining an MSF system on the test cases generated by MultiTest can improve the system's robustness. Our replication package and synthesized testing dataset are publicly available at https://sites.google.com/view/msftest.

*Both authors contributed equally to this work.
†Yang Feng, Lei Ma, and Zhenyu Chen are the corresponding authors.

ICSE '24, April 14–20, 2024, Lisbon, Portugal
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0217-4/24/04…$15.00
https://doi.org/10.1145/3597503.3639191

CCS CONCEPTS

• **Software and its engineering → Software testing and debugging**.

KEYWORDS

Testing, Multi-Sensor Fusion, Perception Systems

ACM Reference Format:
Xinyu Gao, Zhijie Wang, Yang Feng, Lei Ma, Zhenyu Chen, and Baowen Xu. 2024. MultiTest: Physical-Aware Object Insertion for Testing Multi-sensor Fusion Perception Systems. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3597503.3639191

1 INTRODUCTION

Multi-sensor fusion (MSF) plays a vital role in various intelligent machines and software systems. The recent rapid advancements in data-driven artificial intelligence (AI) and sensor technologies further propelled progress in the development of MSF-based systems. Nowadays, prominent industrial-level systems, such as OpenPilot [9], commonly rely on multi-sensor fusion strategies to overcome the inherent limitations of individual sensors and enhance overall system performance [17]. Consequently, MSF-based perception systems have found widespread applications in diverse industrial domains and safety-critical use cases, including self-driving cars [10], unmanned aerial vehicles [24], and robotic systems [13].

Despite the rapid progress, AI-enabled perception systems, similar to any traditional software, can still yield incorrect prediction results, which can further lead to incorrect system behavior. Oftentimes the incorrect behavior could result in severe accidents and losses in safety-critical contexts, e.g., autonomous driving. For instance, a Tesla self-driving car failed to distinguish a white truck against a bright sky [12], causing a fatal collision.

To improve the overall quality of AI-enabled perception systems, software engineering and machine learning researchers have proposed a few quality assurance techniques for the different development stages, e.g., testing [8, 23, 52, 55] and debugging [29, 30]. Among these approaches, testing has emerged as a proven and efficient method to assess the potential risks of deploying an AI-enabled perception system in real-world scenarios. A typical testing workflow takes a small set of tests (e.g., images) as seeds and generates more challenging test cases based on seeds. Existing testing techniques for AI-enabled perception systems usually leverage natural/adversarial perturbations to synthesize new test data [23, 55]. However, they mostly focus on testing single-sensor (e.g., camera or LiDAR) perception systems. Yet, little has been done from the perspective of testing MSF-based systems [21]. A systematic survey from practitioners in autonomous driving shows that there comes an urgent need for multi-modal data synthesis techniques [35].

We argue two critical challenges exist in the field of testing MSF-based perception systems. Firstly, compared with the testing of single-sensor perception systems, testing MSF-based perception systems would require the synthesis of modality-consistent tests across different sensors (e.g., a car appears in an image should have the same pose in the corresponding point cloud). Therefore, a mere combination of image-based and point cloud-based testing methods cannot guarantee such consistency. To address this challenge, Gao et al. collected a set of perturbations that can be leveraged for testing MSF systems [21]. However, the testing efficiency is highly limited due to a lack of proper guidance when generating test cases. Another challenge comes from the realism of synthesized test cases. For perception systems operating in the physical world, the realism of test cases directly impacts the value of detected errors. Some existing perturbations used in testing, such as adversarial noises, seldom occur in real-world environments. Moreover, simple perturbations also limit the diversity of test generation from different perspectives (changing the number of cars and pedestrians, etc.). To address this, Wang et al. proposed an object insertion method for testing image-based object detectors [52]. However, without appropriate physical constraints, the inserted object might appear in invalid positions (e.g., a car hanging in the air) or result in incorrect perspective relationships (e.g., an occluded but visible building).

To address these issues, we propose MultiTest, an automated testing method for MSF-based perception systems based on physical-aware multi-modal object insertion. Given a multi-modal test seed (i.e., a pair of image and point cloud frames), MultiTest automatically selects a 3D object instance (e.g., a car) from database then inserts it into the original data. To generate semantically plausible test data (i.e., the newly inserted car is on the road with correct heading), MultiTest first searches for valid poses of the inserted object. Then, MultiTest synthesizes realistic images and point cloud frames to handle the occlusion between the inserted object and the background data. Finally, MultiTest leverages a fitness-guided approach to insert the object for the purpose of synthesizing more challenging test cases for systems under test. MultiTest further employs metamorphic relations between the synthesized data and seed data to automatically detect faulty perception results. MultiTest can generate modality-consistent and realistic tests from the seed testing data with physical-aware virtual sensors.

To evaluate the effectiveness and efficiency of MultiTest, we conduct experiments on three popular MSF-based object detection systems. We further experiment MultiTest with two single-sensor object detection systems (camera-only and LiDAR-only) to evaluate the generalization ability of MultiTest. We find that MultiTest is capable of generating realistic and modality-consistent test data to satisfy test input specifications for both single-sensor and MSF-based perception systems. The experimental results also demonstrate that MultiTest can effectively detect hundreds of erroneous behavior across different MSF-based perception systems. We further retrain the selected perception systems with test data generated by MultiTest, finding that the system performance (measured by average precision) can be improved by 24% on average.

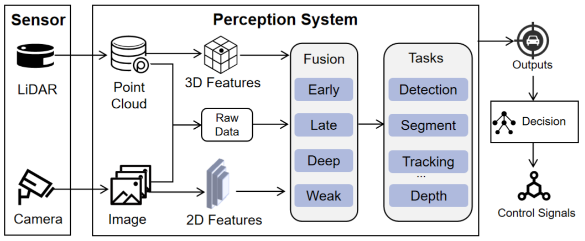


Figure 1: The workflow of MSF perception systems.

In summary, this work makes the following contributions:

- **Method.** We propose an automated testing approach for MSF perception systems based on fitness-guided metamorphic testing. Specifically, we leverage a physical-aware approach to insert new object instances to critical positions of the test seed to generate realistic and modality-consistent data.
- **Tool.** We implement the above method into a tool MultiTest. To the best of our knowledge, MultiTest is the first automated testing tool for MSF-based perception systems. We have released MultiTest and the multi-modal data generated by MultiTest on our anonymous supplementary website: https://sites.google.com/view/msftest.
- **Evaluation.** We conduct extensive experiments to evaluate the performance of MultiTest with five perception systems. The results show that MultiTest can generate realistic and modality-consistent test data and efficiently detect erroneous system behavior. Retraining an MSF system with the generated test cases can significantly increase its robustness.

2 BACKGROUND

2.1 Multi-Sensor Fusion Perception Systems

Perception systems rely on sensors to capture real-world information in a particular data format. Subsequently, they process and fuse data from various different sensors using specific algorithms to help an intelligent system understand the external environment [26]. As shown in Fig. 1, perception systems are commonly integrated as a module within intelligent software systems (e.g., a self-driving car, a robotic, or an unmanned aerial vehicle). They are responsible for executing specific perception tasks, such as object detection and object tracking. The perception results are then communicated to downstream decision modules, such as planning and control, to facilitate the smooth operation of the entire intelligent system. As a result, the quality and reliability of the entire system are significantly influenced by its perception systems.

To further enhance the performance of the system, most industrial-level systems [41] employ the multi-sensor fusion (MSF) strategy to avoid inherent perception limitations of individual sensors, leading to improved accuracy and reliable sensing capabilities. For example, a camera can capture rich semantic information but is sensitive to the environmental changes (e.g., rain and snow), while LiDAR can provide high-quality 3D geometric information but lacks the capability of semantic understanding. The camera-LiDAR fusion is also highly favored among different multi-sensor configurations.

To better fuse heterogeneous data with different characteristics and storage structure, researchers further propose AI-enabled MSF techniques. Based on the stage of data fusion, these fusion techniques can be categorized into four different mechanisms at a high level: *early fusion*, *late fusion*, *deep fusion*, and *weak fusion* [17]. **Early fusion** is the fusion of raw or pre-processed sensor data, which usually fuses heterogeneous data by converting them into the same data type according to rules. While this fusion mechanism has low computational and memory requirements, its inflexibility and significant limitations have led to its infrequent standalone use. **Late fusion** directly merges the output results from both the LiDAR and the camera branch to make the final prediction. Each branch independently processes data from sensors without relying on specific network architecture, which makes it more flexible and efficient. **Deep fusion**, on the other hand, combines hidden features from different branches at varying depths to gain rich semantic information. Frequent interactions between different branches enable the network to learn cross-modalities with diverse feature representations. **Weak fusion** commonly uses rule-based methods to transform data from an additional branch to provide guidance for processing data in the main branch. A typical example of weak fusion is extracting the frustums in the point cloud data using the 2D detection bounding boxes from the image as guidance.

