

Scaling Realistic Flight Simulation and Data Synthesis with Generative 3D Reconstruction and Automatic Annotation

Can Vu and Peng Wei

Mechanical and Aerospace Engineering Department
The George Washington University
{can.vu, pwei}@gwu.edu

Abstract—As autonomous unmanned aircraft system (UAS) operations increasingly rely on deep learning computer vision modules in their avionics, there is a need for scalable workflows to generate robust test and validation datasets. While high-fidelity simulation platforms offer a cost-effective alternative to expensive real-world data collection, their utility is constrained by two bottlenecks: the intensive manual labor required to model realistic 3D environments, and the time-intensive nature of high-fidelity data labeling. Although simulator-native APIs allow for automated labeling, they frequently produce coarse or inaccurate annotations. To address these challenges, this paper proposes a scalable synthetic data generation workflow that integrates the state-of-the-art generative 3D reconstruction model SAM 3D to synthesize high-fidelity mesh assets directly from real-world images. Furthermore, we implement a segmentation-based refinement algorithm that enhances the precision of simulator-generated annotations. Experimental results demonstrate that the proposed workflow reduces manual intervention and enhances the scalability of generating diverse suburban neighborhood flight simulation environments. Furthermore, the label refinement algorithm yields higher-precision annotations for robust perception model validation.

Index Terms—Computer Vision, Generative AI, 3D Reconstruction, Synthetic Data Generation, Automatic Annotation.

I. INTRODUCTION

Recent studies integrating in deep learning-based computer vision into last-mile autonomous package delivery via small Unmanned Aircraft Systems (UAS) have demonstrated promising results in tasks such as landing site identification [3]–[5] and aerial-view obstacle sensing [6]–[8]. These models require large amounts of high-quality, labeled image data for supervised learning and evaluation to ensure the system meets robust performance and safety requirements in uncertain environments. Such assurances are a prerequisite for certifying autonomous UAS landings during package deliveries [9]. However, acquiring diverse image datasets from real-world flight tests is often cost-prohibitive and difficult to scale.

Physics-based simulation frameworks [10]–[12] built on modern rendering engines offer a cheaper alternative to the real-world UAS flight data collection. These UAS simulation platforms simulate physically accurate, high-fidelity environments. However, a bottleneck in the development of flight simulation

environments is the need for high-quality 3D mesh models to form realistic flight scenarios. Structurally accurate meshes are essential for synthesizing interactive environments, as they provide the volumetric and spatial data necessary for physics engines to enforce kinematic constraints and accurately simulate complex system-environment interactions. Meshes must be meticulously created to include not only high-resolution photorealistic textures, but also precise collision geometries, physically based rendering (PBR) materials, and accurate semantic tags. This requires 3D mesh modeling expertise, expensive 3D modeling software, and substantial time investment per mesh asset, limiting the scalability of high-fidelity flight simulation environment and synthetic data generation for vision model test and validation.

Additionally, manual annotation remains the predominant method for generating dataset labels. The process is slow and error-prone; large-scale dataset annotation typically requires significant human labor, domain expertise, and extensive quality control procedures. Automatic dataset labeling tools have become popular for image annotation at scale. Modern workflows integrate foundation models to speed up the labeling process by passing the labor to AI models; however, these workflows still require humans-in-the-loop for quality control to ensure consistency, correctness, and coverage of the produced labels. While image dataset generation via simulator frameworks benefits from prior knowledge of object characteristics and offers capabilities for automatic annotation using geometric transformations, inaccurate labels can be produced due to the projection methods used to generate 2D bounding boxes from 3D geometry.

In this paper, we address the scalability of simulated environment generation and annotation bottlenecks by:

- 1) Proposing a synthetic dataset generation workflow that integrates a state-of-the-art single image generative 3D reconstruction to rapidly synthesize high-quality object mesh models for a simulated suburban neighborhood 3D environment. We show that the proposed workflow can rapidly produce high-fidelity meshes and speed up diverse 3D environment modeling using Cosys-AirSim [2] and Unreal Engine 5 [1].
- 2) Proposing an annotation refinement algorithm designed

We thank our Federal Aviation Administration (FAA) colleagues for their continued funding and support for our research.

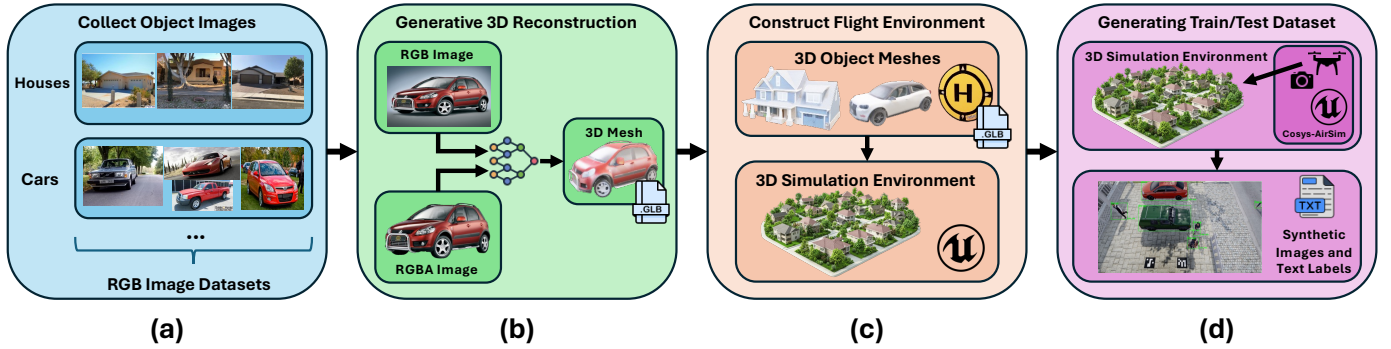


Fig. 1. Our synthetic data generation workflow integrates generative AI with high-fidelity simulation to automate the production of labeled datasets. (a) Curated sets of neighborhood object images are collected from existing datasets. (b) Sequential processing of object images through a 3D reconstruction model generates large quantities of mesh assets. (c) Meshes are imported into and populate the Unreal Engine 5 [1] simulation environment. (d) Data acquisition is performed with Cosys-AirSim [2], which facilitates the capture and labeling of synthetic flight images from simulated UAS flights.

to improve the automated labeling capabilities available in Cosys-AirSim. By processing data downstream of the Cosys-AirSim annotation method, this reduces the percentage of poorly annotated objects introduced by standard projection techniques in the synthetic dataset.

