International Journal of Computational and Electronics Aspects in Engineering
Volume 7, Issue 5, November 2026, pp 562 - 575
https://doi.org/10.26706/ijceae.7.5.20261003

Design and Implementation of a Preprocessing-Aware Multi-Input AI Framework for 3D Object Generation

Hussam Ahmed Rammo[1,*], Mays Q. Sedeeq[2], Razan Abdulhammed[3], Shamil Khudhur Ramadhan[4], Fahad Ahmed Shaban[5]



[1] Technical Engineering College for Computer and Artificial Intelligence / Mosul, Northern Technical University, Nineveh, Mosul, Iraq.

[2] Department of Cybersecurity Engineering, College of Computer Engineering and Artificial Intelligence, Northern Technical University (NTU), Mosul, Iraq.

[3] Artificial Research Center, Northern Technical University, Nineveh, Iraq.

[4] Department of Clinical Chemistry and Artificial Intelligence, High Institute of Health Technologies and Artificial Intelligence/Northern Technical University, Mosul, Iraq.

[5] Department of Computer Networks and Internet, College of Information Technology, Ninevah University, Mosul, Iraq.

***Correspondence:** hussam.rammo@ntu.edu.iq

Article – Peer Reviewed
Received: 5 August 2026
Accepted: 9 October 2026
Published: 1 November 2026

Copyright: © 2026 RAME Publishers
This is an open access article under the CC BY 4.0 International License.

https://creativecommons.org/licenses/by/4.0/

Cite this article: Hussam Ahmed Rammo, Mays Q. Sedeeq, Razan Abdulhammed, Shamil Khudhur Ramadhan, Fahad Ahmed Shaban, "Design and Implementation of a Preprocessing-Aware Multi-Input AI Framework for 3D Object Generation", *International Journal of Computational and Electronic Aspects in Engineering*, RAME Publishers, vol. 7, issue 5, pp. 562-575, 2026.
https://doi.org/10.26706/ijceae.7.5.20261003

Abstract: Although the technical challenges to create 3D content have been lowered by the application of artificial intelligence, practical challenges remain with heterogeneous input types, artefacts from background, computational demands, and integration of the system. This paper proposes a preprocessing-aware multi-input method for creating textured 3D objects from text, image and sketch input in a single application with a unified client server system. The text-to-3D workflow is implemented using FLUX.1-schnell to create an intermediate image, and Hunyuan3D-2.1 is used for the 3D shape and texture synthesis of images generated by text, photographs and sketches. The front end is designed using Flutter, and the backend is developed using Python FastAPI, which handles the processing of pre-processing, model inference, job monitoring and generation of GLB output, while heavy computation is performed remotely on an NVIDIA A100 80 GB GPU. The structure contains background removal, alpha thresholding, object cropping and centred square padding to minimize geometry from background. As a result of functional testing, the successful generation and in-application preview were confirmed for all three workflows. The end-to-end generation times were 80.72 s (sketch-to-3D), 103.89 s (image-to-3D), and ~105 s (text-to-3D) respectively. In the evaluated case, the ground plane artefact was observed when preprocessing was performed using background removal only. For the case in question, a qualitative pre-processing ablation indicated that the observed artefact on the ground plane can be removed with an object-only cleaning. The results show that the current generative models, pre-processing, and inference on cloud can be integrated within a unified 3D generation pipeline. The current assessment is still at prototype level due to lack of repeated-trial statistics, ground truth geometric standards and wider validation.

Keywords: 3D object generation; multi-input framework; text-to-3D; image-to-3D; sketch-to-3D; input preprocessing

1. Introduction

Use of Three-dimensional (3D) content is growing in the fields of education, product visualisation, architectural design, engineering, game development, virtual and Augmented reality and interactive digital media. However, traditional 3D modelling still faces difficulties for non-specialists, as they need to be proficient in mesh building, topology, UV mapping, material assignment, and utilize professional software. These requirements can restrict the ability of students, teachers, designers, and small development teams from creating rapid 3D assets.

The rise of new trends in generative artificial intelligence has led to the development of alternative methods that enable the generation of 3D content from textual information or visual input instead of manual modelling. The earliest approaches to text-to-3D used point cloud generation or expensive optimization, while the more recent methods have used learned implicit, structured and feed-forward representations to produce increasingly useful 3D assets[1],[2]. Single-image reconstruction methods have also enhanced the efficiency of 3D recovery, and recent systems like Hunyuan3D-2.1 [2] include shape generation with physically based rendering (PBR) texture synthesis.