2.2 The Preliminaries of Object Detection

Object detection is a fundamental perception task for intelligent machines, enabling them to understand external environments. The primary objective of object detection is to estimate bounding boxes around objects of interest and predict their classification labels. This task can be categorized into two types based on dimensionality: 2D and 3D object detection. 2D object detection is typically performed on image data. It involves estimating a bounding box for each object, providing the object's position as $[x, y]$ and the bounding box's width w and height h in pixels. In contrast, 3D object detection goes beyond 2D by additionally estimating the position $[x, y, z]$ of each object in 3D space, along with the bounding box's length l, width w, height h, and orientation angles $[roll, pitch, yaw]$. It is worth noting that, in the context of autonomous driving, practitioners usually consider only the yaw orientation (i.e., the heading) for simplicity. 3D object detection can be performed on image data, point cloud data, or a fusion of both, depending on the available sensors and the specific application requirements.

Given different input data types (i.e., images or point clouds), object detection systems may employ diverse model architectures and pipelines. For image data, popular object detection systems utilize convolutional neural networks (CNNs) as the backbone to gather rich semantic hidden features [33, 46, 47]. Subsequently, a head network is used to predict the bounding box's position, size, and the object's classification label. Different from image data, point cloud data includes a set of orderless 3D points. Representative point cloud-based detection systems leverage two strategies to obtain semantic features from the point cloud: (1) voxel-based and (2) point-based methods. Voxel-based methods partition the point cloud into several fixed-resolution 3D grids (voxels) and use 3D CNNs as the backbone to extract point cloud features [57, 61]. By contrast, point-based methods directly extract features from raw point clouds via fully-connected networks [43, 44] or specialized convolution operations [34] for points for 3D detection. MSF-based object detection systems further take both these two input data types into account. Given different fusion strategies, an MSF-based object detection system might either extract features from each sensor's perception [27] or project the information captured in one modality to another modality before extracting features [6].

The accuracy of object detection is usually measured by IOU (intersection over union) [38] and AP (average precision) [15]. IOU measures the overlap area between a ground-truth bounding box B_g and a predicted bounding box B_p over their union. The computation of IOU can be represented as $IOU = \text{area}\left(B_p \cap B_g\right)/\text{area}\left(B_p \cup B_g\right)$. The object is *successfully detected* in case IOU is greater than a given threshold τ. In this paper, we set the IOU threshold τ as 0.5, which is consistent with the previous study [52]. AP is used to measure the performance of the overall detection performance on a dataset, which can be derived by computing the area under the Precision/Recall curve as

$$\text{AP}|_R = \frac{1}{|R|} \sum_{r \in R} \rho_{\text{interp}}(r) \tag{1}$$

where ρ_{interp} is the interpolation function, which is defined as: $\rho_{\text{interp}}(r) = \max_{r':r' \geq r} \geq (r')$, where $\rho(r)$ gives the precision at recall level r. In this paper, we apply forty equally spaced recall levels recommended by KITTI, i.e., $R_{40} = \{1/40,\ 2/40,\ \ldots,\ 1\}$.

For a more in-depth evaluation of the object detection tasks, Wang et al. [52] classified object detection errors into three categories, i.e., *Recognition failures*, *Localization failures*, and *Classification failures*. Note that in this paper, we mainly focus on recognition failures and localization failures.

Recognition failures include two independent types of errors, i.e. *Object missing* and *False detection*. Object missing refers to the case that an object detection system fails to recognize an existing object, while false detection refers to the case that the system treats an arbitrary region without objects as an "object". Given a ground-truth bounding box $B_g \in \mathbb{GT}$ and a predicted bounding box $B_p \in \mathbb{DT}$, the object missing can be formalized as:

$$\exists B_g \in \mathbb{GT} \wedge \forall B_p \in \mathbb{DT}, IOU(B_g, B_p) \leq 0 \tag{2}$$

and the false detection can be be formalized as:

$$\forall B_g \in \mathbb{GT} \wedge \exists B_p \in \mathbb{DT}, IOU(B_g, B_p) \leq 0 \tag{3}$$

Localization failures refer to the errors that an estimated bounding box is too large or too small, which can be formalized as:

$$\exists B_g \in \mathbb{GT} \wedge \exists B_p \in \mathbb{DT}_l, 0 < IOU(B_g, B_p) \leq \tau \tag{4}$$

where $\mathbb{DT}_l$ is detection bounding boxes with neither successful detection nor false detection.

3 APPROACH

In this section, we introduce the design of MultiTest. Fig. 2 presents the high-level workflow of MultiTest. Given a background multi-modal data recorded from real-world and an object instance selected from the object database, MultiTest first executes the *pose estimation* module to calculate the valid locations and

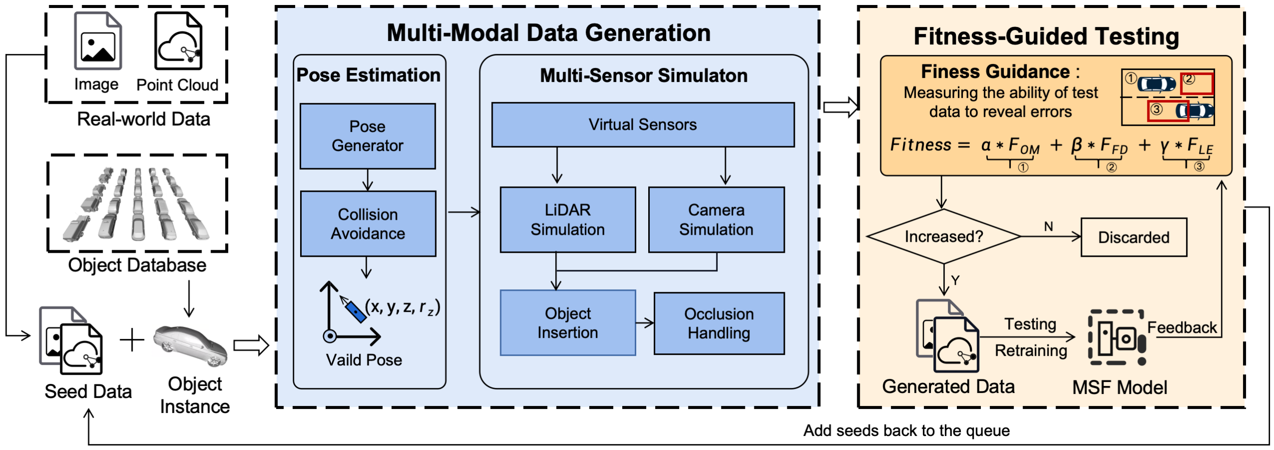


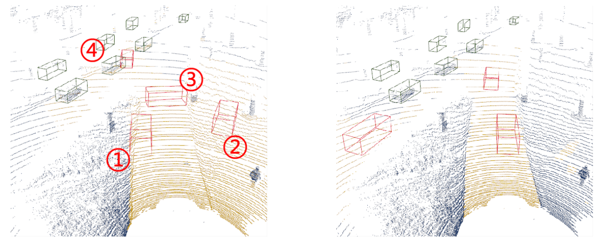
Figure 2: The workflow of MultiTest.

orientations of an object to be inserted. Then the *multi-sensor simulation* module renders the object instance in the form of both image and point cloud given the calculated poses in a physical-aware virtual simulator. The *multi-sensor simulation* module further merges the synthesized image and point cloud of the inserted object with the background data and carefully handles the occlusion. These two modules form the MultiTest's **multi-modal test data generation** pipeline. Finally, the realistic multi-modal test data can be efficiently generated through **fitness guided metamorphic testing**. We detail each module of MultiTest in the following.

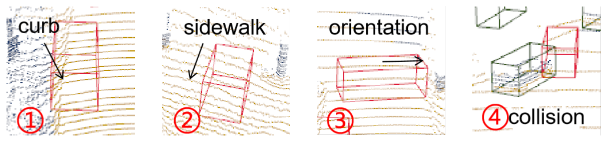
3.1 Pose Estimation

Given the background data and an object instance, the pose estimation module aims to estimate possible valid locations and orientations to create a plausible scene in the real world after inserting the object. This is critical for synthesizing realistic test data for real-world perception systems because failing to address it could result in semantically invalid data. For instance, a synthesized test case should be considered invalid if a car is inserted outside of the road or if the car's heading is incorrect (Fig. 3(c)).