II. RELATED WORKS

A. 2D-to-3D Mesh Reconstruction

3D mesh reconstruction from 2D observations is a fundamentally ill-posed problem, as it requires inferring occluded geometry and complex topologies from limited, low dimensional sets of 2D images. Algorithms based on Multi-View Stereo (MVS) [15] or Structure-from-Motion (SfM) [16], [17] achieve high accuracy in reconstructing intricate geometries, they necessitate complex pipelines involving feature matching, camera calibration, and optimization. Furthermore, these multi-view techniques rely on numerous overlapping images to ensure geometric consistency and result in high data acquisition costs and computational bottlenecks. Consequently, recent research has explored deep learning and generative frameworks for end-to-end 2D-to-3D reconstruction from single image inputs. By removing the need for dense overlapping-view images and reducing the computational overhead of asset creation, these methods hold promise for enabling rapid generation of unique assets and environments at scale.

Sinha et al. [18] introduced SurfNet, utilizing deep residual networks to generate 3D surfaces through spherical parametrization. This parameterization inherently restricted the modeling of shapes with high genus or complex topologies. Pontes et al. [19] proposed Image2Mesh, which uses sparse linear combinations of models parameterized by embedding graphs to reconstruct 3D meshes; however, a sufficient graph representation was needed to create high-quality models. Pan et al. [20] addressed topological constraints through Topology Modification Networks, deforming meshes from a genus-0 template to approximate the target surface. Their method was reported to struggle with non-closed meshes.

Other works explored geometric priors and implicit representations to enhance reconstruction fidelity without multi-view

data. Wu et al. [21] utilized structural symmetry of objects for unsupervised learning of deformable objects, an approach limited to symmetric classes. Xu et al. [22] applied quasi-Monte Carlo sampling to improve deep implicit field-based reconstructions, also leveraging symmetry constraints. Guillard et al. [23] introduced UCLID-Net to operate directly in object space, improving the alignment between 2D inputs and 3D outputs. Tang et al. [24] used the topology-preserved, skeletal shape representation for object surface reconstruction from RGB images. Wolf et al. [25] proposed GS2Mesh to bridge photorealistic rendering and geometric accuracy, extracting surface meshes from 3D Gaussian Splatting representations via stereo-aligned views. Their method was noted to produce noisy Gaussians in areas not covered by the training images, difficulty with transparent surfaces, and scalability issues. Advancements in transformer architectures enabled rapid feed-forward generation. Tochilkin et al. [26] introduced TripoSR, which achieved high-quality mesh generation with inference times under 0.5 seconds. Boss et al. [27] extended these rapid feed-forward capabilities with SF3D, integrating fast UV-unwrapping and illumination disentanglement to produce production-ready assets. Despite their speed, feed-forward transformers can produce smoothed or ambiguous geometry in heavily occluded regions.

Large-scale foundation models have further scaled single-image reconstruction to complex, in-the-wild backgrounds. Meta introduced SAM 3D [13], a new foundation model for 3D that predicts object shape, texture, and pose directly from real-world scenes using a single image. Additionally, Yang et al. [28] extended these robust generative capabilities to full-body human mesh recovery with SAM 3D Body. Both SAM 3D foundation models demonstrate strong improvements to structural fidelity and textural consistency compared to its peers. Xiang et al. introduced TRELIS [29] and TRELIS.2 [30], which use native, compact structured latents and omni-voxel representations to capture intricate topological details to improve 3D generation efficiency while maintaining quality. Yao et al. [31] provided initial benchmarking results between SAM 3D and TRELIS, observing that SAM 3D produced more coherent reconstructions of buildings given aerial-view

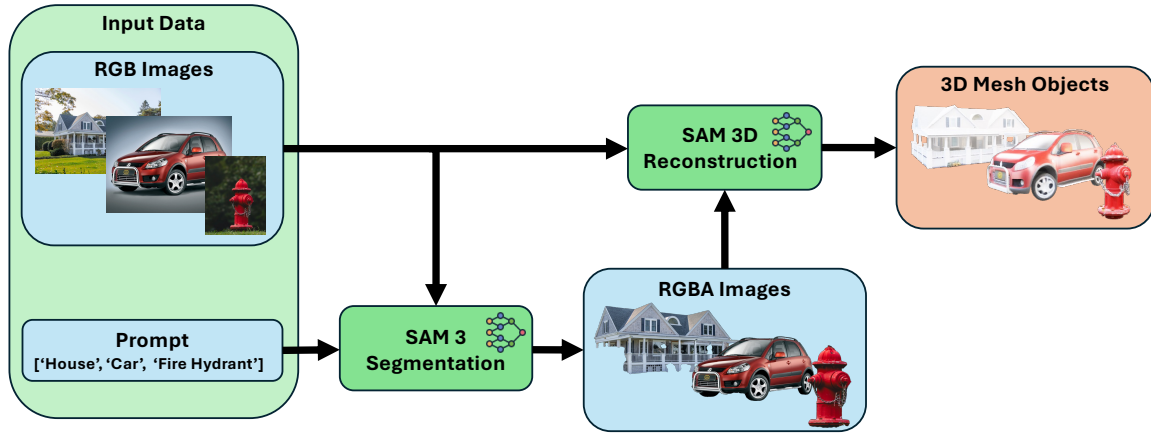


Fig. 2. Generative mesh reconstruction pipeline using SAM 3D [13]. An RGB source image is provided with a corresponding segmented RGBA mask to the generative model to synthesize high-fidelity mesh 3D assets. Object-level 2D segmentation is facilitated by the SAM 3 foundation model [14]. To facilitate large-scale environment population, the pipeline sequentially processes large-scale object datasets to rapidly generate diverse object libraries for high-fidelity 3D simulation.

images of building structures. Based on these promising results, and the need for a scalable workflow for fast and high-fidelity mesh assets, we incorporate a single-image generative 3D mesh reconstruction model into our synthetic data generation workflow.

B. UAS 3D Flight Simulation Frameworks

Gazebo [11] established foundational paradigms for open-source multi-robot simulators through a modular architecture and reliable physics engines. More recently, simulators based on modern game engines have shown promise for creating realistic and accurate environments [32]. In 2017, Microsoft introduced AirSim [33], which integrated Unreal Engine to provide photorealistic visual and physical simulations. Song et al. [12] proposed Flightmare, a flexible quadrotor simulator that decoupled rendering from physical dynamics to facilitate high-speed simulation for reinforcement learning. In 2025, Mittal et al. [10] introduced Isaac Lab, a GPU-accelerated framework that leverages massive parallelism to achieve high throughput for multi-modal robot learning.

After the deprecation of the original AirSim repository, further research and development into the AirSim framework was aimed at improving the software suite for specialized and scalable applications. Community-driven initiatives such as ProjectAirSim [34] have emerged to maintain the foundational AirSim architecture and ensure its compatibility with modern Unreal Engine 5 features. In contrast, other forks like Cosys-AirSim [2] have prioritized research-grade sensor expansion, introducing robust support for thermal imaging and advanced LiDAR models to facilitate complex industrial and multi-modal perception studies. The AirGen framework [35] evolved the AirSim ecosystem by focusing on the automated generation of environments for large-scale agentic AI training.

