IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 22 September 2026, accepted 11 October 2026, date of publication 19 October 2026, date of current version 2 November 2026.

Digital Object Identifier 10.1109/ACCESS.2026.3745292

RESEARCH ARTICLE

Generating 2D Bounding Box Annotations From Convex Collision Hulls in Unreal Engine 5

MARTIN DE FRIES[1,2], DANYEL RIDVANOĞULLARI[3], MARTIN MEINERS[1,2], TONI VISCIDO[1], KARIN THOMAS[2,4], (Member, IEEE), AND RENÉ DEGEN[2,5]

[1]Institute of Automotive Engineering, Faculty of Automotive Systems and Production, Cologne University of Applied Sciences, 50679 Cologne, Germany
[2]Division of Electricity, Department of Electrical Engineering, Uppsala University, 752 37 Uppsala, Sweden
[3]TÜV Rheinland AG, 51105 Cologne, Germany
[4]Department of Engineering Sciences, Mälardalen University, 721 23 Västerås, Sweden
[5]Rhineland Mechatronics Academy–Gesellschaft für intelligente Anwendung in Forschung und Entwicklung mbH, 53844 Troisdorf, Germany

Corresponding author: Martin de Fries (martin.defries@th-koeln.de)

This work was supported in part by European Union through European Regional Development Fund (ERDF); and in part by the Ministry of Economic Affairs, Industry, Climate Action and Energy of the State of North Rhine-Westphalia, Germany, through the Project "MetaMover" under Grant EFRE-20800950.

ABSTRACT This article presents an alternative method for generating bounding box annotations for synthetic image data using Unreal Engine 5. The proposed approach combines image capture, object identification through instance segmentation, and bounding box generation by projecting the vertices of the convex collision hulls of virtual objects onto the image plane. To assess this method, the proposed workflow was compared with established methods based on the projection of the oriented bounding boxes or the full mesh of an object. An evaluation of the annotation quality was performed in a single virtual road-intersection scenario using objects with different geometric complexities, considering different camera perspectives and object sizes. Downstream detector performance was not evaluated. The results show that the projection of convex collision hulls yields bounding boxes that more closely match the visible object extent than the projection of oriented bounding boxes, whereas it shows an annotation quality that is very similar on average to the full-mesh-based approach. The study further indicates that the collision-hull-based method maintains higher annotation quality than the approach using oriented bounding boxes across the investigated object sizes and viewing perspectives. An illustrative parameter-sensitivity analysis further indicates that the annotation quality depends on the convex-decomposition settings, whereas the investigated camera settings have only a limited influence. Furthermore, it has lower computational costs than a full-mesh-based approach. These findings suggest that convex collision hull projection provides a practical and computationally efficient compromise between coarse geometric approximation by oriented bounding box projection and detailed full mesh projection for automated image annotation in object detection.

INDEX TERMS Bounding box annotation, computer vision, convex collision hull, game engine, object detection, synthetic image data.

I. INTRODUCTION

Particularly in the development of autonomous or highly automated vehicles, virtual environments provide an important part of the data required for the development and testing processes. As an industrial example of the passenger car segment, Waymo LLC covers a large share of its test miles through simulation [1]. In the commercial vehicle segment, companies such as LEMKEN and Volvo have developed their vehicle functions using virtual environments [2], [3]. Beyond these industrial applications, empirical studies have further emphasized the importance of simulation-based testing in virtual environments. In a qualitative interview study involving 13 practitioners from seven companies, all

The associate editor coordinating the review of this manuscript and approving it for publication was Franco Cicirelli.

© 2026 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

participants identified simulation tools as components of their development and testing processes for autonomous driving systems [4]. In another study, 87 of 100 surveyed developers reported the use of simulation platforms and virtual environments in testing autonomous driving systems [5]. These findings indicate that virtual testing has become a widely established element of industrial development and testing workflows for autonomous and highly automated vehicles. A major advantage of virtual test drives is, on the one hand, their cost efficiency and, on the other hand, the high level of safety in acquiring data for critical situations, such as near-crash scenarios. Furthermore, virtual scenarios are deterministic and flexible in configuration. Consequently, they are independent of environmental influences and are fully reproducible if required. If a virtual environment is characterized by a high level of detail, it is suitable for generating synthetic data for the development of environmental perception systems. An example is the rendering and annotation of image data for the training and validation of object detectors, such as the widely used You Only Look Once (YOLO) detector [6]. In this context, bounding box annotation can be performed based on information from the virtual environment to automate a labor-intensive manual annotation process. Even in the case of automated annotation, a high accuracy must be achieved. Accurate object detection requires training data with precise bounding boxes. Otherwise, localization may become imprecise [7]. For instance, discrepancies in annotation may be reflected in a lower Intersection over Union (IoU) [8].

The objective of this study is to present an alternative approach for generating synthetic image data for training YOLO object detectors using Unreal Engine 5, which is part of the latest game engine generation. Particular attention is paid to the generation of bounding boxes through the projection of the 3D convex collision hulls (CCH) of virtual objects. Finally, the developed approach was evaluated by comparing it with other annotation principles used in the literature in terms of annotation quality with respect to the visible object extent and computational costs.

II. RELATED WORK

According to [9], synthetic image data can generally be generated using crop-out-based methods, graphics-API-based methods such as OpenGL, 3D-modeling-based methods, and, as in the present case, methods based on game engines. In particular, the up-to-date nature of the simulation software, identification of objects and classes within the field of view (FOV) of the camera, and generation of 2D bounding boxes to annotate the images were analyzed. In the following sections, relevant research articles from 2017 to 2026 from IEEE Xplore, ScienceDirect, Scopus, ACM Digital Library, and Springer are reviewed. The central search terms used were computer vision, YOLO, object detection, simulation, game engine, and synthetic data. Studies with insufficient description of the annotation process were not considered.

A. GAME ENGINES AS SIMULATION TOOLS

For an object detector to perform well in a real environment, the synthetic image data used for training should be as realistic as possible. The high level of detail of the virtual environment required for realistic rendering is primarily achieved through the selection of suitable simulation software. Some of the examined approaches use Epic Games' Unreal Engine 4 [10], [11], which has since been superseded by Unreal Engine 5. Simulation software based on Unreal Engine is also frequently used in the reviewed articles. For instance, the CARLA Simulator [12] is widely used in industry and science [5], [13], [14]. Furthermore, Microsoft's AirSim drone simulator was used to generate annotated image data [15], [16]. Another tool based on Unreal Engine 4 is the NVIDIA Deep Learning Dataset Synthesizer [17], whereas [18] implemented its PCGOD data-generation framework in Unreal Engine 5. Additional studies on the generation of labeled image data have applied the Unity game engine for simulations [19], [20].

B. OBJECT IDENTIFICATION

One task within the annotation process is to identify the objects present in the image and assign them to classes. The approaches in [10], [13], [14], [15], and [19] used semantic segmentation or instance segmentation for this purpose. Using these methods, segmentation maps are generated for each RGB image. In this map, the corresponding pixels or pixel regions in the image were assigned to various objects within the FOV, as described in [13] and [19]. Thus, through pixel-wise analysis, every object to which at least one pixel can be assigned was detected in the image. The disadvantage of this solution is that objects that are completely occluded by transparent objects cannot be detected without additional measures. Another approach adopted by [17] is to check for the possible overlapping of objects within the simulation with the frustum of the virtual camera. The frustum is a pyramid-shaped virtual body that represents the FOV of a camera. If the 3D axis-aligned bounding box of an object intersects the frustum, the object is considered visible in the image. The disadvantage of frustum inspection as an identification principle is that completely occluded objects can be declared visible. Therefore, additional visibility checks are required.

C. BOUNDING BOX GENERATION

To complete the annotation, 2D bounding boxes in image coordinates must be determined for the identified objects. To generate bounding boxes, [10], [13], [15], [19] rely on segmentation maps. In this case, the pixels of an object region with minimum and maximum values for the position in the X- and Y-directions of the image define the side boundaries of the bounding box. Owing to the pixel-precise definition of the side boundaries, the bounding box tightly encloses the visible object extent. However, because only visible pixels

can be used to generate a bounding box, the occluded object regions are not enclosed by the bounding box.

Another approach, used in [14] and [16], is the projection of the corner points of a 3D oriented bounding box (OBB) onto the image plane. This resulted in eight points, where the image points with the minimum and maximum values for the position in the X- and Y-directions defined the side boundaries of the bounding box. However, the geometry of an object and its OBB sometimes differ considerably from one another, which is why the 2D bounding box obtained by projection does not tightly enclose the object in the image. For this reason, [18] used a number of markers placed at exposed positions using an algorithm that produces a tighter bounding box when projected. Although this study focuses on game-engine-based approaches, the Blender-based 3D modeling pipeline presented in [21] is also considered as relevant. Its projection-based bounding box generation uses all vertices of the three-dimensional object mesh. In this study, this approach was referred to as the full-mesh-based method (FM). According to [21], this method yields substantially tighter annotations compared to an OBB projection in Blender, although at a higher computational cost. Unlike bounding box generation by segmentation, all of the three projective methods make it possible to generate a bounding box that includes occluded object regions.

D. CONCLUSION ON RELATED WORK

The literature review shows that current game engines and simulation environments built upon them provide a foundation for generating realistic synthetic image data. Against this background, the Unreal Engine 5 was selected as the simulation environment. Owing to its modernity and graphical capabilities, it is suitable for meeting the requirements of realistic synthetic image data generation. In addition, it offers diverse and high-performance extension capabilities using C++ for the development of an annotation software. Furthermore, this study uses segmentation to identify the objects present in the image. For bounding box generation, a method based on the projection of the vertices of the CCH of the depicted objects is presented. This method is intended to represent an alternative to the projection of algorithmically defined markers [18]. Furthermore, the CCH projection approach is intended to provide more precise annotations than the OBB projection and to require substantially lower computational costs than the FM-based projection approach. For comparison and evaluation, methods for generating bounding boxes through the projection of the OBB, FM projection and instance segmentation were implemented.

III. METHODOLOGY

A. VIRTUAL ENVIRONMENT

The virtual environment was created using the Unreal Engine Editor, as shown in Fig. 1. The center of the virtual environment consists of a road intersection with infrastructure elements. In the immediate surroundings, trees, fields, road elements, and additional objects create realistic test scenarios. Distant landscapes, including mountains, limit the visible horizon. Distant areas were designed with a low level of detail to ensure a computationally efficient simulation.