Pose Generator. Given a candidate scene m (a.k.a. a test seed, composed by a pair of image and point cloud $\langle image, pc \rangle$) to insert and an object o to be inserted, MultiTest's pose generator first samples a set of possible positions and orientations $\mathcal{P}$ ($pose\_i \in \mathcal{P}$, $pose\_i = [x_i, y_i, z_i, r_{zi}]$ is in the LiDAR's coordinate) in the corresponding 3D space. To obtain $\mathcal{P}$, one critical challenge is to split road planes from the 3D scene. State-of-the-art (SOTA) methods [32, 57] use plane segmentation algorithms (e.g., fitting plane equations with RANSAC [19]) to obtain road planes. However, we posit that the road planes obtained through fitting plane equations may inadvertently include non-road objects such as sidewalks, median strips, etc. Therefore, we utilize CENet [7], a concise and efficient point cloud semantic segmentation model, to split the road point cloud from the background point cloud. Subsequently, we meshify the road point cloud to reconstruct the road surface and obtain sample insertion positions and orientations. As shown in Fig. 3, compared to plane-equation based sampling method that is used in the existing testing tool [32], MultiTest can accurately generate possible positions for object insertion, even in complicated road conditions such as intersections.



(a) Invalid insertion with plane-equation based sampling (b) Valid insertion with MultiTest



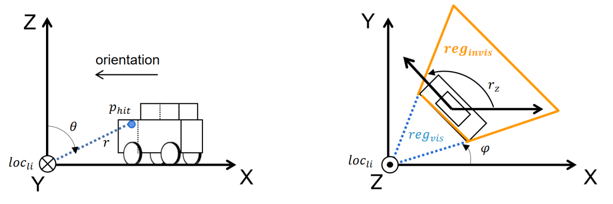
(c) Zoom-in views of different invalid insertion

Figure 3: Validity of object insertion with different pose estimation methods.

Collision Avoidance. To avoid collision with existing objects from the background scene m, we first calculate the minimal 3D bounding box B_o of the object instance to be inserted. Then, given a possible pose of o, $pose\_i$, we check if the 3D box B_o contains any other objects from the background point clouds. If the IOU between B_o and the background point cloud is greater than 0, we consider $pose\_i$ as an invalid pose since it leads to a collision with the existing objects.

3.2 Physical-Aware Multi-Sensor Simulation

One significant threat to the realism of test cases generated by the existing methods is that the random insertion might violate the basic laws of the physical world. For example, images captured



(a) View along the y-axis (b) Top view

Figure 4: Virtual LiDAR emitting lasers to hit a car. (a) View along the y-axis, (b) top view.

by a camera follow the basic law that objects farther away are smaller. Similarly, point clouds recorded by LiDAR should become sparser when the inserted object is farther away. Moreover, different sensor configurations could result in varying field of views (FOVs) and sensing results. Therefore, when rendering an object for insertion, it is critical to use the same sensor settings (camera resolution, number of LiDAR beams, calibration files, etc.) as those used for the background scene. Consequently, MultiTest employs a physical-aware sensor simulation module to render images and point clouds for the object to be inserted. We first construct a set of virtual sensors to simulate their operating modes in the physical world to generate modality-consistent data given their configuration parameters. Then, we carefully handle the occlusion when inserting the object o into the candidate scene m.

LiDAR Simulation. To simulate laser beams of a LiDAR sensor, given the LiDAR horizontal field of view, vertical field of view, the number of beams and angular resolution, we first create a set of rays centered on the virtual LIDAR scanner to simulate each laser beam emitted. Then we leverage the *ray casting* proved by Open3D [60] to simulate laser emission on the object o. As shown in Fig. 4, the ray casting calculates the 3D coordinates of the first hit point $p_{hit}(r, \theta, \phi)$ of the ray on the target surface given the starting point loc_lidar and direction of the ray, where r is the distance, θ is the polar angle and ϕ is the azimuthal angle in spherical coordinates. Whenever an obstacle is inserted into the scene, we execute the ray casting to obtain the updated coordinates in the point cloud.

Considering that the point cloud data captured by a real-world LiDAR could include inevitable noises, we randomly remove a small portion of the point to simulate the laser which is not to be detected by the receiver due to insufficient intensity [28]. We further add Gaussian noise to simulate the measurement noise of the sensor [16]. Note that the proposed virtual LiDAR can simulate any type of laser scanner with the given LiDAR configurations, including its extrinsic calibration, FOVs and resolutions, the number of beams, etc. We refer readers to our supplementary website for more details.

Camera Simulation. We leverage Python API provided by Blender, an open-source 3D computer graphics software, to build our virtual camera sensor. We first build a virtual camera given its configurations, including intrinsic and extrinsic calibration, lens length, resolution, etc. Then, we calculate the pose of the object o in the camera coordinates system $pos\_cam = \mathbf{T}_{\text{velo}}^{\text{cam}} pos\_li$ based on the transformation matrix $\mathbf{T}_{\text{velo}}^{\text{cam}}$ and its corresponding position $pose\_i$ in the LiDAR coordinate. Finally, we place the object o in the position pos_cam and render an image with our virtual camera. Note that we use the minimum 2D rectangle box of the image object o as the ground truth bounding box for 2d object detection. To further improve the realism of the synthesized test cases, we leverage a model S^{2}CRNet [31] with pre-trained weights released by the authors [1] to naturally blend the object o into the target scene m by adjusting the color of the inserted object. Therefore, the inserted object could have near consistent illumination and color balancing compared with the target scene.

Occlusion Handling. When inserting an object, occlusion should be carefully addressed. For instance, the inserted object should not block any objects that are closer to the sensor according to the distance to the sensor. To handle the point cloud occlusion, we remove the point clouds that are occluded by the inserted objects from the 3D scene according to their geometric relationships. As shown in Fig. 4 (b), the ray emitted from the virtual laser scanner intersects the object and divides the space into the visible region reg_vis and the invisible region reg_invis. We sequentially infer the occlusion relation and remove the points in reg_vis for each object according to their distance to the virtual LiDAR. To handle the image occlusion, we record the distance from the center of each object to the virtual camera and ensure the inserted object o only blocks the pixels that are farther than o in the 3D space. To avoid false positives on original objects that are occluded by the inserted objects, we calculate an occlusion ratio for each existing object after each round of insertion. If the occlusion ratio of an occluded object is higher than 90%, we exclude this object from the evaluation by labeling it as "DontCare."

3.3 Metamorphic Relations

Manual labeling of our generated test cases could be time-consuming and labor-extensive. To address this, we leverage metamorphic relations (MRs) [5] to create test oracles. MRs describes the necessary properties of a target software in terms of inputs and their expected outputs [62]. The violation of MRs often indicates potential defects.

MultiTest is designed for MSF perception systems based on metamorphic testing. Specifically, we denote the MSF perception system as PS that detect the results with multi-model data $m \in \mathbb{M}$ including 2D image and 3D point cloud. Given a set of object instances $\mathbb{O}$, an MR can be formalized as follows:

$$MR_1 : \forall o \in \mathbb{O} \wedge \forall m \in \mathbb{M}, \zeta\{PS[\![m]\!] \cup GT_o, PS[\![\sigma(m,o)]\!]\} \quad (5)$$

where σ is the insertion operator for inserting an object instance o into the background scene m, GT_o is the ground truth of the object o (i.e., the estimated bounding box in object detection task) and ζ is a criterion asserting the equality of PS results.

MR_1 is built upon the following two facts: (1) the object insertion operator should not change the correct prediction of PS; (2) and the inserted object should be detected correctly. However, asserting the equality of PS outputs is too strict and thus can lead to a large number of false positives due to a slight drift of the detection results. Therefore, we follow the previous work [23, 52] to leverage soft equality criteria ζ derived from Average Precision (AP). Given

[1]https://github.com/vinthony/S2CRNet-demos

the MR, we can simply obtain the test oracle information without manual annotation by checking if the MR is violated.

3.4 Fitness-Guided Testing

To boost the testing efficiency of MultiTest, we propose a fitness-guided testing process for object insertion. Thus, we can keep a test case m' with high fault-revealing capability from the seed test m.

Fitness Metric. We design a fitness metric that measures the likelihood of a test data to reveal errors. Our fitness metric consists of three fault-revealing capability scores, i.e., object missing, fault detection, and location error (introduced in Sec. 2.2).

Given a ground-truth bounding box $B_g \in \mathbb{GT}$ and a predicted bounding box $B_p \in \mathbb{DT}$, the object missing score can be expressed as follows:

$$F_{OM} = \sum_{B_g \in \mathbb{GT}} I_{OM}(B_g, B_p) * \left(1 - \frac{\min(dis(B_g), dis\_max)}{dis\_max}\right) \tag{6}$$

where dis calculates the distance between a bounding box B and the LiDAR position, dis_max is the max recognition distance of LiDAR, and $I_{OM}(\cdot)$ is an indicator function equal to 1 if and only if it is an object missing error. The intuition behind the score is that the close object missing failure could lead to serious consequences, such as collisions.