C. Synthetic Data Generation for Object Detection

Due to the large overheads associated with real-world data collection, much research has been devoted to exploring syn-

thetic generation alternatives. The SYNTHIA dataset [36] demonstrated that pixel-perfect semantic labels could be automatically extracted from rendered urban scenes. Kiefer et al. [37] and Barisic et al. [38] demonstrated the viability of using synthetic aerial datasets (such as Sim2Air) to train autonomous UAV monitoring systems. By utilizing 3D rendering software to simulate aerial perspectives, these works automatically extracted bounding box annotations directly from the spatial coordinates of the 3D models. However, the rendering approaches often suffered from a Sim-to-Real gap due to simplistic texturing and lack of complex environmental lighting.

To address the limitations of static rendering and expand the Operational Design Domain (ODD) of perception models, subsequent research integrated dynamic simulation environments. Rüter [39] investigated the use of synthetic low-altitude images to detect humans for safe UAS landing assessments. data generation was executed through a combination of 2D image stitching, traditional augmentations, and 3D simulation. Their results indicated simple 2D augmentations were sometimes more effective at covering ODD gaps than fully synthetic images, highlighting ongoing discrepancies in rendering realism. Similarly, Ding et al. [40] leveraged Microsoft AirSim, an Unreal Engine-based platform, to generate synthetic data for small UAS package delivery. By directly querying the game engine, synthetic image data of dynamic landing scenarios with tunable weather, time-of-day, and geometric parameters was automatically generated to evaluate machine learning vision models. A limitation of this approach was the laborious manual effort required for scenario specification; environment assets were handcrafted within the engine, restricting the scalability and diversity of the resulting datasets.

More recently, Zhou and Li [41] used local realism and global domain randomization to build the simulation environment, applied computer graphics techniques to synthesize images and automatically annotate target objects. Lenhard et al. [42] introduced SynDroneVision, a large-scale synthetic dataset

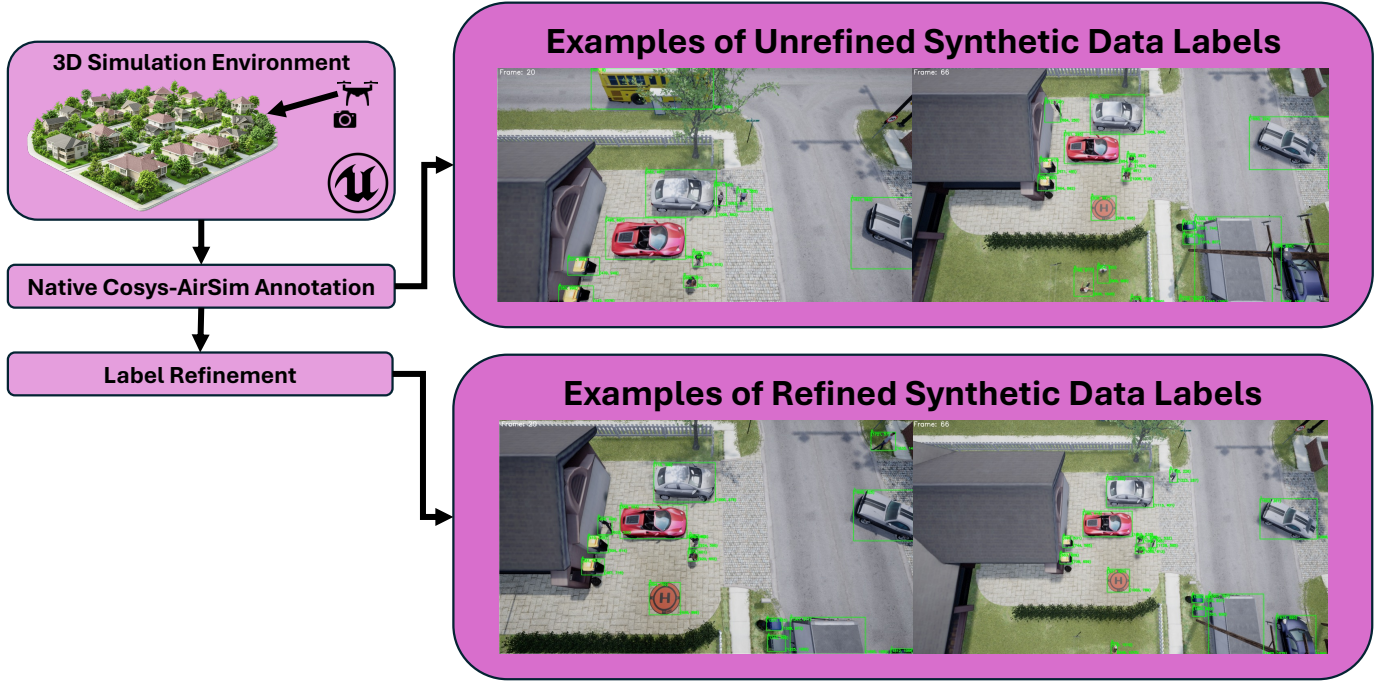


Fig. 3. (Left) Integration of generative 3D assets within the Unreal Engine 5 (UE5) environment, where automated flight paths are executed to capture raw imagery to pass to the annotation and label refinement stages. (Upper Right) Initial bounding boxes generated via the AirSim detection API, exhibiting background noise and loose containment of the bounding boxes. (Bottom Right) The refined bounding boxes after applying the proposed refinement algorithm, demonstrating tighter spatial alignment and improved precision around the objects-of-interest in the scene.

designed for RGB-based drone detection. Game engine simulations allowed rapid iteration through diverse backgrounds, lighting conditions, and target models. They showed that synthetically enriched datasets significantly improve the robustness of state-of-the-art YOLO detectors. Despite the advancements in automated annotation and high-fidelity rendering, modern game engine-based approaches remain constrained by the availability of diverse, high-quality 3D assets.

Because virtual environments rely on pre-modeled objects with manually designed topologies, the resulting synthetic datasets often lack diverse edge cases and varied structural geometries that can be encountered in real-world UAS operations. The proposed workflow enables high-throughput creation of diverse ground assets and UAS simulation scenarios, leveraging real-world object images to improve the visual authenticity of synthetically generated datasets for robust vision model validation. Furthermore, the workflow improves the scalability and precision of object annotations, addressing the bottlenecks associated with manually labeling high volumes of images.

III. METHODS

In this section, the details of our proposed synthetic data generation workflow are given. The overall workflow is shown in Fig. 1. It starts by curating the input image datasets of common neighborhood objects (see in Section IV for sourcing details). These input images are processed through the SAM 3D object reconstruction pipeline, producing meshes to be imported into the AirSim environment. Synthetic flight data is

captured and automatically annotated to form the final labeled dataset that can be used for on-board vision model safety test and validation. Descriptions of the main components of the workflow are given in the subsequent sections below.