FIGURE 1. Virtual environment used for the evaluation, with detailed features of a road intersection in the foreground and a less detailed landscape in the background limiting the horizon.

B. DATA GENERATOR IMPLEMENTATION

A C++ class of type actor, for which Unreal Engine provides a template, was used to implement the routines for generating the annotated image data. The methods contained in this actor enable the capture of image data, identification and classification of visible objects, and their annotation by means of bounding boxes according to different approaches. A systematic overview of the set of methods is presented in Fig. 2 and is described below. Part of this actor template is the *BeginPlay(…)* method, which is executed at the start of the simulation and is commonly used to initialize subsequent operations. Subsequently, the *Tick(…)* method was executed in each simulation frame and used to implement iterative operations.

C. IMAGE GENERATION

The first step of the presented data-generation process, shown in Fig. 2, is to render RGB images that will be annotated later. For this purpose, a scene capture component of the class *USceneCaptureComponent2D* with a defined FOV was created within the *BeginPlay(…)* method of the actor at the start of the simulation. In addition, *BeginPlay(…)* initializes a render target of type *UTextureRenderTarget2D* with defined resolution ($\mathrm{Res_X}$, $\mathrm{Res_Y}$). During the simulation, the operations were executed for each frame in *Tick(…)* if the user triggered them via the keyboard. In this process, an image of the scene is generated by the scene capture component, temporarily stored in the render target, and subsequently saved in a PNG file.

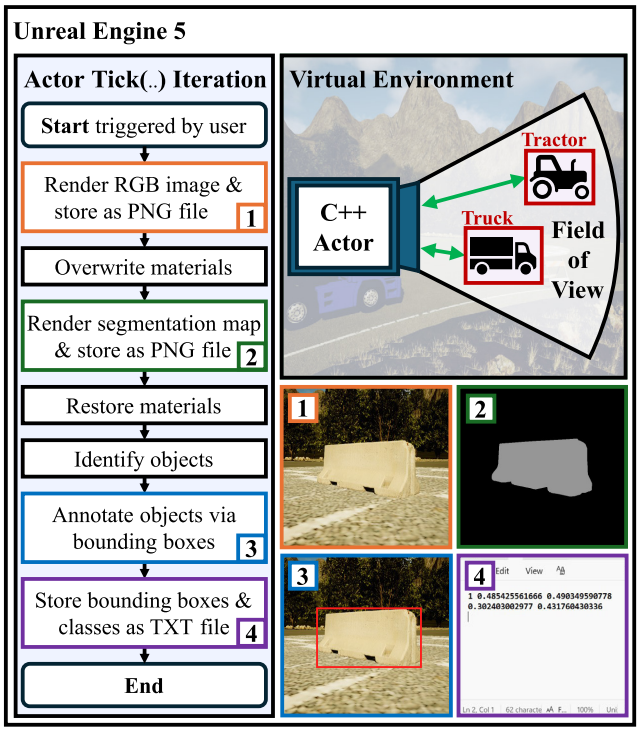

FIGURE 2. Simplified overview visualizing the essential steps of the data-generation process implemented in the *Tick(…)* method of a C++ actor in Unreal Engine illustrated by images showing the results of the steps.

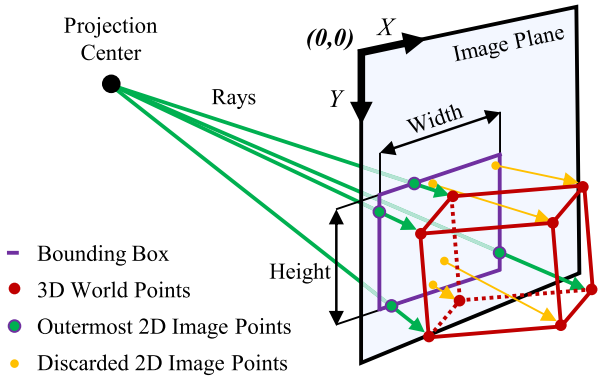

FIGURE 3. Schematic representation of the definition of a bounding box by projecting the corner points of a Cube onto a two-dimensional image plane.

D. OBJECT IDENTIFICATION

To access objects visible in the image for subsequent annotation, they must first be identified. For this purpose, the principle of instance segmentation is implemented in which each pixel is assigned to an object. As in the image generation step, a scene capture component and render target were created at the start of the simulation. In this case, elements irrelevant to segmentation, such as clouds or lighting effects, were excluded from the rendering using configuration flags.

Object identification is performed for each frame in *Tick(…)* when the user triggers the image capture via the keyboard. First, a dedicated method stores the materials of all relevant objects listed for segmentation and overwrites them with a dedicated segmentation material. This material emits 8-bit grayscale values by setting the emissive color of the object pixels according to the predefined custom depth stencil values. Only objects with a stencil value greater than zero were included in the segmentation rendering, whereas objects without an assigned stencil value were excluded from this rendering pass. Consequently, occlusions within the segmentation map are caused only by other objects participating in the segmentation process. An example of a segmentation map is shown as part of Fig. 2. The segmented image is then captured by the scene capture component, rendered in the corresponding render target, and saved as a PNG file. Thereafter, another method restores the previously stored original materials of all elements in the virtual environment. A similar approach, in which a segmented representation is achieved by overwriting the materials, was used in [22].

To identify all visible objects, the segmentation map was examined pixel-by-pixel, and all objects were registered based on their stencil values. Thus, by assigning classes to the stencil values using a lookup table before starting the simulation, objects can be classified based on their values during the simulation. This principle enables the user to determine which objects in an image are to be identified and annotated individually.

E. BOUNDING BOX ANNOTATION

In the next step, all identified objects in the captured images were annotated using 2D bounding boxes. This operation is performed for each frame in *Tick(…)* when the data capture is triggered by the keyboard. For all objects, the program defines a set of points assigned to an object in the image plane of the render target according to four different approaches.

The first approach, abbreviated as SEG, uses a previously defined segmentation map that represents the object points as regions in the image plane. In this case, all pixels belonging to an object are listed from the segmentation map together with their positions in the image plane based on the stencil value.

The second approach involves projecting the eight corner points of the OBB, as illustrated in Fig. 3. For this purpose, the corner points of the OBB of an object-related static mesh component as well as their positions in world coordinates, were determined. Subsequently, the *ProjectWorldToRenderTarget(…)* method determines the positions of the projected corner points in image coordinates, and lists them according to Algorithm 1. This method is based on the implementation of the projection process described in the CARLA Simulator documentation [23], which was scientifically applied in [14].

Algorithm 1 Routine to Project 3D Points on the Render Target Realized Within Method ProjectWorldToRender Target(…)

Require: 3D world point, scene capture, image size
Ensure: 2D pixel position in the render target
Validate capture component and image size
Read camera pose
Build view matrix from inverse camera transform
Build projection matrix from FOV and aspect ratio
Transform world point into camera space
Project point to screen space
return 2D pixel position

Algorithm 2 Routine to Extract All Vertices of a Convex Hull Realized Within Method GetCollisionHullFromActor(…)

Require: actor
Ensure: List of hull vertices in world coordinates
if actor is null **then**
  return empty list
end if
for each static mesh component in actor **do**
  Get static mesh from component
  Get body setup from mesh
  Get component world transform
  for each convex collision hull **do**
    for each local hull vertex **do**
      Transform local vertex to world coordinates
      Store transformed vertex
    end for
  end for
end for
return all stored world vertices

The third approach proposed in this study uses the CCH of an object for geometric approximation. The geometry of this hull approximates the shape of the object, and is used in gaming simulations to represent collisions. In this context, the *GetCollisionHullFromActor(…)* routine shown in Algorithm 2 extracts the vertices of an actor's CCH and returns them in world coordinates.

The function retrieves all static mesh components of class *UStaticMeshComponent* associated with the input object and obtains the referenced *UStaticMesh* and its corresponding *UBodySetup*. *UBodySetup* contains the aggregate collision geometry, whose *ConvexElems* array stores convex collision elements. Thus, the implementation accesses the Unreal Engine's internal, version-specific data structures for representing collision geometry, as documented in the API reference for *UBodySetup* and *FKConvexElem* [24], [25]. The general methodological principle of extracting exact object and scene information from Unreal Engine was also employed by UnrealGT for the automatic generation of ground-truth annotations [10]. However, UnrealGT serves only as a methodological reference and does not necessarily rely on the same engine classes or data structures used in the present implementation.

Through extraction, the vertices of the convex collision elements are available in the local coordinate system of the meshes. These local points are transformed into world coordinates and stored in an array. For the projection of vertices onto the image plane and their listing, *ProjectWorldToRenderTarget(…)* was used once again.

The fourth approach is based on the method projecting the full mesh described in [21], in which all vertices of the three-dimensional polygonal object mesh are used for projection-based bounding box generation. For this purpose, the mesh vertices of all static mesh components associated with an actor were extracted, and their local positions were read from the selected vertex buffer, transformed into world coordinates, and stored in an array. Subsequently, the *ProjectWorldToRenderTarget(…)* method projects all these world-space vertices into image coordinates.

As shown in Fig. 4, all four approaches provide different sets of object points. The segmentation map provides one point for every object-related pixel to define the bounding box boundaries, which is why visualization of the individual points is omitted. The CCH projection provides fewer points than the FM projection but still more than the eight points obtained from the OBB projection. Compared with the OBB- and CCH-based approaches, the FM-based method yields a denser geometric representation with more points.

Subsequently, from the determined points, those with the lowest X-, highest X-, lowest Y-, and highest Y-coordinates in the image coordinate system were identified. Based on these four extreme points, and by incorporating the known image resolution in the X- and Y-directions, the coordinates of the bounding box center point, as well as the width and height of the bounding box for the object in the image, can be determined in a format suitable for YOLO detectors [6].