The fault detection score can be expressed as follows:

$$F_{FD} = \sum_{B_p \in \mathbb{DT}} I_{FD}(B_g, B_p) * \left(1 - \frac{\min(dis(B_p), dis\_max)}{dis\_max}\right) * prob_p \tag{7}$$

where $prob_p$ is the confidence probability of the detection, and $I_{FD}(\cdot)$ is an indicator function equal to 1 if and only if it is a false detection error. Similarly, a close and high-confidence fault detection might lead to wrong control decisions, e.g., emergency braking.

Then, we compute location error scores if and only if it is neither missing detection nor fault detection. Let $\mathbb{GT}_r = \{B_g | I_{OM}(B_g) = 0 \land B_g \in \mathbb{GT}\}$ denotes the cases that do not contain object missing errors and $\mathbb{DT}_r = \{B_p | I_{FD}(B_p) = 0 \land B_p \in \mathbb{DT}\}$ denotes the cases that do not contain fault detection errors:

$$F_{LE} = \max_{B_g \in \mathbb{GT}_r \land B_p \in \mathbb{DT}_r} 1 - IOU(B_p, B_g) \tag{8}$$

This score indicates that a larger difference between B_p and B_g could reveal a more serious detection error.

Finally, our fitness metric can be expressed as a weighted sum of the three scores:

$$Fitness(m) = \alpha * F_{OM} + \beta * F_{FD} + \gamma * F_{LE} \tag{9}$$

where $\alpha + \beta + \gamma = 1$. Similar to the traditional software testing approach to increase the code coverage rate, MultiTest attempts to generate a test set that can increase this fitness score.

Testing Workflow. Algorithm 1 presents the testing workflow of MultiTest guided by our fitness metric. The algorithm takes an MSF perception system PS, a set of seed multi-model data $\mathbb{M}$, an object database $\mathbb{O}$, and the constant maximum number of trials TRY_NUM and object insertion N as input. The goal is to create a set of critical test cases $\mathbb{T}$ with corresponding ground-truth $\mathbb{G}$ for PS. The algorithm first selects a valid pose of object instance from the *pose generator* (Lines 8-10). Then, the sensor simulation module renders the object-level point cloud and image before inserting them into the background data to obtain synthetic data m' (Lines 11-14). After insertion, we generate the label of test case m' (Line 15) and calculate the fitness score by Eq. 9 (Line 16). If the fitness metric increases, the generated test is retained as a new background scene for the next iteration of object insertion (Lines 17-21). By contrast, the generated data is directly discarded, which indicates that PS might not be prone to erroneous behavior in this case. Finally, a successfully generated test case m' with the label L generated by N iterations of object insertion is added to $\mathbb{T}$ and $\mathbb{G}$, respectively.

Algorithm 1: Fitness-guided testing of MultiTest.

```
Input: The test MSF system $PS$, the set of seed multi-model data $\mathbb{M}$,
       the object database $\mathbb{O}$, the maximum of object insertion $N$,
       the maximum number of trials $TRY\_NUM$.
Output: the set of generated data $\mathbb{T}$, the set of ground-truth $\mathbb{G}$.
1  $\mathbb{T} \leftarrow \emptyset$;
2  for $m$ in $\mathbb{M}$ do
3      $L,\ succFlag \leftarrow LoadLabel(m),\ false$;
4      $finess\_init \leftarrow Fitness(m)$;
5      for $i = 1, 2, \ldots, N$ do
6          Randomly sample an object instance $o$ from $\mathbb{O}$;
7          for $i = 1, 2, \ldots, TRY\_NUM$ do
8              $pose \leftarrow PoseGenerator(o, m)$   ▹ pose estimation;
9              if $CollisionAvoidance(pose, m)$ then
10                 continue;
11             $pc \leftarrow LidarSimulator(o, m)$   ▹ sensor simulation;
12             $image \leftarrow CameraSimulator(o, m)$;
13             $m' \leftarrow Insertion(pose, pc, image, m)$;
14             $m' \leftarrow OcclusionHandler(m')$;
15             $lb' \leftarrow LabelGenerator(o, pose_o, m')$;
               // save test $m'$ with higher fitness
16             $finess\_score \leftarrow Fitness(m')$;
17             if $finess\_score > finess\_init$ then
18                 $succFlag,\ m \leftarrow True,\ m'$;
19                 $L \cup \{lb'\}$;
20                 $finess\_init \leftarrow finess\_score$;
21                 break;
22     if $succFlag$ then
23         $\mathbb{T},\ \mathbb{G} \leftarrow \mathbb{T} \cup \{m'\},\ \mathbb{G} \cup \{L\}$;
24 Return: $\{\mathbb{T}, \mathbb{G}\}$ ;
```

4 EXPERIMENTAL SETUP

In this section, we introduce our experimental setup, including our implementation details, perception systems under test, datasets, and the research questions we investigate.

4.1 Implementation Details

In all experiments, we set the maximum number of trails TRY_NUM as 5 and the maximum number of insertions N as 3 in Alg. 1. α, β, γ in Eq. 9 are set to 0.5, 0.25, 0.25, respectively. We leverage the configurations of Velodyne HDL-64E lidar and PointGray Flea2 color camera provided by KITTI [22] to build our virtual simulator in Sec. 3. To conduct the experiments, we implement the MultiTest

Table 1: Perception systems under test and their performance on KITTI benchmark.

SUT	Fusion	Year	2D Det. AP	3D Det. AP
CLOCs [39]	Late fusion	2020	89.82	89.37
EPNet [27]	Deep fusion	2020	89.84	89.54
FConv [54]	Weak fusion	2019	90.19	90.42
★ Second [57]	–	2018	–	89.09
★ CasRCNN [2]	–	2018	90.23	–

upon PyTorch 1.8 and Python 3.7. All experiments are conducted on a server with an Intel i7-10700K CPU (3.80 GHz), 48 GB RAM, and an NVIDIA RTX 3070 GPU (8 GB VRAM).

4.2 Perception Systems Under Test

To evaluate the effectiveness and efficiency of MultiTest, we choose three SOTA camera-LiDAR fusion perception systems covering different fusion mechanisms from an MSF benchmark [21] as the SUTs (system under test). Additionally, we consider one single-sensor detection system for both camera and LiDAR in our experiment. We compare five perception systems and their original performance (AP) on KITTI 2D/3D detection benchmark in Table 1.

4.3 Dataset

We select our test seeds from KITTI dataset [22]. KITTI is one of the most popular autonomous driving datasets, which provides images and point clouds, with detailed sensor configurations. KITTI's data are collected by four high-resolution cameras, a Velodyne HDL-64E LiDAR and an advanced positioning system from multiple categories real-world driving scenarios, such as cities, residential areas, and roads. The KITTI object detection dataset contains 7481 pairs of image and point cloud data with ground-truth labels. The ground-truth labels include both annotated 2D and 3D boxes, difficulty levels and category labels for objects of interest. In this paper, we focus on the detection of car objects at the moderate difficulty level.

For our object databse $\mathbb{O}$, we utilize ShapeNet [4]. ShapeNet is a richly-annotated and large-scale dataset of 3D object models. It contains over 220,000 object-level models from different categories. In this paper, we build a multi-modal object database from the *car* category in ShapeNet, which consists of 3483 objects. We further filtered out damaged models (e.g., empty model files) and uncommon vehicles on the road (e.g., a racing car). Consequently, our object database includes 1674 vehicles from five categories: *sedan*, *coupe*, *suv*, *cab*, and other unclassified cars.

4.4 Research Questions

To evaluate MultiTest's performance, we conduct both quantitative and qualitative experiments to answer the following three research questions (RQs):

- RQ1. How effective is the MultiTest at synthesizing realistic multi-modal data?
- RQ2. How effective is the MultiTest at generating error-reveling tests?
- RQ3. How effective is the MultiTest at guiding the improvement of a SUT through retraining?

To answer **RQ1**, we verify the effectiveness of two modules in MultiTest's multi-modal data synthesis pipeline: *Pose Estimation* and *Multi-sensor Simulation*. We replace each corresponding module with a baseline module to conduct control experiments. Specifically, we replace MultiTest's *Pose Estimation* module with a *Random Pose* module and MultiTest's *Multi-sensor Simulation* module with *Cut&Paste* module. The *Random Pose* module randomly selects a pose for the object o without any constraints. The *Cut&Paste* module directly copies an object's image and point cloud from other data frames in the KITTI dataset without any re-rendering. Hence, we can obtain four multi-modal data synthesis pipelines. We denote them as (1) $C\&P + Pose_{ran}$, (2) $C\&P + Pose_{est}$, (3) $Sim + Pose_{ran}$, and (4) $Sim + Pose_{est}$ (MultiTest). We then conduct both quantitative and qualitative assessments to verify the data realism.