A. Generative 3D Reconstruction for Suburban Object Meshes

SAM 3D is a set of foundation models released in late 2025 by Meta: SAM 3D Objects [13] and SAM 3D Body [28]. Both are generative neural networks for visually grounded 3D reconstruction that predicts layout, texture, and geometry of objects from a single image. Additionally, SAM 3D Body performs human mesh recovery using a novel parametric mesh representation, Momentum Human Rig, which decouples skeletal structure and surface shape. For this paper, we focus on the use of SAM 3D Objects as our mesh reconstruction model and is referred to as SAM 3D throughout for conciseness.

SAM 3D considers the following problem formulation: Let an object be specified by shape S , texture T , rotation r , translation t , and scale s in an image I . Let M be the segmented mask of the object. Here (S, T) are the object characteristics and (r, t, s) is the object's pose in camera coordinates. The reconstruction problem is modeled by the conditional distribution $p(S, T, r, t, s | I, M)$ and the generative model $q(S, T, r, t, s | I, M)$ approximates $p(S, T, r, t, s | I, M)$ as closely as possible.

The SAM 3D architecture employs a two-stage latent flow matching framework consisting of a Geometry Model and a Texture & Refinement Model. The Geometry Model utilizes



Fig. 4. Samples of houses suburban neighborhood environment in Unreal Engine 5 that contains a mixture of both 3D reconstructed meshes and available mesh assets from the Epic Games ecosystem. The left most images in both rows are the original high-resolution RGB image provided as inputs to SAM 3D for mesh reconstruction. The images in the second and third columns are the recreations of the original input image with additional 3D assets such as cars, pedestrians, and landing pads. The additional assets can be re-positioned, or exchanged with other assets to create more diverse scenes. Different times-of-day were captured to qualitatively show the diversity of the data captured through the simulation.

a Mixture-of-Transformers architecture to predict a coarse 3D voxel representation of the object and the (r, t, s) parameters. The Texture & Refinement Model uses the coarse voxel shape, RGB image, and RGBA mask, to refine the geometric details and to synthesize the object texture. A pair of variable autoencoders are used to decode the latent representations from the Texture & Refinement Model into either 3D Gaussian splats [17] or meshes.

In this study, we focus on synthesizing 3D meshes for integration into our simulated environment using a multi-stage inference pipeline. First, source images of common neighborhood objects are processed through the Segment Anything 3 model (SAM 3) [14] to isolate the object(s) of interest (OOI) specified by a set of simple text prompts. SAM 3 generates precise segmentation masks for the OOI, resulting in an RGBA image where the alpha channel A defines the object’s boundary and transparency relative to the background. Examples of the RGB and RGBA inputs are given in Fig. 2.

The RGBA and RGB color images are subsequently passed to SAM 3D, which reconstructs the asset’s geometry and surface texture and yields reconstructed mesh assets in a binary file form of the Graphics Library Transmission Format (GLTF). The GLTF binary (GLB) stores the mesh model, texture maps, and material properties required for high-fidelity rendering in a self-contained file, reducing the storage requirement of mass produced mesh assets. This pipeline is executed on large-scale datasets by sequentially passing the input images to SAM 3D. Because GLB natively encapsulates both the 3D geometry and the synthesized textures into a single file, the assets are directly imported into the Unreal Engine environment to construct the baseline suburban neighborhood.

B. Airsim Unreal Engine Flight Environment Generation

In this study, we develop a simulated last-mile UAS package delivery environment using Unreal Engine 5 with the Cosys-Airsim plugin. Cosys-AirSim [2] is an updated fork of the original Microsoft AirSim repository maintained by Cosys Labs. Upgraded to support Unreal Engine 5 [1], this version enhanced multiple aspects of the original platform, introducing a refined API, highly accurate vehicle simulation, and a broader suite of sensor models. Furthermore, its environmental simulation capabilities have been expanded to incorporate runtime procedural generation tools and dynamic changeable objects.

Following the generation of candidate 3D meshes via the SAM 3D pipeline, the resulting 3D mesh assets were imported and positioned into the simulated neighborhood environment. The scale of the imported objects were adjusted so that object sizes to ensure consistent scale relative to the surrounding scene architecture and UAS dimensions. To simulate the UAS landing area, a 3D reconstructed orange landing pad was placed and scaled accordingly on the front lawn. This direct placement approach enabled us to design baseline suburban neighborhoods tailored exactly to our spatial requirements. Time-of-day and weather were applied prior to UAS simulation and image data capture. Examples of three different neighborhood environments are shown in Fig. 4 and 5.

To introduce necessary variance into the map level topology, we populated the static environment with pedestrian and vehicular actors. We combined the SAM 3D generated meshes with readily available assets from the Epic Games Fab store; thus, we can create realistic environments for simulated UAS package delivery and landing operations. Leveraging Cosys-AirSim’s simulation tools, these entities were configured as dynamic



Fig. 5. Simulated suburban environment for a UAS package delivery scenario. The scene integrates 3D assets reconstructed via the SAM3D pipeline (e.g., residential house, vehicles, and school bus) with supplemental high-fidelity assets from the Epic Games ecosystem. Moving actors were placed in the environment with pre-defined spline paths to simulate a dynamic suburban package delivery landing scenario. This hybrid environment serves as the high-fidelity testbed for synthetic data collection and automated ground-truth annotation procedures.

actors, programmed to navigate throughout the neighborhood. The inclusion of these moving elements creates a dynamic and complex suburban domain with varied spatial conditions. Finally, portions of the neighborhoods are generated by utilizing the prompt capabilities of SAM 3. A prompt is specified list to SAM 3 to produce RGBA masks of different objects of the scene. An example of this list is given in Fig. 2 for segmenting “vehicle” objects. Each of these masks are provided to SAM 3D for mesh generation. With this approach, we can approximate the original input image scenes in our 3D environment.

C. Bounding Box Label Refinement

The label generation pipeline is a two-stage annotation and refinement process to overcome inaccuracies inherent to the default AirSim annotation capabilities. Labels are defined as a vector

$$L = [C, x_{center}, y_{center}, w, h] \quad (1)$$

where C is the object class index and (x_{center}, y_{center}) , w , h are the bounding box center coordinates, width, and height respectively. In this paper, the terms label and annotation are used interchangeably. Initially, bounding boxes are generated using the AirSim annotation functions, which queries the environment for pre-defined target assets and projects their 3D extents onto 2D camera coordinates. These bounding boxes can exhibit spatial drift due to camera field-of-view (FOV) changes, leading to boxes that enclose too much of the background; therefore, a segmentation-based refinement module is applied. To correct bounding box drift and tightly encapsulate visible object pixels, the refinement algorithm leverages the simulator’s semantic segmentation output. It is assumed that each object instance is assigned a specified unique color for segmentation. Given a set of bounding box labels, the refinement process executes as follows:

- 1) **Region of Interest (ROI) Extraction:** For every object bounding box captured in the initial, API-generated bounding box, the width and height dimensions are multiplied by a pre-defined scalar search factor to create an expanded ROI. This padding is to ensure that at least a portion of the target object’s pixels are captured within the ROI, even in the presence of severe bounding box offset due to changes in the camera field-of-view.
- 2) **Instance Identification:** The algorithm isolates all non-background pixels within the ROI and computes the mode of the color distribution to determine the most prevalent pixel color. This dominant color corresponds to the unique instance ID given to the target object in the ROI, which allows us to set the class label of the object.
- 3) **Global Masking and Re-bounding:** There may be cases where an object’s segmentation is broken due to occlusions. In these instances, rather than restricting the corrected bounding box to the local ROI, the algorithm searches the entire segmentation image for the unique instance ID (i.e., the color segments) of the object. After identifying all colored segments of the instance, a bounding box is tightly fitted around the extremes of the visible pixels belonging to the object. Additionally, a fill-ratio filter is applied to ensure the object meets a minimum visibility threshold before the final label is accepted. For example, if the object is mostly hidden behind a tree, we do not wish to capture this label.