F. EVALUATION SETUP

For the systematic evaluation of the presented approaches, the road intersection scenario described above was used as a common baseline environment for the experiments and augmented with different objects. The environment was kept fixed so that all four methods are applied to the same rendered situations; since the quantitative experiments contain no occlusion, the surrounding scene affects neither the projected geometry nor the segmentation-derived reference, and such situations are examined qualitatively in Section IV-E. The evaluation objects were selected such that a broad spectrum of geometric complexity levels was covered, as illustrated in Fig. 5. A Cube represents an idealized and symmetrical standard geometry. The road Barrier and Deer represent more regular geometries, whereas the Stanford Bunny [26] and Roller provide more complex shapes. The Tree additionally represents a filigree geometry with particularly high

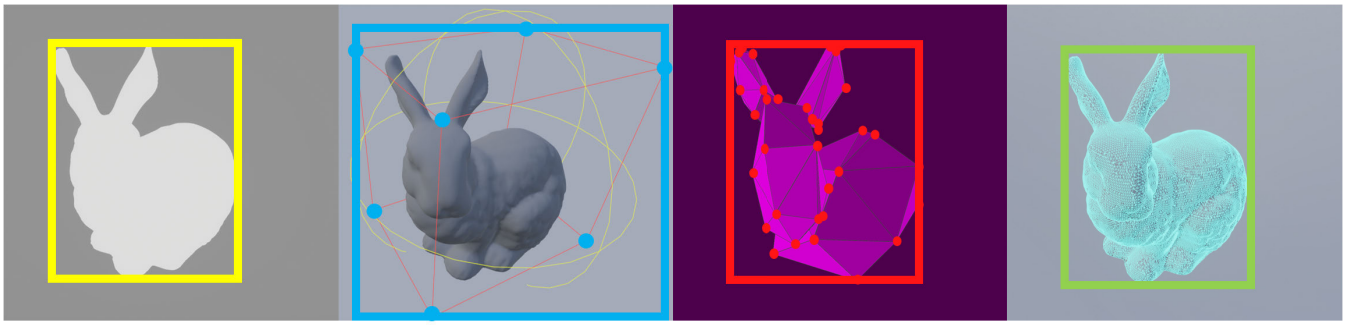

FIGURE 4. Visualization of the Stanford Bunny's bounding boxes derived from points of the (a) segmentation map, (b) oriented bounding box, (c) convex collision hull, and (d) full mesh with occluded projected points omitted from the visualization for clarity.

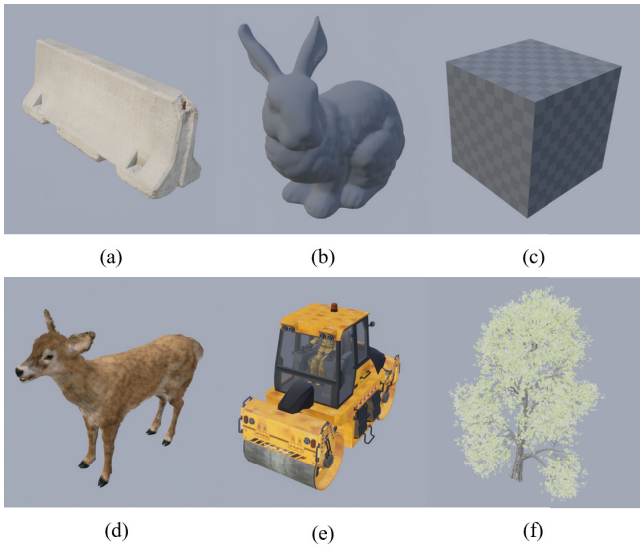

FIGURE 5. Objects with varying geometric complexity for evaluation: (a) Road Barrier, (b) Stanford Bunny, (c) Cube, (d) Deer, (e) Roller, and (f) Tree.

geometric complexity. Thus, the extent to which the precision of automatically derived 2D bounding boxes is influenced by the shape complexity of the underlying 3D object can be investigated. The six objects were selected deliberately to span a defined range of geometric complexity, and were therefore treated as fixed evaluation conditions rather than as a random sample of objects. The results are consequently reported per object, and pooled values were read as averages over this specific set of objects.

The geometric quality of the bounding boxes generated from the CCH, FM, and OBB projections was evaluated with respect to segmentation-derived bounding boxes. Since this reference is obtained from the visible object pixels in the segmentation map, it represents the visible object extent in the image rather than the complete projected geometric extent of the corresponding 3D object. In addition to the geometric comparison, the SEG, CCH, FM, and OBB were considered in the evaluation of computational costs.

TABLE 1. Parameters used for collision hull generation and image capture in the Unreal Engine throughout the experiments on annotation quality by perspective and object size as well as on the computational costs.

Parameter	Value
Hull count	64
Hull precision	1 000 000
Maximum hull vertices	32
FOV (°)	70
Res_X (pixels)	1000
Res_Y (pixels)	1000

For all objects used in the main experiments, the CCHs of their static meshes were generated using the *Auto Convex Collision* routine in the Static Mesh Editor of Unreal Engine [27]. The values in Table 1 for the "Hull Count", "Hull Precision", and "Maximum Hull Vertices" were used when generating the CCH for the experiments on annotation quality by perspective, object size, and computational costs. Hull Count defines the maximum number of convex hulls used to approximate the object geometry. The Maximum Hull Vertices define the maximum number of vertices of each generated convex hull, and Hull Precision controls the voxel resolution used for the convex-decomposition. Note that the parameters Hull Count and Maximum Hull Vertices represent the upper bound of the values that can be set in the *Auto Convex Collision* routine. These maximum values are not necessarily achieved for every object, which is why these figures cannot be used to infer the values listed in Table 2.

The image captures for the main experiments were created using a *USceneCaptureComponent2D*, with the FOV from Table 1 used for all captures. A *UTextureRenderTarget2D* with constant resolution Res_X and Res_Y served as the output target. The consistency of these camera parameters during the experiments on annotation quality by perspective, object size, and on the computational cost ensures that the generated 2D bounding boxes can be compared under identical projective conditions. Unless otherwise specified, the camera setting consisted of a resolution of 1000 × 1000 pixels and an FOV of 70°.

TABLE 2. Geometric characteristics of the six evaluation objects and the corresponding numbers of projected vertices for the convex collision hull (CCH), full mesh (FM), and oriented bounding box (OBB) projection.

		Projected vertices		
Object	Description	CCH	FM	OBB
Barrier	Regular	87	2546	8
Bunny	Complex	167	30 488	8
Cube	Symmetric	9	54	8
Deer	Regular	173	6294	8
Roller	Complex	318	21 880	8
Tree	Filigree	504	69 491	8

The geometric characteristics of the six evaluation objects and corresponding numbers of vertices considered by the three projection-based approaches are summarized in Table 2. The vertex counts illustrate the substantially different geometric complexity of the objects, as well as the reduction in the projected geometry achieved by CCH compared with FM.

To evaluate the influence of viewing direction, all six objects were examined from 16 predefined perspectives. Perspectives 1–8 covered azimuth angles from 0° to 315° in increments of 45° at an elevation of 0°, whereas perspectives 9–16 repeated the same azimuth angles at an elevation of 45°. The camera radius was 300 cm for the road Barrier, Stanford Bunny, Cube, Deer, and Roller and 400 cm for the Tree.

The influence of the object size was investigated using 2000 randomized situations for each of the six evaluation objects, resulting in a total of 12 000 samples. For each image acquisition, a random camera position was sampled on the upper half of a spherical shell around the target object. The radial distance is drawn within predefined minimum and maximum limits, which are selected individually for each object, depending on its dimensions in the virtual environment. The viewing direction was randomly sampled and transformed according to a configurable local offset and rotation relative to the target object. Samples outside the user-defined spatial bounds are rejected, and the accepted camera pose is finally oriented toward a predefined look-at point on the target object. According to [28], samples were assigned to the size classes

- *Small*: visible object region $< 32^2$ px^2,
- *Medium*: 32^2 px$^2 \leq$ visible object region $\leq 96^2$ px^2, and
- *Large*: visible object region $> 96^2$ px^2

based on the visible object region obtained from the segmentation map. Randomization ranges were chosen such that all three size classes would be sufficiently populated. Furthermore, the same samples were used for the paired comparison of the projection-based annotation approaches.

To further investigate the sensitivity of the CCH-based method to convex-decomposition parameters, a separate experiment was conducted using the Stanford Bunny. Two viewpoints from the perspective experiment were selected, with Perspective 11 representing a challenging case and Perspective 14 an easy case. The Hull Count, Hull Precision, and Maximum Hull Vertices were individually varied according to a one-factor-at-a-time procedure. The influence of the Hull Count was evaluated at 1, 4, 16, and 64, Hull Precisions at 10 000, 100 000, and 1 000 000, and Maximum Hull Vertices at 8, 16, and 32. When one parameter was varied, the remaining two parameters were maintained at their highest values.

The sensitivity to camera settings was evaluated for the Stanford Bunny using three combinations of image resolution and field of view: 1000 × 1000 px with an FOV of 70°, 1000 × 1000 px with an FOV of 90°, and 960 × 720 px with an FOV of 90°. The resulting annotation quality was evaluated for small, medium, and large object size classes, with a separate set of samples for each camera setting.

The computational cost evaluation was performed on a system equipped with an AMD Ryzen 9 7950X3D processor, an NVIDIA GeForce RTX 4090 GPU, and 128 GB of RAM under Windows 11 using Unreal Engine 5.6. Execution times were measured on the game thread using *FPlatformTime::Seconds()*. For CCH, FM, and OBB, the measured interval comprised point retrieval, required coordinate transformations and projection, and determination of the outermost projected points. For SEG, the measurement included the GPU-to-CPU pixel read-back operation and determination of the outermost segmentation points. Scene capture and the final storage of the resulting bounding box were excluded from the respective timing intervals. For the measurements, the first 250 recorded data points were discarded to exclude warm-up effects, as a qualitative inspection indicated that execution times stabilized after approximately 250 frames. All applications except Unreal Engine were closed during the measurements. Furthermore, the methods were always executed in the same order (SEG → CCH → FM → OBB), and potential order-related effects were neglected.

In addition to quantitative experiments, selected practically relevant edge cases were qualitatively investigated using the Stanford Bunny. These included partial and severe occlusions, image-boundary truncation, transparent materials, overlapping objects, actors consisting of multiple static-mesh components, and near-clipping situations.