However, strong generative models alone do not address all practical application requirements. Heterogeneous data input, pre-processing, model orchestration, computational resources, system job monitoring, output delivery and interactive visualisation will need to be integrated into end-user systems. The image conditioned reconstruction is also sensitive to the irrelevant image regions. In the development of the present system, it is sometimes necessary to reconstruct a floor, road, shadow and surrounding background as unwanted planar geometry under the target object. This observation inspired explicit object-isolation preprocessing to be incorporated into the generation pipeline.

This study proposes a pre-processing-aware multi-input framework able to handle text, image and sketch inputs in the same client-server application to overcome such problems. The photos, sketches and FLUX.1-generated intermediate images are used by Hunyuan3D-2.1 [2] to create textured 3D assets, while the text prompts are converted to intermediate images with FLUX.1-schnell [3].

This work is not about proposing a new generative foundation model but is about system integration. The study compares the performance of the fully fledged prototype in terms of the observed end-to-end latency and investigates if there is any possibility to integrate the three workflows – text, image and sketch – within a shared 3D generation pipeline on the one hand, and to see if background-induced geometry artefacts can be mitigated by preprocessing on the other. The framework is thus functionally validated, the latency is measured, and a qualitative preprocessing ablation is conducted, while a more extensive quantitative ablation is proposed as a future task.

The objectives of this study are to:

- Design and implement a unified multi-input framework for generating textured 3D objects from text, image, and sketch inputs.
- Integrate preprocessing, generative AI models, and remote GPU execution within a client-server pipeline that produces standard GLB outputs.
- Evaluate the implemented framework through functional testing, observed end-to-end latency measurements, and a qualitative preprocessing ablation.

The principal contributions of this work are summarized as follows:

- A unified multi-input 3D generation framework that integrates text-to-3D, image-to-3D and sketch-to-3D pipelines into a single client-server application and generates textured GLB assets.
- A pre-processing based image-conditioning technique that removes background, uses alpha thresholding, cropping objects, and centred padding to reduce the geometry artefacts that are caused by the background.
- A modular text-to-3D pipeline that integrates FLUX.1-schnell text-to-image generation with Hunyuan3D-2.1 image-conditioned 3D generation, bringing all types of input to a unified 3D synthesis stage.
- Implementation and application level evaluation with NVIDIA A100 80 GB GPU on cloud, functional, qualitative processing-level ablation, and recorded end-to-end generation times for all three workflows.

The remainder of this paper is organised as follows. The related works on text-to-3D, image-to-3D, and sketch-to-3D generation are summarized in section 2 and the proposed work is placed in the context of the previous studies. The system

architecture, generative model, preprocessing method, deployment environment and experimental setup are described in Section 3. The experimental results are presented in Section 4 with functional validation, latency measures and the ablation study of the preprocessing. The implications and implications for practice are discussed in section 5. The paper ends with Section 6 on the limitations of the study and directions for further research work in Section 7.

2. Related Work

2.1 Text-Conditioned 3D Generation

The use of text to 3D generation has evolved from coarse representations, optimisation-based approaches, to more direct and expressive generative pipelines. Point-E [1] could generate 3D point clouds from textual prompts and score distillation sampling was proposed by DreamFusion [4] to optimize the 3D representation with a pre-trained 2D text-to-image diffusion model. Later Magic3D [5] used a coarse-to-fine approach to enhance the resolution and reduce the optimisation time. n later works, ProlificDreamer further enhanced text-to-3D generation through variational score distillation, aiming to improve the fidelity and diversity of generated 3D content[6]. Multi-view diffusion methods such as MVDream [7] further explored consistent multi-view image generation as a prior for 3D generation. Conditional generation was further advanced by Shap-E[8], which learns conditional implicit 3D representations from text and image inputs. GaussianDreamer [9] subsequently explored text-to-3D generation by bridging 2D and 3D diffusion models through 3D Gaussian splatting. These techniques have made significant advances but mostly they are geared towards the generative model itself and not towards embedding multiple types of user input into a unified application process. More recently, the algorithms and difficulties of text-to-3D shape generation have also been explored, giving a wider insight into the possibilities and limitations of the existing algorithms[10]. A wider study of 3D content generation also outlines the current methods and categorizes them to illustrate both the abundance of methods and the variety of representations available to create 3D assets. [11].