This procedure is used over a global segmentation search for multiple reasons. The initial API query serves as a semantic filter, effectively ignoring irrelevant environmental geometry and narrows the search space to target classes. Furthermore, using the API-generated box as a spatial prior provides a robust mechanism for instance disambiguation; by calculating the mode of the color distribution within a localized ROI, the algorithm correctly identifies the target instance ID even in scenes with high object density or significant overlap. This ensures that the resulting labels are semantically accurate and computationally efficient to generate. This is shown in Fig. 3.

IV. EXPERIMENT SETUP

The 3D mesh generation workflow is performed on a workstation running Ubuntu 22.04.5 LTS consisting of an AMD Ryzen Threadripper PRO 7955WX 16-Core CPU and two NVIDIA GeForce RTX 5090 GPUs. The meshes were imported into the AirSim suburban neighborhood environment, along with the final synthetic data collection and annotation, with a Windows desktop computer with an AMD Ryzen 9700x CPU and one NVIDIA GeForce RTX 5070Ti GPU.

To curate a comprehensive dataset for the synthetic data generation workflow, images of common suburban objects were systematically aggregated from multiple established benchmark datasets. A realistic simulated neighborhood requires a high-fidelity representation of diverse structural and dynamic assets, including residential buildings, passenger vehicles, and critical urban infrastructure such as stop signs and traffic lights. To

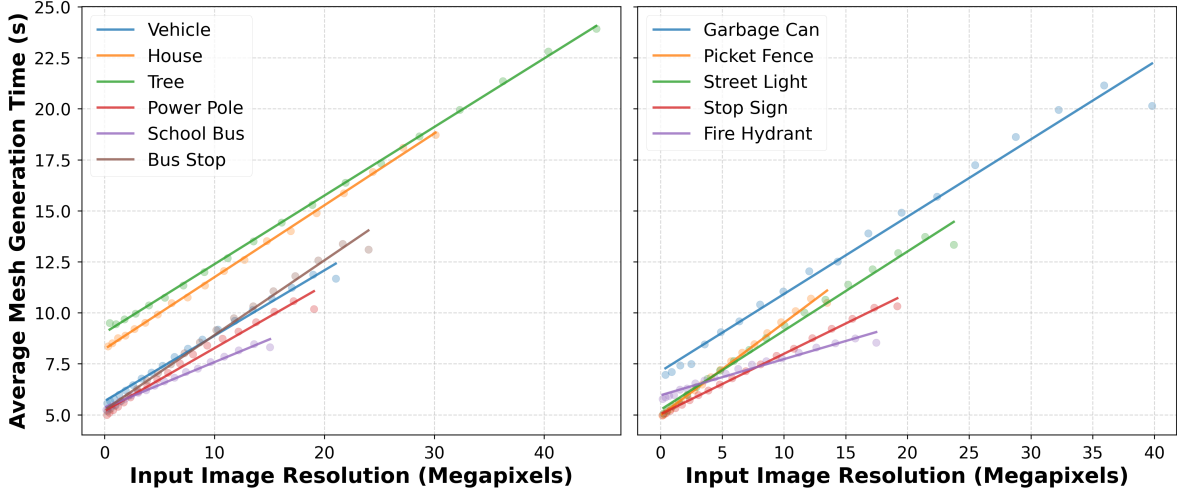


Fig. 6. Computational scaling of mesh generation time relative to input image resolution. High resolution images, ranging from 15 to 40 MP, were selected of each object type. The image heights and widths were scaled using resolution factors ranging from 0.1 to 1.0. Data points represent the mean processing time over 10 independent trials for each resolution setting, with solid lines representing linear regression fits. The results show a linear correlation across diverse object classes as input image resolution increases. We average the runs of each resolution to account for minor fluctuations in hardware performance during execution.

achieve this requisite visual variance, high-resolution instances of vehicles were sourced from the Stanford Cars Dataset [43], leveraging its repository of 16,185 images across 196 fine-grained car classes to ensure diverse vehicular type, geometry, texture. For residential structures, family home images were extracted from the House Prices and Images - SoCal dataset [44]. To populate the environment with essential street-level elements, we utilized the Microsoft Common Objects in Context (MS COCO) dataset [45], which provides highly contextualized instances of common outdoor objects like stop signs, fire hydrants, and benches. Finally, to supplement the dataset with additional infrastructural variance and ensure robust multi-class coverage, we integrated targeted subsets from the Open Images dataset [46]. Its massive scale allows for the extraction of varied neighborhood objects guaranteeing that the generated 3D assets are diverse enough to generate different environments for our landing scenario.

The default SAM 3D weights available from Hugging Face are used. For the label refinement, we set a search factor of 1.2. Through experimentation, we found this to be an optimal scaling value for ROI expansion at altitudes $\leq 25m$, successfully capturing target pixels despite camera-induced spatial drift. To ensure high-quality training data, labels are subjected to a fill-ratio filter, where instances with less than 5% visibility are discarded to prevent the inclusion of overly noisy or ambiguous samples.

V. RESULTS AND DISCUSSION

A. Generated Suburban 3D Mesh Objects

The experiments demonstrate that the integration of generative AI boosts the scalability and efficiency of synthetic environment generation for UAS simulation. We analyzed the performance of the generative model by measuring the average

mesh generation time against varying input image resolutions. The image heights and widths were scaled down using resolution factors ranging from 0.1 to 1.0, and the average mesh generation time was reported over 10 trials per image. These plots are given in Figure 6. In this section, we refer to resolution as the MegaPixel (MP) value of the image.

The input data included diverse imagery of common sub-urban objects featured at various resolutions; for instance, the house image was captured at approximately 30 MP, whereas the tree image reached upwards of 40 MP. The data shows a linear correlation between resolution and processing time across all object categories. For complex assets such as houses and trees, the generation time scaled from approximately 8 to 9 seconds at low resolutions to approximately 24 seconds at 40 MP. In contrast, smaller urban infrastructure like fire hydrants and stop signs consistently required less than 15 seconds, even at peak input resolution. This shows that generative AI can handle high-fidelity source imagery without the exponential computational overhead typically associated with traditional photogrammetry or manual modeling workflows.