For quantitative evaluation, two complementary metrics were used. The Intersection over Union (IoU) serves as a measure of the spatial overlap between the projected and referenced bounding boxes. The reference values A_{seg}, x_{seg}, and y_{seg} are provided by the bounding box generated from the segmentation map, whereas the bounding boxes from projection-based approaches provide the values for A_{proj}, x_{proj}, and y_{proj}. Values close to one indicate a high degree of agreement between the two boxes. It is defined as

$$\text{IoU} = \frac{A_{\text{intersection}}}{A_{\text{union}}} = \frac{A_{\text{proj}} \cap A_{\text{seg}}}{A_{\text{proj}} \cup A_{\text{seg}}}. \tag{1}$$

In addition, the Center Offset (CO) was determined to capture the positional deviation between the centers of the

TABLE 3. Camera perspectives used in the evaluation with elevation and azimuth given relative to the object coordinate system.

Elevation 0°								
Perspective	1	2	3	4	5	6	7	8
Azimuth (°)	0	45	90	135	180	225	270	315
Elevation 45°								
Perspective	9	10	11	12	13	14	15	16
Azimuth (°)	0	45	90	135	180	225	270	315

two bounding boxes. Smaller values indicate a more precise positional agreement. Because the center coordinates are given in a normalized form, they are scaled by the image resolution, such that the distance can be specified in pixels. The CO was computed as

$$\mathrm{CO} = \sqrt{(\Delta x\,\mathrm{Res_X})^2 + (\Delta y\,\mathrm{Res_Y})^2}$$
$$\text{with} \quad \Delta x = x_{\mathrm{proj}} - x_{\mathrm{seg}}, \quad \Delta y = y_{\mathrm{proj}} - y_{\mathrm{seg}}, \qquad (2)$$

where Δx and Δy denote the normalized horizontal and vertical center differences, respectively, and $\mathrm{Res_X}$ and $\mathrm{Res_Y}$ denote the horizontal and vertical image sizes, respectively. Furthermore, CO is an absolute pixel distance and is therefore not normalized by the object size. The absolute values are expected to increase with the size of the object in the image.

For statistical analysis of annotation quality, paired two-sided t-tests were used to compare CCH with FM and OBB within the respective object-size classes. The paired design was justified because all the methods were evaluated in the same rendered situations. The twelve annotation-quality comparisons, comprising three object-size classes and four contrasts each, define one family. Separately, the two computational cost comparisons between CCH and other projection-based approaches define another family. Within each family, the resulting p-values were adjusted using the Holm–Bonferroni correction. The reported 95% confidence intervals are nominal, unadjusted intervals for the respective mean difference, and are intended as effect-size estimates rather than as simultaneous intervals.

By combining these metrics and statistical analyses, the quality of the automatically generated 2D bounding boxes is assessed not only with respect to their overlap with the visibility-based reference, but also with respect to their positional agreement with it. This enables a differentiated evaluation of the three projection-based methods with respect to object geometry, perspective, object size, and camera setting, as well as their computational costs.

IV. RESULTS AND DISCUSSION

To compare the precision of the projection-based annotation approaches visualized in Fig. 6, an evaluation was conducted under the simulation conditions described in the previous section using the metrics defined therein. The first experiment focused on the behavior as a function of perspective, whereas

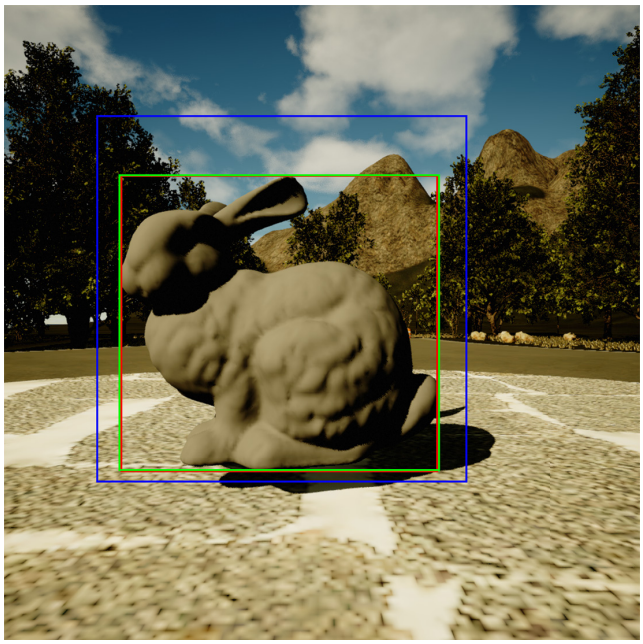

FIGURE 6. Image of the Stanford Bunny from the side perspective, showing bounding boxes obtained from the projection of the oriented bounding box (blue), the projection of the convex collision hull (red), the projection of the full mesh (green) and the segmentation-map-derived reference (yellow, visually covered by the red and green boxes).

the second experiment investigated the relationship between the evaluation metric and size of objects in the image. In addition to annotation quality, the annotation approaches were compared with respect to their computational costs. Finally, the behavior of the CCH-based method was investigated with regard to variations in hull-generation parameters and under selected practical edge cases. Training an object detector for evaluation purposes was omitted because the present work addressed the quality of the automatically generated 2D bounding boxes themselves rather than the performance of a downstream learning method.

A. ANNOTATION QUALITY BY PERSPECTIVE

In the perspective experiment, the IoU and CO metrics were determined for Barrier, Stanford Bunny, Cube, Deer, Roller, and Tree from 16 predefined viewpoints according to the equations 1 and 2. The bounding boxes generated from the CCH, FM, and OBB projections were compared to the segmentation-derived reference. The investigated camera perspectives are listed in Table 3. Perspectives 1–8 represent azimuth angles from 0° to 315° in increments of 45° at an elevation of 0°, whereas Perspectives 9–16 repeat the same azimuth angles at an elevation of 45°.

The IoU results for all the six objects are shown in Fig. 7. The influence of this perspective strongly depends on the underlying object geometry. For the Cube, all three projection-based approaches produced nearly identical results across the investigated perspectives, with IoU values between approximately 0.9857 and 0.9900. For the remaining objects, the CCH-based bounding boxes consistently showed

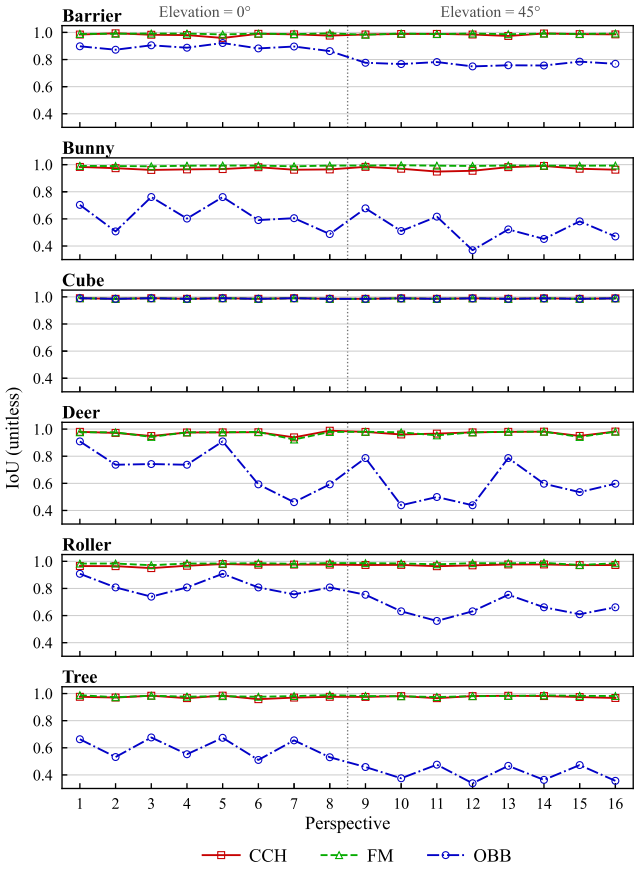

FIGURE 7. Intersection over Union (IoU) for all six objects across all 16 described perspectives showing the trends of the CCH-based (□, red), FM-based (△, green), and OBB-based (○, blue) bounding box generation approaches.

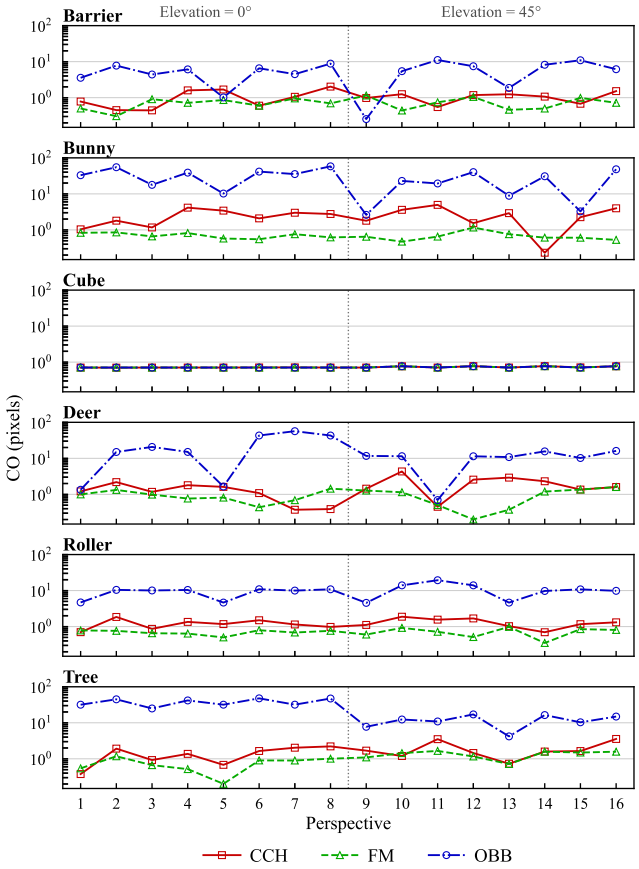

FIGURE 8. Center offset (CO) for all six objects across all 16 described perspectives with logarithmic scale showing the trends of the CCH-based (□, red), FM-based (△, green), and OBB-based (○, blue) bounding box generation approaches.

a higher agreement with the reference than those derived from the OBB. This difference was particularly pronounced for complex and asymmetrical geometries. For the Stanford Bunny, the IoU ranged from 0.9489 to 0.9904 for CCH and 0.9875 to 0.9955 for FM, whereas the OBB-based values ranged from 0.3690 to 0.7614. A similar behavior was observed for the object Tree, for which CCH and FM achieved IoU ranges of 0.9586 to 0.9845 and 0.9730 to 0.9885, respectively, compared with 0.3384 to 0.6763 for OBB.