Quantitative Assessment. We randomly select 200 data pairs from KITTI's validation dataset as the initial seeds for each pipeline and compared the realism of the data generated by each pipeline. Then we utilize **FID** (Frechet Inception Distance) [25] and **FRD** (Frechet Range Distance) [63] to measure the realism of the image and point cloud, respectively. **FID [25]** evaluates the squared Wasserstein distance between feature vectors extracted from the inception-v3 [50] model from the generated samples G and real samples R. The computation of FID can be represented as:

$$\mathrm{FID}(\mathrm{R},\mathrm{G}) = \left\|\mu_r - \mu_g\right\|^2 + \mathrm{Tr}\left(\Sigma_r + \Sigma_g - 2\left(\Sigma_r\Sigma_g\right)^{1/2}\right) \tag{10}$$

where μ and Σ represent the mean values and covariance matrix of generated samples respectively and Tr denotes the trace operator of the matrix. Similar to FID, **FRD [63]** measures the similarity between two sets of point cloud data using the feature vectors from a pre-trained RangeNet++ [37]. Note that we set the configurable parameters of these metrics with default/recommended settings.

We further propose a new **Modality-Consistency (MC)** metric, which is specifically designed for our multi-modal object insertion. For each inserted object $o \in O$, we first calculate its minimum bounding box in the point cloud and project it onto the image as B_g^p. Then, we measure the average IOU between 2D bounding boxes of an image B_g^i and projected bounding boxes B_g^p through:

$$MC = \frac{1}{|O|} * \sum_{o \in O} (IOU(B_g^i, B_g^p)) \tag{11}$$

Intuitively, this metric measures if the inserted object has consistent poses and dimensions in both image and point cloud data.

To better demonstrate the realism of test data generated by different pipelines, we also implement two SOTA object-insertion based testing methods for single-sensor perception systems (i.e., MetaOD [52] (camera-based) and TauLim [32] (LiDAR-based)) and leverage their combination as another comparison baseline for our quantitative assessment in this RQ.

Qualitative Assessment. We further conduct a user study to qualitatively assess the naturalness of MultiTest's generated multi-modal data. We recruited sixteen participants through the mailing list of the CS department at a research university. All participants had a minimum of a master's degree in SE/CS. Seven out of sixteen participants had more than two years of experience in the field of autonomous driving. During the study, we randomly selected twenty data instances as test seeds. We then ask a participant

Table 2: Realism of test data generated by different pipelines.

Data Generation	Image FID ↓	Point Cloud FRD ↓	Consistency MC ↑
$C\&P + Pose_{ran}$	106.13	148.24	0.31
$C\&P + Pose_{est}$	107.62	115.62	0.30
$Sim + Pose_{ran}$	91.34	111.14	0.83
$MetaOD + TauLim$	65.57	133.61	0.01
$Sim + Pose_{est}$ (MultiTest)	86.89	88.87	0.82

to rank the multi-modal data synthesized by four different pipelines from each seed through a questionnaire. For each of the twenty data instances, a participant needs to rank the data quality from three perspectives: (1) the image's naturalness, (2) the point cloud's naturalness, and (3) the modality-consistency between a pair of image and point cloud. To mitigate bias, we randomly assigned the order of data synthesized by different pipelines for each test seed. We further refer readers to our supplementary website for the complete questionnaire. After finishing the user study, we use the one-side Wilcoxon rank-sum test [11] to verify if any of the four pipelines is significantly preferred by the participants.

To answer **RQ2**, we utilize random testing as a baseline to evaluate the effectiveness of our proposed fitness-guided testing strategy. We also include MetaOD [52] (camera-based), TauLim [32] (LiDAR-based), and their combinations as the comparison baselines. Specifically, we randomly select 200 data instances as the initial seeds from the validation set of the KITTI dataset to generate test cases using both random testing and MultiTest's fitness-guided testing. We then calculate the AP difference between the original datasets and the generated test cases of a perception system under test for each configuration. Furthermore, we count the number of errors with each error category, i.e., *object missing*, *false detection*, and *location error* (see Sec .2.2). Note that we only count the errors when a perception system has a confidence level greater than 0.5 for the purpose of mitigating its trivial faults. Each experiment is repeated five times to mitigate the effect of randomness.

To answer **RQ3**, we retrain all five perception systems on the corresponding test cases generated by MultiTest or other baselines in RQ2. For each system, we randomly generate 1000 test cases for each perception system. Then we select 20% of the test cases and add them to the original KITTI training set. We keep the training configuration consistent with each system's original settings and parameters. We set the number of epochs as 40 for retraining. After retraining, we compare the performance of each retrained perception system with its original counterpart on the remaining 80% of generated test cases. We retrain each system for five times to mitigate the effect of randomness and compare its average performance.

5 RESULT ANALYSIS

5.1 RQ1: Realism Validation

Table 2 shows the quantitative results of different data synthesis pipelines. Compared with baselines, MultiTest achieves the second-lowest FID and the lowest FRD. These results indicate that MultiTest can generate high quality and realistic image and point cloud data after the object insertion. We notice that

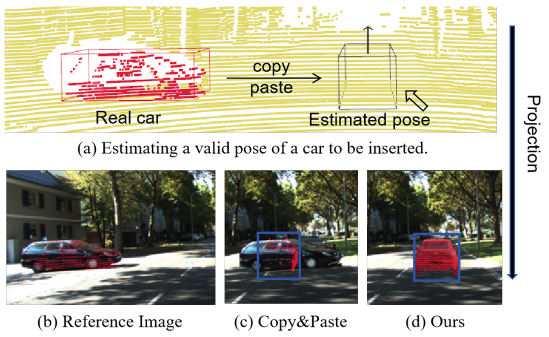


Figure 5: Modality consistency of data synthesized with different simulation methods.

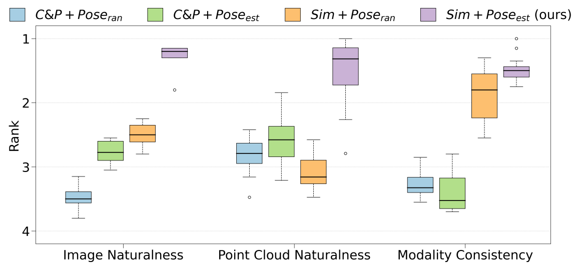


Figure 6: Participants' choices over different data synthesis pipelines in *image naturalness*, *point cloud naturalness*, and *modality consistency*.

MetaOD+TauLim achieves better image quality (FID). A plausible explanation is that MetaOD only inserts one object per image while MultiTest may insert multiple objects. Our further investigation shows that when only inserting one object, MultiTest can generate images with similar quality (FID: 62.53). However, the combination of MetaOD and TauLim produces the lowest MC, indicating that directly combining two single-sensor testing methods may produce data with high inconsistency. We also find that compared with the *Copy&Paste* method, our *Multi-sensor Simulation* module can generate much more modality-consistent data pairs of images and point clouds. A plausible explanation is that *Copy&Paste* does not take sensor's position and inserted object's pose into account. As the example shown in Fig. 5 (a), given an estimated pose to insert a car object, *Copy&Paste* directly copies a car object from existing data (Fig. 5 (b)). However, the image data can not be rotated to align with the given pose, resulting in modality inconsistency (Fig. 5 (c)) compared with MultiTest (Fig. 5 (d)). Furthermore, we find that our *Pose Estimation* module outperforms the *Random Pose* on the naturalness of the generated point cloud. By replacing *Random Pose* with MultiTest's *Pose Estimation*, the FRD values decreased by 22% and 21% for *Copy&Paste* and *Multi-sensor Simulation*, respectively. This is largely attributed to the fact that *Random Pose* does not take physical constraints into account when inserting objects. Consequently, it inevitably inserts objects into invalid positions.

Fig. 6 shows the participants' preference among four different data synthesis pipelines in terms of the *image naturalness*, *point cloud naturalness*, and *modality consistency* between image and

Table 3: Testing results of five perception systems with with different guidance approaches.