The throughput of the system was further validated by assessing the total generation time relative to the volume of data produced. The results (see Fig. 7) exhibit linear scaling, where mesh generation time on average increases proportionally to the dataset size. For a representative batch of 150 unique images (per object type), the generative pipeline completed the reconstruction of complex objects like cars and houses in approximately 1,550 seconds, while simpler assets like stop signs were completed in approximately 1,000 seconds. This capability allows for the creation of hundreds of unique, simulation-ready 3D assets in a time frame that is orders of magnitude faster than manual 3D mesh creation workflows, which often require hours for a single high-quality asset.

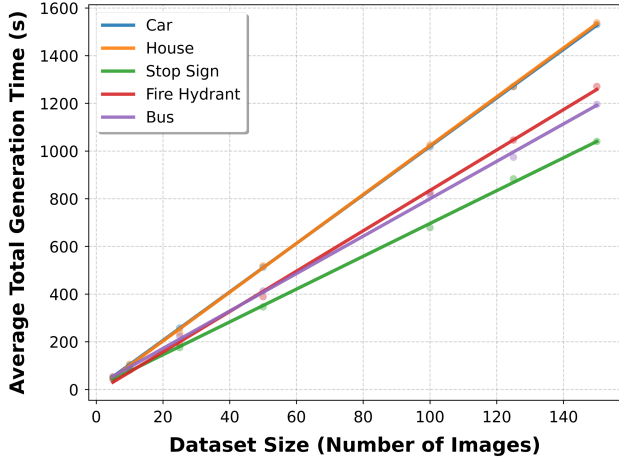


Fig. 7. Computational scaling of mesh generation time relative to input dataset size. The datasets contain large quantities of object images at various resolutions for each class. Data points represent the mean processing time over 3 runs for each dataset size, with solid lines representing linear regression fits. We average the runs of each dataset size to account for minor fluctuations in hardware performance during execution. The results show a linear correlation in average mesh generation time as the number of images increases across all object classes.

Qualitative assessment of the generated outputs reveals that the SAM 3D pipeline maintains geometric and textural fidelity. The 3D meshes accurately capture the structural properties of the 2D source images, preserving features such as the lettering on school buses and the specific geometry of power poles; however, the reconstruction produced holes in the mesh structure of larger houses, while relatively smaller objects (e.g. cars) had closed reconstructed geometry. Furthermore, we noted that lowering the input image resolution reduced the texture and material detail of the reconstructed geometry. When these generative assets are integrated into the Unreal Engine 5 (UE5) environment, they provide a diverse and realistic visual landscape for autonomous flight (see Figures 4 and 5). The transition from 2D imagery to 3D objects enables the rapid generation of diverse simulation scenarios, ensuring that the resulting synthetic datasets provide the variety necessary for training, testing, and validating robust on-board perception models.

The results indicate that the generative AI-driven reconstruction provides a viable path for massive-scale synthetic data synthesis. By bypassing the manual modeling bottleneck, we can generate thousands of precisely labeled frames within highly varied virtual worlds. These findings suggest that generative AI not only reduces the labor-intensive nature of simulation design, but also ensures the high-fidelity environmental diversity required for the safety-critical validation of UAS in autonomous package delivery operations.

B. AirSim Environment and Image Data Generation

The scalability and precision of the data collection and automatic annotation pipeline were further evaluated within the synthesized Unreal Engine 5 environment. We first measured

the throughput of the end-to-end data collection process, where a simulated UAS traversed the environment while capturing image data from a downward-facing camera. The results indicate a highly efficient, linear scaling of collection time relative to the number of images produced. Specifically, the system successfully captured 100 images in 82.21 seconds and scaled to 10,000 images in 8,385.31 seconds. This results in a production rate of approximately 0.84 seconds per frame, including the overhead for real-time rendering, physics simulation, and raw label generation. When compared to the time required for manual data labeling, ranging from several seconds to minutes per frame depending on scene complexity, this automated approach offers an increase in dataset generation speed without sacrificing output volume.

We generate segmentation masks for several key object classes such as people, vehicles, and trash bins, which represent potential obstacles for safe autonomous landing in neighborhood settings. To ensure the quality of these automatically generated labels, we implemented a refinement layer on top of the native Cosys-AirSim annotation API. The performance of this refinement was quantified using three metrics: label deletion, object pixel density, and object pixel containment. The label deletion metric is defined as the percentage of original unrefined labels that failed to find a corresponding match in the refined set (based on a non-zero Intersection over Union). The object pixel density is defined as the ratio of object-specific pixels (identified via color-masking within the box) to the total pixel area of the bounding box ($Area_{object}/Area_{bbox}$). Higher values indicate an improvement to the label, since refined bounding boxes will encapsulate less background and increase the ratio value. The object pixel containment is defined as the percentage of the total instance pixels present in the full segmentation mask that were successfully captured within the boundaries of the bounding box. This measures the extent to which the refinement process avoids cropping out parts of the object. An increase in this value indicates a tighter fit around the target.

Our analysis shows that the proposed refinement process reduces background captured in the bounding boxes, leading to higher-quality training data. Table I gives the resulting metrics described above of the label refinement algorithm for a sample size of 10,000 images collected from the environment (see Fig. 5) with a simulated UAS camera. For classes such as “people,” the object pixel density improved from an unrefined 0.24 to a refined 0.48, doubling the “tightness” of the label. Similar improvements were observed in vehicle classes, where “car” density increased from 0.55 to 0.74. Despite this tightening, the containment metrics remained near-perfect, consistently reaching 0.99 to 1.00. This indicates that the refined labels are not only tighter, but also accurately encompass the entire visible geometry of the target objects without truncation. The robustness of the refinement algorithm is further evidenced by the minimal loss of object instances. Across all classes, the percentage of “lost” objects remained below 3%, with the highest loss occurring in the “car” class (2.7%) due to the removal

TABLE I
LABEL REFINEMENT METRICS BY OBJECT CLASS FOR 10,000 COLLECTED IMAGES

Object Class Name	Label Deletion (%) (Count)	Object Pixel Density (Unref $\rightarrow$ Ref)	Object Pixel Containment (Unref $\rightarrow$ Ref)
people	2.1% (1084)	0.24 $\rightarrow$ 0.48	0.93 $\rightarrow$ 0.99
car	2.7% (950)	0.55 $\rightarrow$ 0.74	0.99 $\rightarrow$ 1.00
truck	1.8% (137)	0.54 $\rightarrow$ 0.79	0.98 $\rightarrow$ 1.00
bus	1.5% (63)	0.47 $\rightarrow$ 0.64	0.97 $\rightarrow$ 1.00
trash bin	0.4% (133)	0.60 $\rightarrow$ 0.80	1.00 $\rightarrow$ 0.99
landing pad	0.1% (6)	0.74 $\rightarrow$ 0.78	1.00 $\rightarrow$ 1.00

of heavily occluded or distant objects that would otherwise introduce noise into a training dataset. Smaller, high-priority objects like “landing pads” showed virtually no loss (0.1%) and perfect containment scores. As shown in Figure 3, the refinement step transforms loose, approximate bounding boxes into precise annotations that align perfectly with the object silhouettes. This precision, combined with the rapid generation rate, demonstrates that our generative AI-driven workflow and automatic annotation pipeline provide a scalable and reliable framework for generating synthetic datasets tailored to deep learning-enabled UAS package delivery.