The CO results in Fig. 8, shown on a logarithmic scale, complement these findings. For the Cube, the positional deviations remained below one pixel for all the methods and perspectives. For more complex objects, FM generally produced the lowest COs, whereas CCH remained substantially closer to the reference than OBB. For example, the maximum CO for the Stanford Bunny was 4.951 pixels for CCH and 1.163 pixels for FM, compared to 58.158 pixels for OBB. For the Tree, the corresponding maximum values were approximately 3.558, 1.655, and 47.707 pixels.

Overall, the experiment shows that the quality of projection-based 2D bounding boxes is affected by both object geometry and viewing perspective. Although OBB projection shows high agreement with the reference for the Cube, its agreement varies considerably for more complex objects. The CCH-based method provides a substantially more precise and stable approximation across the investigated viewpoints and, at the same time, remains considerably closer to the FM results than to the OBB results.

B. ANNOTATION QUALITY BY OBJECT SIZE

In the second experiment, the IoU and CO metrics were examined as a function of object size in the image for all six evaluation objects. Based on the visible object region, 12 000 randomized samples were assigned to the three size classes defined above. This resulted in 8204 small, 2794 medium, and 1002 large samples, respectively. The allocation differs by object: 1377/457/166 for the Barrier, 1086/669/245 for the Stanford Bunny, 1401/434/165 for the Cube, 1504/389/107 for the Deer, 1426/415/159 for the Roller, and 1410/430/160 for the Tree. The segmentation-derived bounding box is again used as a visibility-based reference.

As shown in Fig. 9, the CCH-based method achieved a substantially higher agreement with the reference than

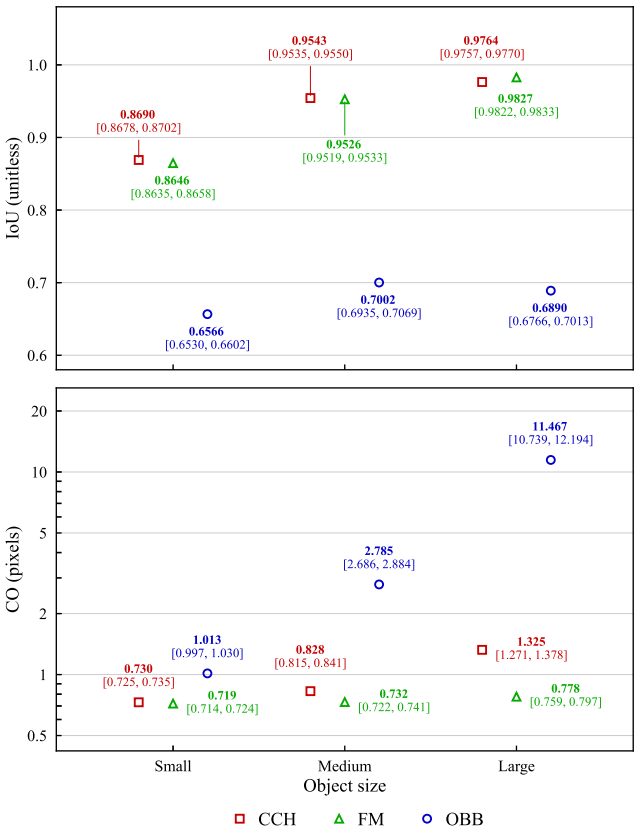

FIGURE 9. Mean Intersection over Union (IoU) and center offset (CO, logarithmic scale) of the bounding boxes generated by CCH (□, red), FM (△, green), and OBB (○, blue) projection for the small, medium, and large object-size classes with the 95% confidence intervals in brackets.

the OBB-based approach across all size classes. The mean IoU values of CCH were 0.8690, 0.9543, and 0.9764 for small, medium, and large objects, respectively, compared with 0.6566, 0.7002, and 0.6890 for OBB. The corresponding mean paired differences $\mathrm{IoU}_{\mathrm{CCH}} - \mathrm{IoU}_{\mathrm{OBB}}$ listed in Table 4 increase from 0.212 for small objects to 0.254 for medium objects and 0.287 for large objects. All differences were statistically significant after Holm–Bonferroni correction ($p_{\mathrm{adj}} < 0.001$).

Similar behavior was observed for the CO. The mean paired difference $\mathrm{CO}_{\mathrm{OBB}} - \mathrm{CO}_{\mathrm{CCH}}$ increased from 0.284 pixels for small objects to 1.957 pixels for medium objects and 10.142 pixels for large objects ($p_{\mathrm{adj}} < 0.001$ for all size classes). Thus, the advantage of CCH over OBB is more pronounced with increasing object size for both spatial overlap and positional agreement.

Compared with FM, the CCH-based method achieved very similar IoU values. The mean paired IoU differences were 0.00439, 0.00170, and −0.00638 for small, medium, and large objects, respectively, with all three differences reaching statistical significance ($p_{\mathrm{adj}} < 0.001$). FM showed slightly lower COs than CCH, with mean differences $\mathrm{CO}_{\mathrm{FM}} - \mathrm{CO}_{\mathrm{CCH}}$ of −0.0106, −0.0962, and −0.547 pixels for the three size classes.

TABLE 4. Paired comparisons on the pooled size data of all objects with mean differences in IoU (unitless) and CO (pixels) reported with the unadjusted 95% confidence intervals in brackets at the 1000 px by 1000 px setting with an FOV of 70°.

Comparison	Mean diff.	95% CI	t	p_{adj}
Small (df = 8203)				
IoU: CCH − OBB	0.212	[0.209, 0.216]	116.94	< 0.001
IoU: CCH − FM	0.00439	[0.00414, 0.00463]	35.71	< 0.001
CO: OBB − CCH	0.284	[0.267, 0.300]	33.74	< 0.001
CO: FM − CCH	−0.0106	[−0.0127, −0.0084]	−9.57	< 0.001
Medium (df = 2793)				
IoU: CCH − OBB	0.254	[0.247, 0.261]	74.49	< 0.001
IoU: CCH − FM	0.00170	[0.00129, 0.00210]	8.18	< 0.001
CO: OBB − CCH	1.957	[1.86, 2.05]	39.34	< 0.001
CO: FM − CCH	−0.0962	[−0.1071, −0.0853]	−17.29	< 0.001
Large (df = 1001)				
IoU: CCH − OBB	0.287	[0.275, 0.299]	47.10	< 0.001
IoU: CCH − FM	−0.00638	[−0.00706, −0.00571]	−18.58	< 0.001
CO: OBB − CCH	10.142	[9.44, 10.85]	28.20	< 0.001
CO: FM − CCH	−0.547	[−0.600, −0.494]	−20.23	< 0.001

TABLE 5. CCH-based annotation quality for the six evaluation objects across the three object-size classes at the 1000 px by 1000 px setting and an FOV of 70°. Cell colors are scaled independently within each column, ranging from white for the minimum value to green for the maximum IoU and red for the maximum CO.

	IoU (unitless)			CO (pixels)		
Object	Small	Medium	Large	Small	Medium	Large
Barrier	0.8776	0.9617	0.9848	0.741	0.766	0.893
Bunny	0.9133	0.9602	0.9704	0.784	0.899	1.640
Cube	0.8550	0.9515	0.9844	0.719	0.735	0.741
Deer	0.8699	0.9594	0.9715	0.664	0.744	1.836
Roller	0.8528	0.9479	0.9759	0.757	0.850	1.324
Tree	0.8557	0.9415	0.9721	0.729	0.932	1.550

The object-specific CCH results in Table 5 further show that the influence of object size varies with object geometry. While the IoU generally increased from the small to the large size class for all six objects, the CO showed larger object-dependent differences, particularly for large objects. The Cube exhibited comparatively stable COs across all size classes, whereas more complex objects, such as the Bunny, Deer, and Tree, showed larger positional deviations for the large size class.

For the Cube, the Auto Convex Collision routine returns a single box. Therefore, the convex collision hull and the oriented bounding box describe the same geometry and coincide by construction. The residual difference of approximately 1.5×10^{-5} IoU reflects only numerical effects of the decomposition. Therefore, the Cube contributed 2000 practically zero differences to the pooled comparison, attenuating the reported advantage of CCH over OBB. Excluding the samples with the cube, the mean paired IoU differences increase from 0.212, 0.254 and 0.287 to 0.256,

TABLE 6. Mean paired differences in IoU (unitless) per evaluation object and object-size class at the 1000 px by 1000 px setting with an FOV of 70°, where S, M, and L denote the small, medium, and large object-size class, respectively. Positive values indicate an advantage of the CCH-based approach. All differences between CCH and OBB are statistically significant with $p < 10^{-7}$, whereas differences marked with † are not statistically significant at the 5% level. For the Cube, the *Auto Convex Collision* routine returns a single box, so CCH, FM, and OBB approximate the same geometry and the residual differences reflect numerical effects only.

	Samples n			IoU: CCH − OBB			IoU: CCH − FM		
Object	S	M	L	S	M	L	S	M	L
Barrier	1377	457	166	0.09745	0.12973	0.16472	0.00684	0.00509	−0.00111†
Bunny	1086	669	245	0.34418	0.36934	0.40378	0.00619	−0.00066†	−0.01652
Cube	1401	434	165	0.00001	0.00001	0.00002	0.00001	0.00001	0.00002
Deer	1504	389	107	0.28360	0.31696	0.33281	0.01041	0.00876	0.00020†
Roller	1426	415	159	0.17453	0.21091	0.26956	−0.00501	−0.00736	−0.01125
Tree	1410	430	160	0.39647	0.44820	0.52021	0.00799	0.00580	−0.00250

TABLE 7. Median execution times per object at the 1000 px by 1000 px setting with an FOV of 70°, based on 2000 situations per object.