Metric	Method	3D Object Detection					2D Object Detection				
		EPNet	FConv	CLOCs	⋆ Second	Avg.	EPNet	FConv	CLOCs	⋆ CasRCNN	Avg.
AP Difference ↑	MetaOD	0.05	1.84	0.14	—	0.68	0.01	1.21	0.12	1.12	0.62
	TauLim	0.14	2.40	0.28	5.78	2.15	0.01	0.63	0.16	—	0.27
	MetaOD+TauLim	0.17	4.36	2.02	5.86	3.10	0.01	1.49	1.28	1.05	0.96
	MultiTest (Random)	0.52	20.34	5.77	9.15	8.95	1.91	16.89	6.17	20.98	11.49
	MultiTest (Guided)	10.25	32.31	22.64	26.01	22.80	2.84	24.29	16.64	30.09	18.47
Location Error	MetaOD	1	17	7	—	8	2	32	18	35	22
	TauLim	2	15	7	16	10	18	22	18	—	19
	MetaOD+TauLim	2	19	8	17	12	19	21	17	25	21
	MultiTest (Random)	9	83	25	23	35	8	50	21	56	34
	MultiTest (Guided)	11	86	24	21	36	7	70	22	80	45
False Detection	MetaOD	1	89	52	—	47	0	2	1	2	1
	TauLim	15	60	33	56	41	3	2	1	—	2
	MetaOD+TauLim	14	62	37	61	43	2	3	1	2	2
	MultiTest (Random)	10	65	38	67	45	0	1	1	1	1
	MultiTest (Guided)	9	81	39	71	50	0	4	1	3	2
Object Missing	MetaOD	0	12	7	—	7	0	4	3	4	3
	TauLim	4	18	7	1	8	0	2	2	—	2
	MetaOD+TauLim	4	23	10	1	10	0	2	2	2	1
	MultiTest (Random)	31	135	90	34	72	5	75	69	75	56
	MultiTest (Guided)	49	178	153	51	108	12	96	121	90	80

point cloud. We find that more participants prefer MultiTest over the three baseline methods on all three degrees. Wilcoxon’s rank-sum test results suggest that the mean ranking differences between MultiTest and the second-chosen method are statistically significant (p<0.0001). These results suggest that our autonomous driving practitioners agree that MultiTest provides the most realistic and natural synthesized data among all four pipelines.

5.2 RQ2: Fault Detection Capability

Table 3 shows the testing results of five perception systems on 2D/3D object detection tasks. A larger difference in average precision (AP) signifies that a perception system performs considerably worse on the generated test cases compared to the original dataset. Compare with other baselines, we observe a clear and consistent trend of decreased AP performance in MultiTest across various tasks and perception systems. These results affirm that MultiTest is effective in generating critical and challenging test cases. Furthermore, our proposed method proves to be proficient in detecting various categories of errors, particularly in identifying *object missing* errors. However, it is worth noting an exception, where only a small number of false detection errors were detected for 2D object detection. This might be due to the relatively small image size in KITTI compared with the point cloud. As a result, cases where a detected bounding box is disjoint from all ground truth bounding boxes (i.e., IOU=0) are rare.

We also find that compared with the random baseline, our fitness-guided testing leads to a more significant decrease in average precision (AP) compared to the random testing baseline. This finding confirms the effectiveness of MultiTest’s fitness metric and its guidance, establishing that MultiTest is more efficient at generating test cases. Additionally, we observe that MultiTest triggered a higher number of object missing errors compared to the random testing baseline. These results suggest that MultiTest’s fitness-guided strategy improves overall testing efficiency, resulting in the synthesis of more error-revealing test cases.

Table 4: Five perception system’s performance after retraining with different approaches on the generated tests.

Model	Det.	MultiTest (Guided)	MultiTest (Random)	MetaOD	TauLim	MetaOD +TauLim	Original
EPNet	3D	80.87	77.34	75.51	74.37	75.39	75.53
	2D	89.73	88.88	89.04	88.77	88.80	89.11
FConv	3D	82.72	79.30	49.34	65.19	49.88	57.98
	2D	90.63	88.30	62.46	71.90	63.88	70.56
CLOCs	3D	83.63	81.47	68.73	69.79	70.74	62.34
	2D	90.49	89.19	80.36	80.31	80.29	71.10
⋆ Second	3D	76.43	74.89	—	71.04	71.51	65.78
⋆ CasRcnn	2D	94.86	93.20	60.03	—	59.82	60.21
Average AP		86.17	84.07	69.35	74.48	70.04	69.08

5.3 RQ3: Performance Improvement

Table 4 shows the AP performance of all subject perception systems after retraining with test cases generated by MultiTest or other baseline methods. MultiTest can significantly improve the performance of all systems by retraining. This result indicates that the data generated by MultiTest includes challenging cases that can significantly help improve a perception system’s robustness, showing the effectiveness of our testing. Compared with MultiTest, other baseline methods show relatively low capability in improving performance and in some cases can even be harmful. A possible reason is that test cases generated by single-sensor testing methods or their combination fail to maintain modal consistency and therefore hardly benefit in the retraining process.

Moreover, we find that the average AP performance improvement with the fitness-guided testing and random testing across different systems are 86.05 and 83.79, respectively. Though both testing strategies can synthesize data that potentially help improve a perception system's performance, our fitness-guided testing is specifically more effective and efficient. This might largely attribute to the fact that MultiTest's fitness-guided testing can generate more error-revealing test cases.

6 DISCUSSION

Data realism affects testing effectiveness. MultiTest is designed to automatically generate realistic multi-modal data to assess the potential risks of a perception system when deployed in real-world scenarios. Both our quantitative evaluation and user study results confirm the realism of the data synthesized by MultiTest. Experiments with five perception systems further suggest that realistic data facilitates effective and efficient testing. We attribute MultiTest's superior performance to two important designs.

Firstly, our physical-aware object insertion can generate natural yet challenging multi-modal test data for a perception system. Existing mutation-based testing methods usually face challenges that the perturbation (e.g., adversarial noises) might not be naturally exist in the real world. Therefore, it is questionable if the error-revealing test cases generated by these methods could help understand a SUT's potential risks. Compared with existing object insertion testing methods, MultiTest takes real-world physical constraints into account to ensure the semantic validity of its insertion.

Secondly, MultiTest simultaneously generates modality-consistent data across different sensors, which is crucial for testing MSF-based perception systems. Existing testing methods usually focus on single-sensor perception systems. Indeed, a combination of different single-sensor testing methods can also be used for testing an MSF system. However, without ensuring modality-consistency, the validity of the generated test cases becomes questionable. For instance, mis-aligned pairs of images and point clouds are rarely found in real-world autonomous driving applications.

Testing efficiency trade-offs. Different parameter configurations could potentially affect MultiTest's testing efficiency. Intuitively, inserting more objects could create more challenging test cases and potentially reveal more system faults. However, as the number of insertions grows, it might become harder for MultiTest to find possible positions for the insertion. We set up an ablation study to verify MultiTest's testing efficiency trade-offs. Specifically, We set the maximum number of trials TRY_NUM as 10 and the maximum object insertion number N from 1 to 6. We randomly select 50 test seeds from KITTI and record the proportion of seeds that successfully generate test cases with different N and calculate the average number of iterations required per seed.

Table 5 validates our hypothesis. As the number of objects increases, the proportion of successful seeds decreases. About 50% of the seeds are able to insert three objects, and this value drops to about 25% when N is 6. Moreover, the average number of iterations grows exponentially with the number of inserted objects. We further qualitatively check the seeds and find that the complex scenes in autonomous driving (e.g., cars driving on a busy city road) could significantly increase the difficulty of object insertion. In this paper, we set N is 3 to balance the testing efficiency trade-offs.

Table 5: Number of successful seeds and average of iterations for insertion in different numbers of inserted objects.

Model	Metric	Number of inserted objects: N					
		1	2	3	4	5	6
EPNet	Seeds	0.70	0.48	0.28	0.20	0.14	0.12
	Iterations	1.46	6.42	16.93	27.70	43.57	53.00
FConv	Seeds	0.92	0.76	0.60	0.50	0.40	0.30
	Iterations	1.00	3.58	7.30	11.20	16.90	25.87
CLOCs	Seeds	0.74	0.66	0.48	0.34	0.24	0.22
	Iterations	2.03	4.94	11.25	20.53	34.92	41.45
⋆ Second	Seeds	0.84	0.72	0.54	0.40	0.32	0.20
	Iterations	1.29	4.56	10.89	19.65	28.25	50.20
⋆ CasRCNN	Seeds	0.88	0.74	0.56	0.48	0.32	0.26
	Iterations	1.45	4.73	10.25	14.71	27.00	36.54

Limitations and future work. Our MultiTest is able to generate test cases to reveal three different types of faults in object detection. In future work, one may consider controlling the coefficients in Eq. 9 to generate test cases with preferences for different fault types according to the developer's intention. Currently, MultiTest leverages a model, S^2CRNet [31], to improve the naturalness of the synthesized images. It would be worth-while to explore more advanced methods, e.g., SOTA generative AI and diffusion models [45, 48]. Furthermore, we have only experimented MultiTest with object detection systems. One can consider how to leverage MultiTest to test different MSF perception systems, e.g., object tracking systems. Compared with testing object detection, testing object tracking would require the synthesis of sequential data (i.e., a series of temporally correlated data). How to ensure the temporal consistency between data frames and how to generate trajectories for the object to be inserted could be challenging but worth investigating.

Threats to validity. In terms of *construct validity*, one potential threat comes from the measurement of the realism of MultiTest's generated data since there is no ground truth label. To mitigate this, we conduct both quantitative and qualitative experiments to assess the quality of the synthesized data. Another construct threat lies in the randomness inherent in our experiments, specifically in the testing (RQ2) and retraining (RQ3). To combat this, we repeat our experiment on testing and retraining for five times to reduce the influence of randomness.