C. Limitations

While the proposed refinement strategy successfully bounds visible object fragments, the reliance on a static padding scale presents a clear limitation. For instance, small objects viewed from very high altitudes possess significantly lower pixel densities, requiring dynamically smaller padding scales compared to low-altitude observations. Future iterations of the pipeline will investigate dynamic padding parameterized by altitude and pixel density, as well as centroid-based initialization, to further mitigate bounding box drift prior to the refinement search. Additionally, we assume that each object is assigned to a unique segmentation color. This limits the number of objects that we can label depending on the availability of colors to assign.

Regarding the generative AI component, several constraints were identified that impact the broad applicability of the workflow. The generative 3D reconstruction pipeline is currently optimized for single-view input imagery, which result in approximations on the facets of an object not visible to the original camera. While the current fidelity is sufficient for the majority of our simulated flight perspectives, the transition to multi-view generative models could further enhance the structural completeness of complex assets. Furthermore, the quality of synthesized textures remains dependent on the lighting and clarity of the source photographs, suggesting that a pre-processing stage for lighting normalization could improve visual consistency within the Unreal Engine environment. There are also inherent limitations in the mesh topology produced by the generative model. AI-generated meshes often feature dense, unoptimized triangulation that can impact simulation performance when thousands of unique objects are rendered simultaneously. While modern engines like Unreal Engine 5

mitigate this through specialized geometry pipelines, future work can explore automated mesh decimation to ensure compatibility with lower-performance simulation hardware. Finally, the reconstruction accuracy remains sensitive to the initial 2D segmentation masks, meaning that any errors in the underlying generative segmentation model can propagate into the final 3D geometry.

VI. CONCLUSION

This study demonstrated a scalable workflow for synthetic flight data generation by leveraging generative AI 3D mesh reconstruction and automated annotation refinement. The results indicate that the use of generative 3D reconstruction models mitigates the manual modeling bottleneck, allowing for the rapid population of diverse suburban environments with simulation-ready assets in a matter of minutes. Our data collection pipeline proved efficient, maintaining linear scalability and producing high-fidelity frames with precise annotations at a rate of approximately 0.84 seconds per image. By bypassing the labor-intensive requirements of manual modeling and labeling, this framework enables the rapid iteration and validation of UAS perception models in realistic simulation environments. Future work will focus on the development of dynamic refinement parameters to handle varying altitudes and the exploration of multi-view generative techniques to further increase the geometric fidelity of synthesized virtual worlds.

REFERENCES

- [1] Epic Games, “Unreal Engine 5.5 Documentation,” 2024, (Accessed: 06 April 2026). [Online]. Available: <https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5-5-documentation>
- [2] W. Jansen *et al.*, “COSYS-AIRSIM: A Real-Time Simulation Framework Expanded for Complex Industrial Applications,” in *2023 Annual Modeling and Simulation Conference (ANNSIM)*, 2023, pp. 37–48.
- [3] J. Xu, C. X. Vu, A. Taye, A. Tabrizian, and P. Wei, “Small UAS Landing Site Detection with Deep Learning based Computer Vision,” in *AIAA AVIATION FORUM AND ASCEND 2025*. AIAA, 2025. [Online]. Available: <https://arc.aiaa.org/doi/abs/10.2514/6.2025-3522>
- [4] S. Abdollahzadeh, M. S. Allili, A. Boulmerka, and J.-F. Lapointe, “A Vision-Based Framework for Safe Landing Zone Mapping of UAVs in Dynamic Environments,” *IEEE Open Journal of the Computer Society*, vol. 7, no. 01, pp. 492–503, Jan. 2026. [Online]. Available: <https://doi.ieeecomputersociety.org/10.1109/OJCS.2026.3663268>
- [5] E. Miccio, P. Veneruso, R. Opromolla, G. Fasano, C. Tiana, and G. Gentile, “Vision-aided Sensing Pipeline with AI-based vertipoint detection for Precision Navigation in UAM Approach and Landing Scenarios,” *Aerospace Science and Technology*, vol. 168, p. 111180, 2026. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S127096382501243X>