Time (ms)	SEG	CCH	FM	OBB
t_{Barrier}	9.1399	0.0313	0.4531	0.0036
t_{Bunny}	9.0246	0.0480	5.5248	0.0039
t_{Cube}	10.8206	0.0121	0.0130	0.0035
t_{Deer}	11.3510	0.0482	1.1515	0.0036
t_{Roller}	8.9817	0.0766	3.8957	0.0036
t_{Tree}	8.8039	0.1167	12.3277	0.0042

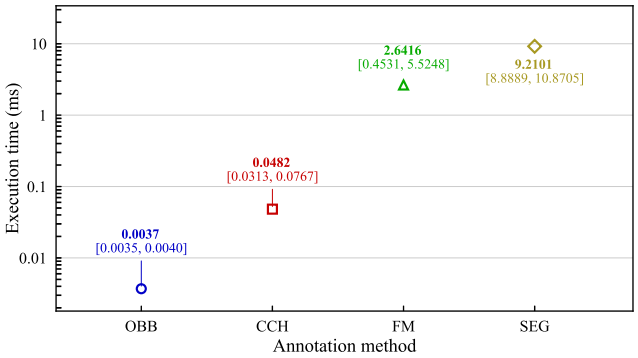

FIGURE 10. Median execution time of the bounding box generation based on OBB projection (○, blue), CCH projection (□, red), FM projection (△, green) and segmentation map (◇, yellow) shown on a logarithmic scale with the first and third quartiles in brackets.

0.301 and 0.344 for the small, medium and large size classes, respectively. Hence, the pooled values reported here are conservative.

Because the six objects constituted fixed evaluation conditions, the comparisons were additionally evaluated per object, as summarized in Table 6. The CCH-based method achieved a higher mean IoU than the OBB-based approach for all eighteen object-by-class cells, with $p < 10^{-7}$ for every cell. The differences with respect to FM are an order of magnitude smaller and change sign between objects, ranging from −0.0165 for the Stanford Bunny in the large size class to +0.0104 for the Deer in the small size class. In three of the eighteen cells the difference was not statistically significant. Overall, CCH provides a more precise approximation than OBB across all evaluated geometries and object sizes, whereas neither CCH nor FM is systematically superior to the other with respect to the visibility-based reference.

C. COMPUTATIONAL COSTS

In the third experiment, the computational costs of the four annotation approaches were evaluated using the execution times obtained for six test objects. Table 7 summarizes the median execution time for each object. The pooled results in Fig. 10 additionally provide the median and first and third quartiles across all 12 000 evaluated situations.

As summarized in Fig. 10, the median execution time was 0.0482 ms for CCH, compared with 2.6416 ms for FM, and 0.0037 ms for OBB across all objects. The SEG-based approach required the highest median execution time of 9.2101 ms. Accordingly, the CCH-based method was approximately 55 times faster than FM and 191 times faster than SEG. The OBB-based approach was approximately 13 times faster than CCH for the chosen object selection. The object-specific results in Table 7 indicate that the execution times of CCH and FM increase monotonically with the number of projected vertices in Table 2, whereas the OBB and SEG times are largely independent of the object geometry.

The differences between the projection-based methods were also assessed using paired two-sided statistical t-tests. The execution time for CCH was significantly lower than that for FM, with a mean paired FM-minus-CCH time difference of 3.8588 ms (95% CI [3.7830, 3.9346], $t(11\,999) = 99.79$, $p_{\text{adj}} < 0.001$). In contrast, CCH required significantly more time than OBB, with an OBB-minus-CCH time difference of −0.0537 ms (95% CI [−0.0543, −0.0530], $t(11\,999) = -167.15$, $p_{\text{adj}} < 0.001$).

Overall, the results quantified the intended computational trade-off of the CCH-based approach. It requires more computation than the projection of the eight OBB corners but substantially reduces the computational cost compared with the full mesh projection, while retaining a considerably more detailed geometric representation. The comparatively

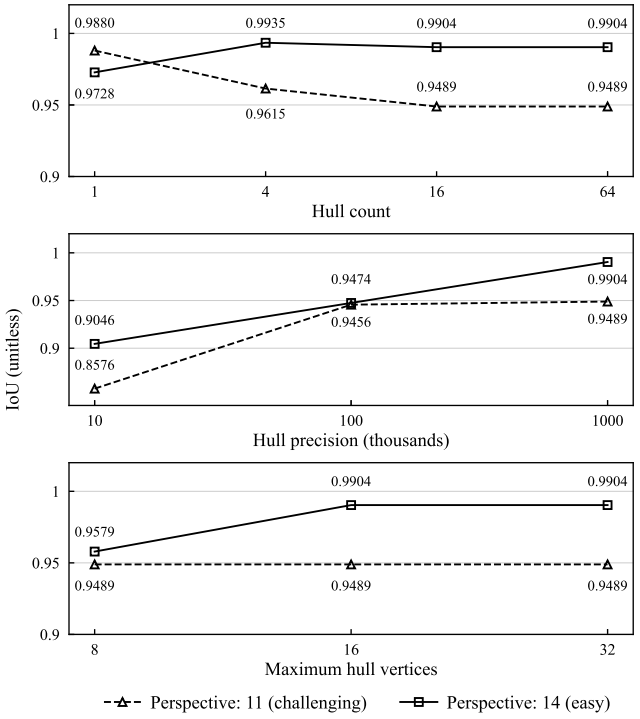

FIGURE 11. Intersection over Union (IoU) of the CCH-based bounding boxes of the Stanford Bunny for a "challenging" perspective 11 (△) and an "easy" perspective 14 (□) as a function of the convex-decomposition parameters Hull Count, Hull Precision, and Maximum Hull Vertices.

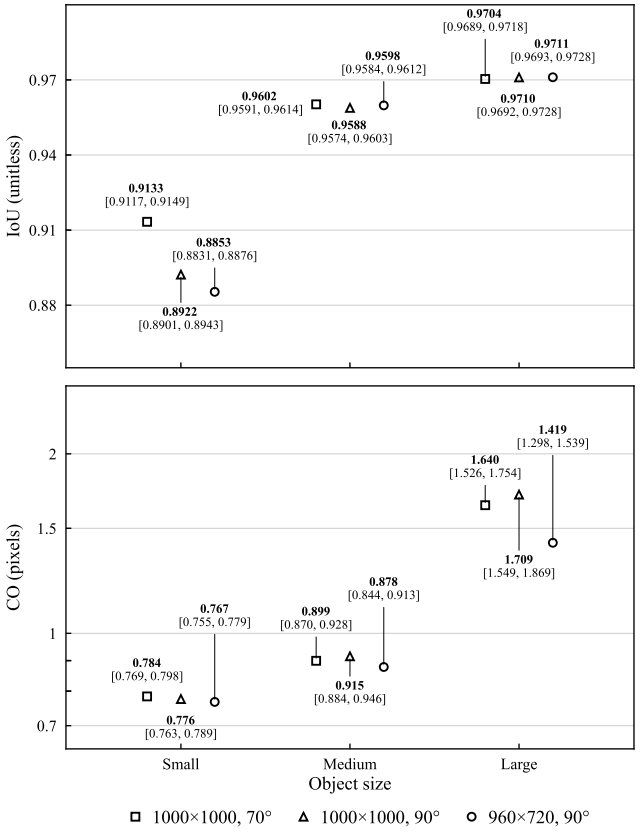

FIGURE 12. Mean Intersection over Union (IoU) and center offset (CO, logarithmic scale) of the CCH-based bounding boxes of the Stanford Bunny for the small, medium, and large object-size classes under three camera settings: 1000 × 1000 px with an FOV of 70° (□), 1000 × 1000 px with an FOV of 90° (△), and 960 × 720 px with an FOV of 90° (○). Numerical labels show the mean values with the corresponding 95% confidence intervals in brackets.

high execution time of the SEG-based approach should be interpreted in the context of the included GPU-to-CPU readback, and therefore represents the practical method-specific annotation cost rather than isolated CPU computation time.

D. PARAMETER SENSITIVITY

The first parameter-sensitivity experiment investigated the influence of the convex-decomposition parameters on the quality of the Stanford Bunny's CCH-based bounding box. For this purpose, the Hull Count, Hull Precision, and Maximum Hull Vertices were varied individually for the ''challenging'' perspective 11 and the ''easy'' perspective 14 defined in Table 3. For each operating point described here, only one measurement was taken, which is why this was a non-statistical, illustrative evaluation. The resulting IoU values are shown in Fig. 11.

The results indicate that the influence of these parameters depends on the viewing perspective. Increasing Hull Count did not result in a monotonic improvement in IoU. For the scenario with the challenging perspective, the IoU decreased from 0.9880 for one hull to 0.9489 for 16 and 64 hulls, whereas for the easy perspective, values between 0.9728 and 0.9935 were obtained. Hull Precision showed the clearest effect: increasing it from 10 000 to 1 000 000 increased the IoU from 0.8576 to 0.9489 for the challenging perspective, and from 0.9046 to 0.9904 for the easy perspective. Increasing Maximum Hull Vertices from 8 to 16 improved the IoU of the easy perspective from 0.9579 to 0.9904, whereas no further change was observed at 32 vertices. For the challenging perspective, the IoU remained 0.9489 for all three settings.

To complement the analysis of the convex-decomposition parameters, the sensitivity of the CCH-based annotation quality to selected camera settings was additionally examined in a second experiment using the Stanford Bunny object. Three combinations of image resolution and field of view were evaluated for the small, medium, and large object-size classes. The corresponding mean IoU and CO results based on 2000 independently drawn situations per camera setting are shown in Fig. 12. The data points were randomly selected for each setting.

The results indicate that the influence of the camera settings depends on the object-size class. For medium and large objects, the mean IoU remained nearly unchanged across the three settings, with largely overlapping 95% confidence intervals. In contrast, the mean IoU for small objects decreased from 0.9133 at 1000 × 1000 px and an FOV of 70° to 0.8853 at 960 × 720 px and an FOV of 90°. The CO showed only minor variations for small and medium objects,

whereas larger differences were observed for the large size class, with mean values ranging from 1.419 to 1.709 pixels. Because the three configurations do not form a full factorial design, interactions between resolution and FOV cannot be assessed. Nevertheless, the pairwise contrasts isolate the FOV and the resolution changes.

Overall, the experiments show that the CCH-based annotation quality is sensitive to the underlying convex-decomposition parameters. However, an increase in hull complexity does not necessarily lead to higher agreement with the segmentation-derived reference for every viewpoint. In particular, Hull Precision had the most consistent influence among the investigated parameter settings, whereas the effects of Hull Count and Maximum Hull Vertices were more dependent on the selected perspective. In comparison, the investigated camera settings had a comparatively limited influence on annotation quality, particularly for medium and large objects.