2.2 Image-Conditioned Reconstruction and High-Fidelity Asset Generation

Another development that has been achieved in single-image 3D reconstruction is the efficient feed-forward reconstruction. To address the challenge of efficient single-image 3D reconstruction, novel-view synthesis from a single image was demonstrated in Zero-1-to-3 [12] and transformer-based and multi-view reconstruction approaches such as LRM [13] and InstantMesh [14] were proposed. Wonder3D [15] further demonstrated single-image to 3D generation using cross-domain diffusion and multi-view normal and colour representations. TripoSR [16] also reaffirmed the importance of a single-image reconstruction for real-world inference. More recent studies have also explored the application of single image 3D reconstruction to the generation of plant models; this work has shown that the image-based reconstruction is relevant to special categories of object and real-world modelling tasks[17]. In addition, few recent works have explored retrieval based methods of dense 3D point-cloud reconstruction from a single image, especially in the presence of complex background scenes. These strategies emphasize the significance of the proper understanding of images and background handling in image-conditioned 3D reconstruction[18]. Another application that relies on reliable object segmentation is image-conditioned 3D reconstruction. For 3D generation, benchmarking of unsupervised object segmentation in real-world images [19] indicates that separating foreground objects from complex backgrounds is a challenge. Recently, there have been efforts to generate 3D objects from a single image by incorporating textual instructions for the generation of 3D objects [new reference] to better control the 3D asset that is created [20]. There have also been approaches in image-based deep learning that attempt to predict age and gender with explainable multi-task Swin Transformer networks from dorsal hand images [21], which shows that transformer-based models can be used for visual feature analysis [21].Later works, such as TRELLIS [22] and Hunyuan3D-2.0 [23] use richer latent representations and better geometry-appearance modelling. In this work, the core 3D generator is the Hunyuan3D-2.1[2] which separates the process of generating shapes and PBR textures, and serves as a common generation stage for photographs, sketches, and text-generated images.

2.3 Sketch-Based 3D Generation

Modelling with sketches is an alternative to the use of traditional 3D modelling, and enables users to define the structure of an object in an intuitive manner using only a few sketches. For example, Deep3DSketch+ [24] is able to recover 3D geometry from a single sketch drawn freehandly, based on the learnt structural priors and the sketch enhancement. However, in general sketch-oriented approaches are realized by dedicated reconstruction systems. The proposed framework treats sketches as visual signals, and processes them, flowing through the same generation stage (Hunyuan3D-

2.1) as the photographs, which avoids the need for a separate sketch-specific 3D model and shares the textured-output pipeline.

2.4 Text-to-Image as an Intermediate Generation Stage

The proposed text-to-3D pipeline takes the approach to cascading the process, converting text into an image first, and then 3D synthesizing. The architectures of rectified-flow transformer have been found to be good in generating high-resolution images[25]. Therefore, FLUX.1-schnell [3] is applied to create an intermediate image with a resolution of 512 × 512, which is then processed by Hunyuan3D-2.1[2]. In contrast to direct text-to-3D methods like DreamFusion[4], Magic3D [5] and Shap-E [8] this design preserves the modularity and replaceability of the text-to-image component and the image-to-3D component. Recently survey paper on diffusion based methods for 3D generation has been presented, which gives a wider picture of the generative approaches relevant to the current 3D synthesis pipeline[26]. The main drawback is that an error or content of the intermediate image can make its way into the final 3D reconstruction.

2.5 Positioning of the Present Work

The current state of the art has made significant advances in text-conditioned generation, single-image reconstruction and sketch-based modelling. However, most studies concentrate on a particular generative architecture or input pathway. Less has been written about application-level systems, which include both text and image inputs, as well as sketch input, and integrate them with application-specific pre-processing, remote execution on GPUs, standardisation of 3D output, and an interactive preview.

The proposed work is thus centered on the system-level integration and not the development of a new foundation model. The underpinning generative capabilities are offered by FLUX.1-schnell and Hunyuan3D-2.1, but it is the integration of these models with preprocessing, cloud execution, job management, GLB output, and a unified user facing workflow that is provided by this study. Table 1 provides a summary of some representative approaches and places the proposed framework into the context of research focus and scope of application.