- [6] Z. Zhao, J. Lee, Z. Li, C. H. Park, and P. Wei, "Vision-based Perception with Safety Awareness for UAS Autonomous Landing," in *AIAA SCITECH 2023 Forum*. AIAA, 2023. [Online]. Available: <https://arc.aiaa.org/doi/abs/10.2514/6.2023-0126>
- [7] C. X. Vu, M. Ghazanfari, K. Dong, A. Taye, A. Tabrizian, and P. Wei, "Transformer or CNN? Benchmarking Real-Time Detection Transformer and YOLOv8 for Small UAS Autonomous Landing," in *AIAA AVIATION FORUM AND ASCEND 2025*, 2025, p. 3521. [Online]. Available: <https://arc.aiaa.org/doi/abs/10.2514/6.2025-3521>
- [8] P. Veneruso, E. Miccio, R. Opromolla, G. Fasano, G. Gentile, and C. Tiana, "Enhanced Vision-Based Obstacle Sensing During UAM Approach and Landing Operations," in *2025 AIAA DATC/IEEE 44th Digital Avionics Systems Conference (DASC)*, 2025, pp. 1–10.
- [9] Federal Aviation Administration, "Package Delivery by Drone (Part 135)," 2023, accessed: 2026-04-10. [Online]. Available: https://www.faa.gov/uas/advanced_operations/package_delivery_drone
- [10] M. Mittal *et al.*, "Isaac Lab: A GPU-Accelerated Simulation Framework for Multi-Modal Robot Learning," in *arXiv preprint arXiv:2511.04831*, 2025. [Online]. Available: <https://arxiv.org/abs/2511.04831>
- [11] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154 vol.3.
- [12] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A Flexible Quadrotor Simulator," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. Tomlin, Eds., vol. 155. PMLR, 16–18 Nov 2021, pp. 1147–1157. [Online]. Available: <https://proceedings.mlr.press/v155/song21a.html>
- [13] X. Chen *et al.*, "SAM 3D: 3Dfy Anything in Images," 2025. [Online]. Available: <https://arxiv.org/abs/2511.16624>
- [14] N. Carion *et al.*, "SAM 3: Segment Anything with Concepts," 2025. [Online]. Available: <https://arxiv.org/abs/2511.16719>
- [15] F. Wang *et al.*, "Learning-Based Multi-View Stereo: A Survey," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 01, pp. 1–20, Jan. 5555. [Online]. Available: <https://doi.ieeecomputersociety.org/10.1109/TPAMI.2026.3654665>
- [16] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," in *ECCV*, 2020.
- [17] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D Gaussian Splatting for Real-Time Radiance Field Rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, July 2023. [Online]. Available: <https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/>
- [18] A. Sinha, A. Unmesh, Q. Huang, and K. Ramani, "SurfNet: Generating 3D Shape Surfaces Using Deep Residual Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [19] J. K. Pontes, C. Kong, S. Sridharan, S. Lucey, A. Eriksson, and C. Fookes, "Image2Mesh: A Learning Framework for Single Image 3D Reconstruction," 2017. [Online]. Available: <https://arxiv.org/abs/1711.10669>
- [20] J. Pan, X. Han, W. Chen, J. Tang, and K. Jia, "Deep Mesh Reconstruction From Single RGB Images via Topology Modification Networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.
- [21] S. Wu, C. Rupprecht, and A. Vedaldi, "Unsupervised Learning of Probably Symmetric Deformable 3D Objects From Images in the Wild," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [22] Y. Xu, T. Fan, Y. Yuan, and G. Singh, "Ladybird: Quasi-monte carlo sampling for deep implicit field based 3d reconstruction with symmetry," in *Computer Vision – European Conference on Computer Vision (ECCV) 2020*. Springer, Cham, 2020, pp. 248–263.
- [23] B. Guillard, E. Remelli, and P. Fua, "UCLID-Net: Single View Reconstruction in Object Space," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 3244–3253. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/21327ba33b3689e713cdf1641128004-Paper.pdf
- [24] J. Tang, X. Han, M. Tan, X. Tong, and K. Jia, "SkeletonNet: A Topology-Preserving Solution for Learning Mesh Reconstruction of Object Surfaces from RGB Images," 2021. [Online]. Available: <https://arxiv.org/abs/2008.05742>
- [25] Y. Wolf, A. Bracha, and R. Kimmel, "GS2Mesh: Surface Reconstruction from Gaussian Splatting via Novel Stereo Views," in *European Conference on Computer Vision (ECCV)*, 2024.
- [26] D. Tochilkin *et al.*, "TriposR: Fast 3D Object Reconstruction from a Single Image," *arXiv preprint arXiv:2403.02151*, 2024.
- [27] M. Boss, Z. Huang, A. Vasishtha, and V. Jampani, "SF3D: Stable Fast 3D Mesh Reconstruction with UV-unwrapping and Illumination Disentanglement," 2024. [Online]. Available: <https://arxiv.org/abs/2408.00653>
- [28] X. Yang *et al.*, "SAM 3D Body: Robust Full-Body Human Mesh Recovery," 2026. [Online]. Available: <https://arxiv.org/abs/2602.15989>
- [29] J. Xiang *et al.*, "Structured 3D Latents for Scalable and Versatile 3D Generation," 2024. [Online]. Available: <https://arxiv.org/abs/2412.01506>
- [30] —, "Native and Compact Structured Latents for 3D Generation," 2025. [Online]. Available: <https://arxiv.org/abs/2512.14692>
- [31] J. Yao, L. Mou, and Q. Li, "SAM 3D for 3D Object Reconstruction from Remote Sensing Images," 2025. [Online]. Available: <https://arxiv.org/abs/2512.22452>
- [32] A. Verma, N. D. Stafeno, C. Wendt, C. Aggarwal, A. Baserinia, and A. G. Rodriguez, *Using Game Engines as Simulation Training Platforms to Enable Autonomous Flight*. Aerospace Research Central, 2024. [Online]. Available: <https://arc.aiaa.org/doi/abs/10.2514/6.2024-4125>
- [33] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles," in *Field and Service Robotics*, 2017. [Online]. Available: <https://arxiv.org/abs/1705.05065>
- [34] IAMA Simulations, "ProjectAirSim," 2026, gitHub repository. Accessed: 2026-04-11. [Online]. Available: <https://github.com/iamaisim/ProjectAirSim/tree/main>
- [35] General Robotics, "AirGen : The evolution of AirSim," November 2024, accessed: Apr. 6, 2026. [Online]. Available: <https://www.generalrobotics.company/post/airgen>
- [36] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [37] B. Kiefer, D. Ott, and A. Zell, "Leveraging Synthetic Data in Object Detection on Unmanned Aerial Vehicles," in *2022 26th International Conference on Pattern Recognition (ICPR)*, 2022, pp. 3564–3571.
- [38] A. Barisic, F. Petric, and S. Bogdan, "Sim2Air - Synthetic Aerial Dataset for UAV Monitoring," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3757–3764, 2022.
- [39] J. Rüter, T. Maienschein, S. Schirmer, S. Schopferer, and C. Torens, "Filling the Gaps: Using Synthetic Low-Altitude Aerial Images to Increase Operational Design Domain Coverage," *Sensors*, vol. 24, no. 4, 2024. [Online]. Available: <https://www.mdpi.com/1424-8220/24/4/1144>
- [40] D. Ding, C. X. Vu, A. Taye, A. Tabrizian, P. Wei, and Z. Zhao, *Synthetic Data Generation for Computer Vision Based Autonomous Landing for Small UAS Package Delivery*. AIAA, 2025. [Online]. Available: <https://arc.aiaa.org/doi/abs/10.2514/6.2025-3713>
- [41] N. Zhou and T. Li, "Synthetic Dataset Generation Method for Object Detection," *International Journal of Computational Intelligence Systems*, vol. 18, no. 1, p. 81, 2025. [Online]. Available: <https://doi.org/10.1007/s44196-025-00817-4>
- [42] T. R. Lenhard, A. Weinmann, K. Franke, and T. Koch, "SynDroneVision: A Synthetic Dataset for Image-Based Drone Detection," in *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Los Alamitos, CA, USA: IEEE Computer Society, Mar. 2025, pp. 7637–7647. [Online]. Available: <https://doi.ieeecomputersociety.org/10.1109/WACV61041.2025.00742>
- [43] J. Krause, J. Deng, M. Stark, and L. Fei-Fei, "Collecting a Large-scale Dataset of Fine-grained Cars," 2013. [Online]. Available: <https://api.semanticscholar.org/CorpusID:16632981>
- [44] ted8080, "House Prices and Images - SoCal," Dataset available from <https://www.kaggle.com/datasets/ted8080/house-prices-and-images-socal>, 2019.
- [45] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 740–755.
- [46] A. Kuznetsova *et al.*, "The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale," *International Journal of Computer Vision*, vol. 128, no. 7, pp. 1956–1981, 2020.