E. EDGE-CASE BEHAVIOR

A qualitative edge-case evaluation investigated the behavior of the CCH-based bounding box generation method under challenging situations in synthetic data generation. The Stanford Bunny was used as the test object. The investigated scenarios included occlusion, image-boundary truncation, transparent materials, overlapping objects, multiple static-mesh components, and near-clipping situations. Selected examples of the investigated edge cases are shown in Fig. 13.

For partial and severe occlusions, the CCH-based method generates a complete bounding box as long as the Stanford Bunny is successfully detected. If the occluding object was not classified as relevant, it would not appear in the segmentation map used for the preceding object detection stage. Consequently, the Bunny remained detected and was annotated even when fully occluded in the RGB image. In contrast, if the occluding object was classified as relevant and completely covered the Bunny in the segmentation map, the Bunny was no longer detected and no annotation was generated. Once parts of the Bunny became visible again, the CCH-based method included the occluded geometric extent in the resulting bounding box. The same behavior was observed for the transparent material configuration. This dependency on segmentation-based detection is a limitation of the current implementation. A not-yet-integrated line-of-sight test could provide a more consistent treatment of occlusion. In the overlapping-object scenario, the CCH projection continues to represent the complete projected geometry of the detected object. Actors consisting of several spatially separated static-mesh components are likewise enclosed by a single bounding box.

Image-boundary truncation revealed limitations when parts of the projected collision hull extended beyond the captured image region. In such cases, the generated bounding box did not necessarily fully enclose the visible object region because relevant projected vertices lay outside the image boundaries. A possible solution would be to perform the projection on an enlarged virtual projection plane and determine the final bounding box from the image boundaries and the outermost projected points of the visible object extent.

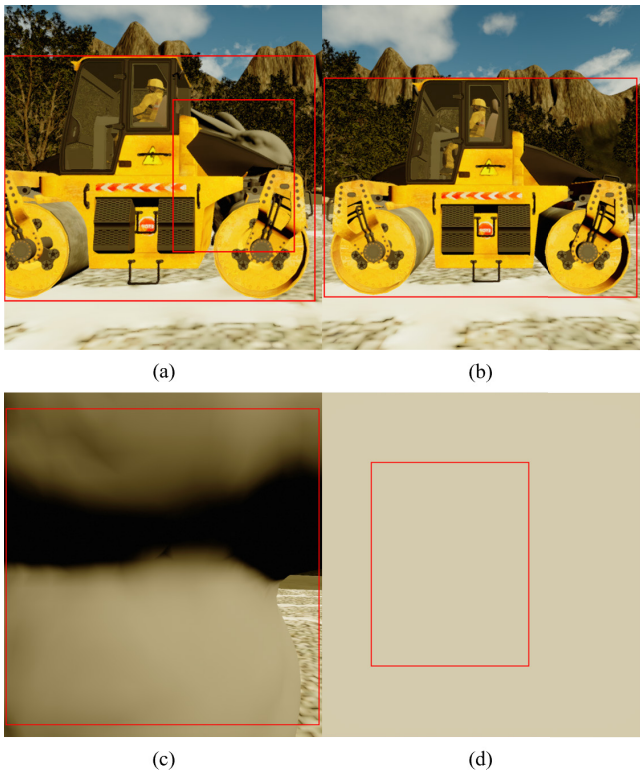

FIGURE 13. Selected edge cases for the CCH-based annotation: (a) Annotated Stanford Bunny severely occluded by the annotated Roller, (b) Stanford Bunny fully occluded by an annotated Roller, so that no Bunny annotation is generated, (c) near-clipping behavior of the Stanford Bunny with annotation, and (d) annotated Stanford Bunny fully occluded by a non-annotated plane.

Near clipping resulted in similar deviations when the Stanford Bunny extended to the image boundaries, whereas the generated CCH bounding box did not. The behavior became particularly critical once the near-clipping plane intersected the object geometry because the set of projectable vertices changed substantially. An enlarged projection plane could potentially reduce the deviations caused by points outside the captured image region, although additional handling may be required for the geometry directly affected by the near-clipping plane. Increasing the convex-decomposition detail improves the truncation and near-clipping results, confirming the dependence of the method on the quality of the underlying collision-hull representation.

F. LIMITATIONS OF THE METHOD

Nevertheless, the present results must be interpreted within the context of methodological limitations. First, the evaluation uses segmentation-derived bounding boxes as a reference and, therefore, primarily measures agreement with the visible object extent rather than with the complete geometric extent of an object in the image space. This distinction

becomes particularly relevant for occluded or truncated objects, because projection-based methods may include geometric regions that are not represented in the segmentation map. Therefore, all experiments, except for the edge-case analysis, were conducted without occlusions. Second, the accuracy of the proposed CCH-based method depends on the quality of the underlying collision-hull approximation. The parameter sensitivity analysis further suggests that increasing the convex-decomposition complexity does not necessarily improve the resulting 2D bounding box monotonically and that its influence can depend on the viewing perspective. In the future, parameter sensitivity analysis will require further investigation with a broader parameter portfolio, more than one object, and multiple viewpoints. Although most of the experiments were extended to six objects, 16 predefined viewpoints, three different object-size classes, and selected image-capture settings, they remain limited to a single controlled simulation scenario and a restricted set of object geometries and camera configurations. The investigated edge cases represent selected qualitative scenarios, rather than an exhaustive robustness evaluation. In particular, the current workflow depends on the preceding segmentation-based object identification stage, which can lead to inconsistent behavior for fully occluded objects, depending on whether the occluding object is annotated. Moreover, image-boundary truncation and near-clipping can lead to incomplete CCH-derived bounding boxes when relevant projected vertices lie outside the captured image region or are affected by the clipping plane. Finally, the computational-cost results were specific to the implemented pipeline and measurement setup, and the influence of the generated annotations on downstream object detector training was not evaluated.

V. CONCLUSION

In this study, we propose an alternative approach for generating synthetic image data for the training of object detectors, which uses the projection of convex collision hulls (CCHs) of virtual objects to generate 2D bounding boxes. First, the state of the art was analyzed. It became apparent that current game engines, owing to their graphical capabilities, are widely used as a suitable foundation for rendering synthetic image data. In addition, the use of segmentation maps for object identification has been adopted from related studies. Building on existing projection-based approaches, the CCH-based method offers an alternative that achieves a high annotation quality but remains computationally efficient. The approach was implemented using Unreal Engine 5 and evaluated against bounding boxes obtained from the oriented bounding box projection, full mesh projection, and segmentation maps without evaluating detector performance. The evaluation across six objects, different viewing perspectives, and object-size classes illustrated that the CCH projection provides more precise bounding boxes than the OBB projection. Across the small, medium, and large object-size classes, the mean paired IoU advantages of CCH over OBB were 0.212, 0.254, and 0.287, respectively. However, for simple geometries such as cubes the OBB projection already showed high agreement with the reference. For the Cube, the Auto Convex Collision routine of the Unreal Engine returns a single box, so the CCH and OBB coincide by construction, apart from negligible numerical differences. Therefore, the Cube contributed null differences to the pooled comparison, making the reported advantage of CCH over OBB conservative. Compared with FM projection, the CCH-based method achieved similar IoU values, whereas FM required approximately 55 times the execution time of CCH. OBB remains computationally less expensive than CCH but at the cost of considerably reduced annotation quality for more complex object geometries. These results therefore support the use of CCH projection as a practical compromise between the computationally inexpensive but geometrically coarse OBB projection and the more expensive full mesh projection, while achieving an annotation quality comparable to that of the FM-based method. Additional experiments indicate that the CCH-based annotation quality depends on the underlying convex-decomposition parameters and that increasing the hull complexity does not necessarily improve the resulting 2D bounding box. Within the investigated camera settings, the annotation quality remained comparatively stable, particularly for medium and large objects. Simultaneously, the results in the evaluation of the annotation quality must be interpreted with respect to the segmentation-derived, visibility-based reference, the restricted set of objects and camera configurations investigated in a single, occlusion-free road-intersection scenario, and the selected qualitative edge cases. In future work, the treatment of occlusion could be extended by incorporating a line-of-sight test, whereas image-boundary truncation and near-clipping cases could benefit from improved handling of the projected geometry outside the captured image region. Further work could also investigate the adaptive selection of convex-decomposition parameters, extend the evaluation to a broader range of objects and scenarios, and assess the influence of the generated annotations on downstream object-detector training. Thus, this study can also be viewed as a basis for a broader benchmark of projection-based annotation methods.

ACKNOWLEDGMENT

The authors wish to express their gratitude to their colleagues at the respective institutions. They acknowledge the use of OpenAI ChatGPT and Anthropic Claude for language editing and improving clarity, translation, error checking, and assistance with Python code for selected plots. All results were reviewed and verified by the authors.

REFERENCES

[1] D. Dolgov. (2021). *How we've built the World's Most Experienced Urban Driver*. Waymo. Accessed: Apr. 20, 2026. [Online]. Available: https://waymo.com/blog/2021/08/most-experienced-urban-driver

[2] M. de Fries, M. Irmer, S. Röttgermann, K. Thomas, and R. Degen, "A methodology for image reproduction in virtual environments applied to an agricultural assistance function," *Results Eng.*, vol. 28, Dec. 2025, Art. no. 108313, doi: 10.1016/j.rineng.2025.108313.

[3] Y. Zhang. (2025). *Digital Twins: The Ultimate Virtual Proving Ground*. Volvo Autonomous Solutions. Accessed: Apr. 20, 2026. [Online]. Available: https://www.volvoautonomoussolutions.com/en-en/news-and-insights/insights/articles/2025/jun/digital-twins--the-ultimate-virtual-proving-ground.html

[4] Q. Song, E. Engström, and P. Runeson, "Industry practices for challenging autonomous driving systems with critical scenarios," *ACM Trans. Softw. Eng. Methodol.*, vol. 33, no. 4, pp. 1–35, Apr. 2024, doi: 10.1145/3640334.