In terms of *internal validity*, one potential threat is that the data generated by the virtual simulator may differ from real-world. Besides, the quality of 3D models in MultiTest's object database could also affect the quality of synthesized data. To mitigate these, we utilize the well-known simulators (i.e., Blender and Open3D) and select a popular 3D object database ShapeNet.

In terms of *external validity*, one potential threat is that our analysis results may not be generalized to other perception systems. To combat this threat, we experimented with three MSF systems with different fusion mechanisms and two single-sensor detection models to evaluate the effectiveness of MultiTest.

7 RELATED WORKS

Multi-sensor Fusion. MV3D [6] is a pioneering work in the field of AI-enabled multi-sensor fusion. It introduces a multi-view based deep fusion framework that extracts hidden features from different view representations of 3D point clouds and images, enabling region-wise feature fusion. To further enhance system performance, subsequent deep-fusion based methods directly fuse the features extracted from raw data to mitigate information loss caused by point cloud view transformations. For instance, MVX-net [49], a voxel-based fusion method, and EPNet [27], a point-based fusion method, are notable examples of such techniques. In the realm of late fusion, CLOCs [39] is a representative work that fuses the output results of different detectors. DFMOT [53] is another late-fusion based work for object tracking that employs a four-level deep association mechanism to achieve a fast fusion of 2D and 3D trajectories. Weak fusion often involves using 2D proposals as guidance to extract the frustum region from the point cloud [42, 54]. F-PointNets [42] pioneered this approach for object detection, and FConv [54] extended it further by incorporating a post-grouping aggregation scheme, leading to end-to-end estimation and improved performance.

Quality Assurance of Perception Systems. The first group of related works is on automatic testing for visual perception systems [52, 55, 58]. Wang et al. first propose MetaOD [52] to test object detection systems by inserting objects based on metamorphic testing theory. However, this approach faces limitations when it comes to inserting image object instances in valid positions due to the lack of 3D information. Another set of related work focuses on LiDAR-based perception systems robustness testing [8, 23, 32]. Guo et al. [23] and Christian et al. [8] leverage data mutation operators to generate test point clouds and check the failure in perception systems based on metamorphic relationships. However, extending these testing methods to MSF perception systems becomes challenging due to the multi-modal data required by MSF systems.

On the other hand, there are a few works that focus on benchmarking MSF systems [21, 59]. Recently, Gao et al. [21] create an early public benchmark of MSF systems and perform a large-scale empirical study to investigate their robustness performance. Along this direction, we design and implement MultiTest for automated testing of MSF perception systems. We adopt the MSF system designed for object detection tasks provided by the MSF benchmark as our experimental subject. There are also some other works that assess a perception system's quality from the security perspective by inserting adversarial objects [1, 3, 51].

General Deep Learning System Testing. Inspired by the effectiveness of code coverage applied to traditional software testing, researchers have proposed several neuron coverage criteria to guide the testing process of Deep Learning (DL) systems. Pei et al. [40] first propose the neuron coverage criterion to measure the extent of state exploration in deep neural networks. Ma et al. [36] further refined the neuron coverage criterion and proposed a set of fine-grained testing criteria. With the guidance of the neuron coverage criterion, several test generation techniques have been proposed to detect erroneous behaviors of DNNs by increasing the neuron coverage [14, 56]. To detect erroneous behaviors of DL systems effectively, several domain-specific guidance metrics are proposed [18, 20]. DeepGini leverages Gini impurity to measure the fault-revealing capabilities of test cases [18]. In contrast, our work takes a different approach by focusing on a context-specified problem, specifically on the testing of real-world MSF perception systems. This specialized focus allows us to address the unique challenges and requirements posed by the systems under test.

8 CONCLUSION

In this paper, we present MultiTest, the first automated testing tool designed specifically for MSF perception systems. MultiTest employs a physical-aware approach to render modality-consistent object instances using virtual sensors. Then, MultiTest synthesizes realistic images and point clouds by inserting object instances into valid yet challenging positions within the target scene. Moreover, MultiTest incorporates fitness metric guidance to boost the testing efficiency and effectiveness. We evaluate the performance of MultiTest using three state-of-the-art MSF-based detectors and two single-sensor based detectors. The experimental results demonstrate that MultiTest efficiently detects erroneous behavior in the systems under test and improves a system's robustness through retraining. In the future, we plan to extend the capabilities of MultiTest to support a wider range of MSF perception systems and tasks, such as object tracking and depth completion. By doing so, we aim to broaden its applicability and impact in the field of multi-sensor fusion system testing.

ACKNOWLEDGEMENT

We would like to thank anonymous reviewers for their constructive comments and feedback. This project was partially funded by the National Natural Science Foundation of China under Grant No.61932012, No.61832009 and No.62002158. This work was also supported in part by Canada CIFAR AI Chairs Program, the Natural Sciences and Engineering Research Council of Canada (NSERC No.RGPIN-2021-02549, No.RGPAS-2021-00034, No.DGECR-2021-00019); JST-Mirai Program Grant No.JPMJMI20B8, JSPS KAKENHI Grant No.JP21H04877, No.JP23H03372; as well as the support from TIER IV, Inc. and Autoware Foundation.

REFERENCES

[1] Mazen Abdelfattah, Kaiwen Yuan, Z. Jane Wang, and Rabab Ward. 2021. Towards Universal Physical Attacks On Cascaded Camera-LiDAR 3D Object Detection Models. In *2021 IEEE International Conference on Image Processing (ICIP)*. 3592–3596. https://doi.org/10.1109/ICIP42928.2021.9506016

[2] Zhaowei Cai and Nuno Vasconcelos. 2018. Cascade R-CNN: Delving Into High Quality Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[3] Yulong Cao, Ningfei Wang, Chaowei Xiao, Dawei Yang, Jin Fang, Ruigang Yang, Qi Alfred Chen, Mingyan Liu, and Bo Li. 2021. Invisible for both Camera and LiDAR: Security of Multi-Sensor Fusion based Perception in Autonomous Driving Under Physical-World Attacks. In *2021 IEEE Symposium on Security and Privacy (SP)*. 176–194. https://doi.org/10.1109/SP40001.2021.00076

[4] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. 2015. ShapeNet: An Information-Rich 3D Model Repository. *arXiv preprint arXiv:1512.03012* (2015).

[5] Tsong Yueh Chen, Fei-Ching Kuo, Huai Liu, Pak-Lok Poon, Dave Towey, TH Tse, and Zhi Quan Zhou. 2018. Metamorphic Testing: A Review of Challenges and Opportunities. *ACM Computing Surveys (CSUR)* 51, 1 (2018), 1–27.

[6] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. 2017. Multi-View 3D Object Detection Network for Autonomous Driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1907–1915.

[7] Hui-Xian Cheng, Xian-Feng Han, and Guo-Qiang Xiao. 2022. CENet: Toward Concise and Efficient LiDAR Semantic Segmentation for Autonomous Driving. In *2022 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 01–06.

[8] Garrett Christian, Trey Woodlief, and Sebastian Elbaum. 2023. Generating Realistic and Diverse Tests for LiDAR-Based Perception Systems. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. 2604–2616. https://doi.org/10.1109/ICSE48619.2023.00217

[9] CommaAI. [n. d.]. Openpilot. https://github.com/commaai/openpilot.

[10] Yaodong Cui, Ren Chen, Wenbo Chu, Long Chen, Daxin Tian, Ying Li, and Dongpu Cao. 2022. Deep Learning for Image and Point Cloud Fusion in Autonomous Driving: A Review. *IEEE Transactions on Intelligent Transportation Systems* 23, 2 (2022), 722–739. https://doi.org/10.1109/TITS.2020.3023541

[11] Jack Cuzick. 1985. A Wilcoxon-type test for trend. *Statistics in medicine* 4, 1 (1985), 87–90.

[12] Alex Davies. 2019. Tesla's Latest Autopilot Death Looks Just Like a Prior Crash. https://www.wired.com/story/teslas-latest-autopilot-death-looks-like-prior-crash/.

[13] César Debeunne and Damien Vivet. 2020. A Review of Visual-LiDAR Fusion based Simultaneous Localization and Mapping. *Sensors* 20, 7 (2020), 2068.

[14] Xiaoning Du, Xiaofei Xie, Yi Li, Lei Ma, Yang Liu, and Jianjun Zhao. 2019. DeepStellar: Model-based quantitative analysis of stateful deep learning systems. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 477–487.

[15] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2010. The Pascal Visual Object Classes Challenge: A Retrospective. *International journal of computer vision* 88, 2 (2010), 303–338.

[16] Jin Fang, Xinxin Zuo, Dingfu Zhou, Shengze Jin, Sen Wang, and Liangjun Zhang. 2021. LiDAR-Aug: A General Rendering-Based Augmentation Framework for 3D Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4710–4720.