[5] G. Lou, Y. Deng, X. Zheng, M. Zhang, and T. Zhang, "Testing of autonomous driving systems: Where are we and where should we go?" in *Proc. 30th ACM Joint Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, New York, NY, USAMachinery, Nov. 2022, pp. 31–43, doi: 10.1145/3540250.3549111.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788, doi: 10.1109/CVPR.2016.91.

[7] C. Liu, K. Wang, H. Lu, Z. Cao, and Z. Zhang, "Robust object detection with inaccurate bounding boxes," in *Computer Vision–(ECCV)*, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds., Cham, Switzerland: Springer, 2022, pp. 53–69, doi: 10.1007/978-3-031-20080-9_4.

[8] J. Murrugarra-Llerena, L. Kirsten, and C. R. Jung, "Can we trust bounding box annotations for object detection?" in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, New Orleans, LA, USA, Jun. 2022, pp. 4812–4821, doi: 10.1109/CVPRW56347.2022.00528.

[9] H. Schieber, K. Can Demir, C. Kleinbeck, S. H. Yang, and D. Roth, "Indoor synthetic data generation: A systematic review," *Comput. Vis. Image Understand.*, vol. 240, Mar. 2024, Art. no. 103907, doi: 10.1016/j.cviu.2023.103907.

[10] T. Pollok, L. Junglas, B. Ruf, and A. Schumann, "UnrealGT: Using unreal engine to generate ground truth datasets," in *Advances in Visual Computing*, G. Bebis et al., Eds., Cham, Switzerland: Springer, 2019, pp. 670–682, doi: 10.1007/978-3-030-33720-9_52.

[11] Y. Wang, W. Deng, Z. Liu, and J. Wang, "Deep learning-based vehicle detection with synthetic image data," *IET Intell. Transp. Syst.*, vol. 13, no. 7, pp. 1097–1105, Jul. 2019, doi: 10.1049/iet-its.2018.5365.

[12] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Annu. Conf. Robot Learn.*, vol. 78, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., Cambridge, MA, USA: PMLR, 2017, pp. 1–16. [Online]. Available: https://proceedings.mlr.press/v78/dosovitskiy17a.html

[13] T. Bu, X. Zhang, C. Mertz, and J. M. Dolan, "CARLA simulated data for rare road object detection," in *Proc. IEEE Int. Intell. Transp. Syst. Conf. (ITSC)*, Sep. 2021, pp. 2794–2801, doi: 10.1109/ITSC48978.2021.9564932.

[14] J. Jang, H. Lee, and J.-C. Kim, "CarFree: Hassle-free object detection dataset generation using Carla autonomous driving simulator," *Appl. Sci.*, vol. 12, no. 1, p. 281, Dec. 2021, doi: 10.3390/app12010281.

[15] L. Laux, S. Schirmer, S. Schopferer, and J. Dauer, "Build your own training data—Synthetic data for object detection in aerial images," in *Proc. SE Workshops*. Bonn, Germany: Gesellschaft für Informatik, 2022, pp. 182–190, doi: 10.18420/se2022-ws-18.

[16] S. T. Brassai, N. Szántó, A. Bajka, O. Bárdi, A. Németh, and A. Hammas, "Simulation environment implementation for generation of training samples," in *Proc. 25th Int. Carpathian Control Conf. (ICCC)*, May 2024, pp. 1–6, doi: 10.1109/ICCC62069.2024.10569502.

[17] T. To. (2018). *NDDS: NVIDIA Deep learning Dataset Synthesizer*. NVIDIA Corporation. Accessed: Apr. 20, 2026. [Online]. Available: https://github.com/NVIDIA/Dataset_Synthesizer

[18] W. Remmas, M. Lints, and J. J. Uudmäe, "PCGOD: Enhancing object detection with synthetic data for scarce and sensitive computer vision tasks," *IEEE Access*, vol. 13, pp. 91325–91333, 2025, doi: 10.1109/ACCESS.2025.3572719.

[19] S. Borkman, A. Crespi, S. Dhakad, S. Ganguly, J. Hogins, Y.-C. Jhang, M. Kamalzadeh, B. Li, S. Leal, P. Parisi, C. Romero, W. Smith, A. Thaman, S. Warren, and N. Yadav, "Unity perception: Generate synthetic data for computer vision," 2021, *arXiv:2107.04259*.

[20] J. Jaspar, E. Viennet, D. Gualandris, J.-L. Sauvaget, and F. Fogelman-Soulié, "Using synthetic images to improve and test object detection in the context of the autonomous vehicle," in *Proc. 12th Eur. Workshop Vis. Inf. Process. (EUVIP)*, Sep. 2024, pp. 1–6, doi: 10.1109/EUVIP61797.2024.10772859.

[21] L. Eversberg and J. Lambrecht, "Generating images with physics-based rendering for an industrial object detection task: Realism versus domain randomization," *Sensors*, vol. 21, no. 23, p. 7901, Nov. 2021, doi: 10.3390/s21237901.

[22] X. Li, J. Park, C. Reberg-Horton, S. Mirsky, E. Lobaton, and L. Xiang, "Photorealistic arm robot simulation for 3D plant reconstruction and automatic annotation using unreal engine 5," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2024, pp. 5480–5488, doi: 10.1109/CVPRW63382.2024.00557.

[23] CARLA Team. *Bounding Boxes*. CARLA Unreal Engine 5 Documentation. Accessed: Sep. 17, 2026. [Online]. Available: https://carla-ue5.readthedocs.io/en/latest/tuto_G_bounding_boxes/#bounding-boxes

[24] Epic Games. *UBodySetup*. Epic Games Developer Documentation. Accessed: Apr. 24, 2026. [Online]. Available: https://dev.epicgames.com/documentation/unreal-engine/API/Runtime/Engine/UBodySetup

[25] *FKConvexElem*. Epic Games Developer Documentation. Accessed: Apr. 24, 2026. [Online]. Available: https://dev.epicgames.com/documentation/unreal-engine/API/Runtime/Engine/FKConvexElem

[26] G. Turk. (2000). *The Stanford Bunny*. Georgia Institute of Technology. Accessed: Apr. 20, 2026. [Online]. Available: https://faculty.cc.gatech.edu/~turk/bunny/bunny.html

[27] Epic Games. *Add a Collision Hull to a Static Mesh Using the Auto Convex Collision Tool in Unreal Engine*. Epic Games Developer Documentation. Accessed: Apr. 20, 2026. [Online]. Available: https://dev.epicgames.com/documentation/unreal-engine/add-a-collision-hull-to-a-static-mesh-using-the-auto-convex-collision-tool-in-unreal-engine

[28] P. Dollar and T.-Y. Lin. (2015). *Cocoeval.Py*. COCO API. Accessed: Apr. 20, 2026. [Online]. Available: https://github.com/cocodataset/cocoapi/blob/master/PythonAPI/pycocotools/cocoeval.py

MARTIN DE FRIES received the M.Sc. degree in mechatronics from Cologne University of Applied Sciences, Germany, in 2022. He is currently pursuing the Ph.D. degree in cooperation with Uppsala University, Sweden. Since 2022, he has been a Research Associate with Cologne University of Applied Sciences, where he is involved in and leading several research projects. His research interests include the development of virtual test methods, simulator development, and data generation for sensor-based systems for environment perception. In this context, he focuses on virtual strategies for prototyping and testing future vehicle technologies.

DANYEL RIDVANOĞULLARI received the B.Eng. and M.Sc. degrees in automotive engineering from Cologne University of Applied Sciences, Germany, in 2021 and 2024, respectively. Before completing the M.Sc. degree, he completed several internships in the fields of inspection and vehicle development. He was a Research Associate with Cologne University of Applied Sciences. In 2025, he joined TÜV Rheinland Group, where he is currently an Inspection Engineer and an Officially Recognized Technical Expert. During his thesis work and research activities, he focused on vehicle simulation in virtual test fields. His current work includes the inspection of vehicles for their safe operation.

MARTIN MEINERS received the M.Sc. degree from Cologne University of Applied Sciences, Germany. He is currently pursuing the joint Ph.D. degree in cooperation with Uppsala University, Sweden, and Cologne University of Applied Sciences. Since 2023, he has been a Research Associate with the Automotive Safety and Innovation Laboratory (ASIL-Lab), Cologne. His current work focuses on the virtual testing and validation of automated driving systems, with an emphasis on X-in-the-Loop (XiL) methods, including driving simulators and real-time target machines.

TONI VISCIDO received the Dipl.-Ing. degree in electrical engineering and the Dr.-Ing. degree from RWTH Aachen University, Aachen, Germany. From 2001 to 2010, he was with FKA—Forschungsgesellschaft Kraftfahrwesen Aachen mbH, where he held positions, including the Head of Energy Efficiency and the Head of the Business Area Vehicle E/E. From 2010 to 2014, he was with DSA—Daten- und Systemtechnik Aachen GmbH, Aachen, as the Head of Project and Product Management and the Head of Sales and Business Development. Since 2014, he has been a Professor of vehicle electrics and electronics with Cologne University of Applied Sciences, Germany. His research interests include vehicle electronics, vehicle communication and diagnostics, energy management, telematics, car IT, and data-driven mobility systems.

KARIN THOMAS (Member, IEEE) received the master's degree in engineering physics and the Ph.D. degree in engineering physics with a specialization in electricity from Uppsala University, Sweden, in 2003 and 2008, respectively. She was appointed a Docent in engineering science with specialization in electricity, in 2020. She is currently an Associate Professor with the Department of Electrical Engineering, Uppsala University; and a Professor with Mälardalen University, Sweden. Previously, she was an Assistant Professor and a Development Engineer with Wave Power Company. She is also involved in research related to power systems stability, grid connection of distributed renewable electricity generation, and charging of electric vehicles.

RENÉ DEGEN received the M.Sc. degree in mechatronics from Cologne University of Applied Sciences, Germany, in 2018, and the Ph.D. degree from Uppsala University, in 2023. From 2017 to 2020, he was a Research Associate with Cologne University of Applied Sciences. In 2020, he became a Chief Engineer with the CAD CAM Center Cologne for the field of mechatronics. In 2022, he founded the Rhineland Mechatronics Academy, which connects academic and industrial activities within one company. He later established Heavy Path, where he is also the Chief Executive Officer, focusing on the simulation of heavy transport operations. In parallel, he continues his research in the fields of virtual test environments and future mobility systems in cooperation with Uppsala University.