[17] Di Feng, Christian Haase-Schütz, Lars Rosenbaum, Heinz Hertlein, Claudius Glaeser, Fabian Timm, Werner Wiesbeck, and Klaus Dietmayer. 2020. Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges. *IEEE Transactions on Intelligent Transportation Systems* 22, 3 (2020), 1341–1360.

[18] Yang Feng, Qingkai Shi, Xinyu Gao, Jun Wan, Chunrong Fang, and Zhenyu Chen. 2020. DeepGini: prioritizing massive tests to enhance the robustness of deep neural networks. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 177–188.

[19] Martin A Fischler and Robert C Bolles. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24, 6 (1981), 381–395.

[20] Xinyu Gao, Yang Feng, Yining Yin, Zixi Liu, Zhenyu Chen, and Baowen Xu. 2022. Adaptive test selection for deep neural networks. In *Proceedings of the 44th International Conference on Software Engineering*. 73–85.

[21] Xinyu Gao, Zhijie Wang, Yang Feng, Lei Ma, Zhenyu Chen, and Baowen Xu. 2023. Benchmarking Robustness of AI-Enabled Multi-Sensor Fusion Systems: Challenges and Opportunities. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2023)*. 871–882.

[22] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 3354–3361.

[23] An Guo, Yang Feng, and Zhenyu Chen. 2022. LiRTest: augmenting LiDAR point clouds for automated testing of autonomous driving systems. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*. 480–492.

[24] Abhishek Gupta and Xavier Fernando. 2022. Simultaneous Localization and Mapping (SLAM) and Data Fusion in Unmanned Aerial Vehicles: Recent Advances and Challenges. *Drones* 6, 4 (2022), 85.

[25] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *Advances in neural information processing systems* 30 (2017).

[26] Keli Huang, Botian Shi, Xiang Li, Xin Li, Siyuan Huang, and Yikang Li. 2022. Multi-modal Sensor Fusion for Auto Driving Perception: A Survey. *arXiv preprint arXiv:2202.02703* (2022).

[27] Tengteng Huang, Zhe Liu, Xiwu Chen, and Xiang Bai. 2020. EPNet: Enhancing Point Features with Image Semantics for 3D Object Detection. In *European Conference on Computer Vision*. Springer, 35–52.

[28] Alireza G Kashani, Michael J Olsen, Christopher E Parrish, and Nicholas Wilson. 2015. A Review of LIDAR Radiometric Processing: From *Ad Hoc* Intensity Correction to Rigorous Radiometric Calibration. *Sensors* 15, 11 (2015), 28099–28128.

[29] Edward Kim, Divya Gopinath, Corina Pasareanu, and Sanjit A. Seshia. 2020. A Programmatic and Semantic Approach to Explaining and Debugging Neural Network Based Object Detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[30] Guillaume Leclerc, Hadi Salman, Andrew Ilyas, Sai Vemprala, Logan Engstrom, Vibhav Vineet, Kai Xiao, Pengchuan Zhang, Shibani Santurkar, Greg Yang, et al. 2022. 3DB: A Framework for Debugging Computer Vision Models. *Advances in Neural Information Processing Systems* 35 (2022), 8498–8511.

[31] Jingtang Liang, Xiaodong Cun, Chi-Man Pun, and Jue Wang. 2022. Spatial-Separated Curve Rendering Network for Efficient and High-Resolution Image Harmonization. In *European Conference on Computer Vision*. Springer, 334–349.

[32] Ju Lin, Jiawei Liu, Quanjun Zhang, Xufan Zhang, and Chunrong Fang. 2022. TauLiM: test data augmentation of LiDAR point cloud by metamorphic relation. In *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*. 217–221.

[33] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. SSD: Single Shot MultiBox Detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 21–37.

[34] Ze Liu, Han Hu, Yue Cao, Zheng Zhang, and Xin Tong. 2020. A Closer Look at Local Aggregation Operators in Point Cloud Analysis. *ECCV* (2020).

[35] Guannan Lou, Yao Deng, Xi Zheng, Mengshi Zhang, and Tianyi Zhang. 2022. Testing of autonomous driving systems: where are we and where should we go?. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 31–43.

[36] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, et al. 2018. DeepGauge: multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE international conference on automated software engineering*. 120–131.

[37] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. 2019. RangeNet ++: Fast and Accurate LiDAR Semantic Segmentation. In *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 4213–4220.

[38] Rafael Padilla, Sergio L Netto, and Eduardo AB Da Silva. 2020. A Survey on Performance Metrics for Object-Detection Algorithms. In *2020 international conference on systems, signals and image processing (IWSSIP)*. IEEE, 237–242.

[39] Su Pang, Daniel Morris, and Hayder Radha. 2020. CLOCs: Camera-LiDAR Object Candidates Fusion for 3D Object Detection. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 10386–10393.

[40] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. In *proceedings of the 26th Symposium on Operating Systems Principles*. 1–18.

[41] Zi Peng, Jinqiu Yang, Tse-Hsun (Peter) Chen, and Lei Ma. 2020. A First Look at the Integration of Machine Learning Models in Complex Autonomous Driving Systems: A Case Study on Apollo *(ESEC/FSE 2020)*. Association for Computing Machinery, New York, NY, USA, 1240–1250. https://doi.org/10.1145/3368089.3417063

[42] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. 2018. Frustum PointNets for 3D Object Detection From RGB-D Data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 918–927.

[43] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2017. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[44] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *Advances in neural information processing systems* 30 (2017).

[45] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical Text-Conditional Image Generation with CLIP Latents. *arXiv preprint arXiv:2204.06125* (2022).

[46] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An Incremental Improvement. *arXiv preprint arXiv:1804.02767* (2018).

[47] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc.

[48] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-Resolution Image Synthesis With Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10684–10695.

[49] Vishwanath A Sindagi, Yin Zhou, and Oncel Tuzel. 2019. MVX-Net: Multimodal VoxelNet for 3D Object Detection. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 7276–7282.

[50] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[51] James Tu, Huichen Li, Xinchen Yan, Mengye Ren, Yun Chen, Ming Liang, Eilyan Bitar, Ersin Yumer, and Raquel Urtasun. 2022. Exploring Adversarial Robustness of Multi-sensor Perception Systems in Self Driving. In *Proceedings of the 5th Conference on Robot Learning (Proceedings of Machine Learning Research, Vol. 164)*, Aleksandra Faust, David Hsu, and Gerhard Neumann (Eds.). PMLR, 1013–1024. https://proceedings.mlr.press/v164/tu22a.html

[52] Shuai Wang and Zhendong Su. 2020. Metamorphic object insertion for testing object detection systems. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. 1053–1065.

[53] Xiyang Wang, Chunyun Fu, Zhankun Li, Ying Lai, and Jiawei He. 2022. DeepFusionMOT: A 3D Multi-Object Tracking Framework Based on Camera-LiDAR

Fusion with Deep Association. *arXiv preprint arXiv:2202.12100* (2022).

[54] Zhixin Wang and Kui Jia. 2019. Frustum ConvNet: Sliding Frustums to Aggregate Local Point-Wise Features for Amodal 3D Object Detection. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1742–1749.

[55] Xiaoyuan Xie, Ying Duan, Songqiang Chen, and Jifeng Xuan. 2022. Towards the Robustness of Multiple Object Tracking Systems. In *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 402–413.

[56] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon See. 2019. DeepHunter: a coverage-guided fuzz testing framework for deep neural networks. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 146–157.

[57] Yan Yan, Yuxing Mao, and Bo Li. 2018. SECOND: Sparsely Embedded Convolutional Detection. *Sensors* 18, 10 (2018), 3337.

[58] Zhiyi Zhang, Pu Wang, Hongjing Guo, Ziyuan Wang, Yuqian Zhou, and Zhiqiu Huang. 2021. DeepBackground: Metamorphic testing for Deep-Learning-driven image recognition systems accompanied by Background-Relevance. *Information and Software Technology* 140 (2021), 106701.

[59] Ziyuan Zhong, Zhisheng Hu, Shengjian Guo, Xinyang Zhang, Zhenyu Zhong, and Baishakhi Ray. 2022. Detecting multi-sensor fusion errors in advanced driver-assistance systems. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*. 493–505.

[60] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2018. Open3D: A modern library for 3D data processing. *arXiv preprint arXiv:1801.09847* (2018).

[61] Yin Zhou and Oncel Tuzel. 2018. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[62] Zhi Quan Zhou, DH Huang, TH Tse, Zongyuan Yang, Haitao Huang, and TY Chen. 2004. Metamorphic testing and its applications. In *Proceedings of the 8th International Symposium on Future Software Technology (ISFST 2004)*. Software Engineers Association Xian, China, 346–351.

[63] Vlas Zyrianov, Xiyue Zhu, and Shenlong Wang. 2022. Learning to Generate Realistic LiDAR Point Clouds. In *European Conference on Computer Vision*. Springer, 17–35.