Table 1. Comparison between representative 3D generation approaches and the proposed framework

Related work	Focus	Limitation	Relation to the proposed framework
Point-E [1]	Text-to-3D point-cloud generation.	Produces point-cloud representations that require additional processing for direct use as conventional surface meshes	The proposed framework produces textured GLB assets that can be previewed directly within the application.
DreamFusion [4]	Text-to-3D generation using score distillation sampling.	Requires computationally expensive per-prompt optimisation.	The proposed framework shifts computationally intensive inference to a remote cloud GPU within an application-level workflow.
Magic3D [5]	High-resolution text-to-3D generation using a coarse-to-fine optimisation strategy.	Primarily designed for text-conditioned 3D generation.	The proposed framework supports text, image, and sketch inputs within a common application workflow.
Shap-E [8]	Conditional 3D generation from text or image inputs using implicit functions.	Earlier-generation output fidelity relative to more recent 3D generative models.	The proposed framework adds application-level integration, preprocessing, remote execution, and in-application preview.
TripoSR [16]	Fast single-image 3D reconstruction.	Primarily focuses on single-image reconstruction rather than a unified multi-input application workflow.	The proposed framework combines text, image, and sketch workflows and uses a shared textured 3D generation stage.

TRELLIS [22]	Text- and image-conditioned 3D generation using structured latent representations.	Primarily presented as a research-oriented generative system rather than an end-user application workflow.	The proposed framework emphasizes practical accessibility through a unified interface, remote execution, and direct preview.
Hunyuan3D-2.1 [2]	High-fidelity 3D shape generation with PBR texture synthesis.	Requires substantial GPU memory and computational resources.	Used as the common 3D generation stage and executed remotely so that the client does not require a high-end local GPU.
Proposed system	Text-, image-, and sketch-to-3D generation within a unified application, producing textured GLB assets.	Depends on the availability and stability of the remote backend and network connection.	Integrates three input types with preprocessing, cloud-based execution, standardised GLB output, and in-application preview.

2.6 Research Gap

Although significant progress has been made, there are still three system-level gaps that stand out in this context: (i) Text, image, and sketch inputs are still treated separately; (ii) There is still limited application-level treatment of background noise-induced geometry artefacts from preprocessing; and (iii) High-memory GPUs and research-oriented execution environments are required. The proposed framework aims to fill these gaps, by combining the three input types, adding explicit object-isolation preprocessing, moving inference to the remote gpu hardware and generating standard GLB assets for in-application preview and export. Consequently, the contribution is no new generative foundation model, but an application framework, that is designed to be used in a pre-processing-aware way.

3. Materials and Methods

3.1 Overall Architecture

The proposed system follows a two-tier client-server architecture. The client is a Flutter application built for Windows, Android, and iOS, which handles user input, displays the generation settings, tracks the generation process and displays the generated 3D asset. The computational pipeline and model execution are handled by the server implemented in Python, using FastAPI. The communication between both tiers is carried out via a versioned REST interface with the use of JSON, the upload of images is done as multipart form data, and the generation of GLB files are served as static output to the client. This separation means all the GPU-dependent operations occur outside of the network, meaning that the client doesn't need to have any local machine-learning dependencies or high-end graphics hardware. This architecture then decreases the amount of processing work needed on the client, but increases reliance on the remote backend.

Figure 1 illustrates the overall system architecture, including the supported input paths, generation stages, and cloud-based execution layer.

3.2 Client Application

Client application is written in Dart using the Flutter framework [27]. Application state is managed using the provider package, HTTP communication is handled through dio, and generated 3D assets are rendered Generated 3D assets are rendered using model_viewer_plus, based on the <model-viewer> platform for interactive 3D and AR visualisation[28]. The viewer renders glTF and GLB assets directly, and presents them as augmented reality on compatible devices. Because of this, GLB was chosen as the main output format, because it is a standard format and can be compactly represented for in-application visualisation and reuse in compatible 3D environments.

On the main screen are three explicit workflows: text-to-3D, image-to-3D, and sketch-to-3D, which are presented on the interface. The application interface is shown in figure 2, which includes the main application screen and the sketch-to-3D settings interface.

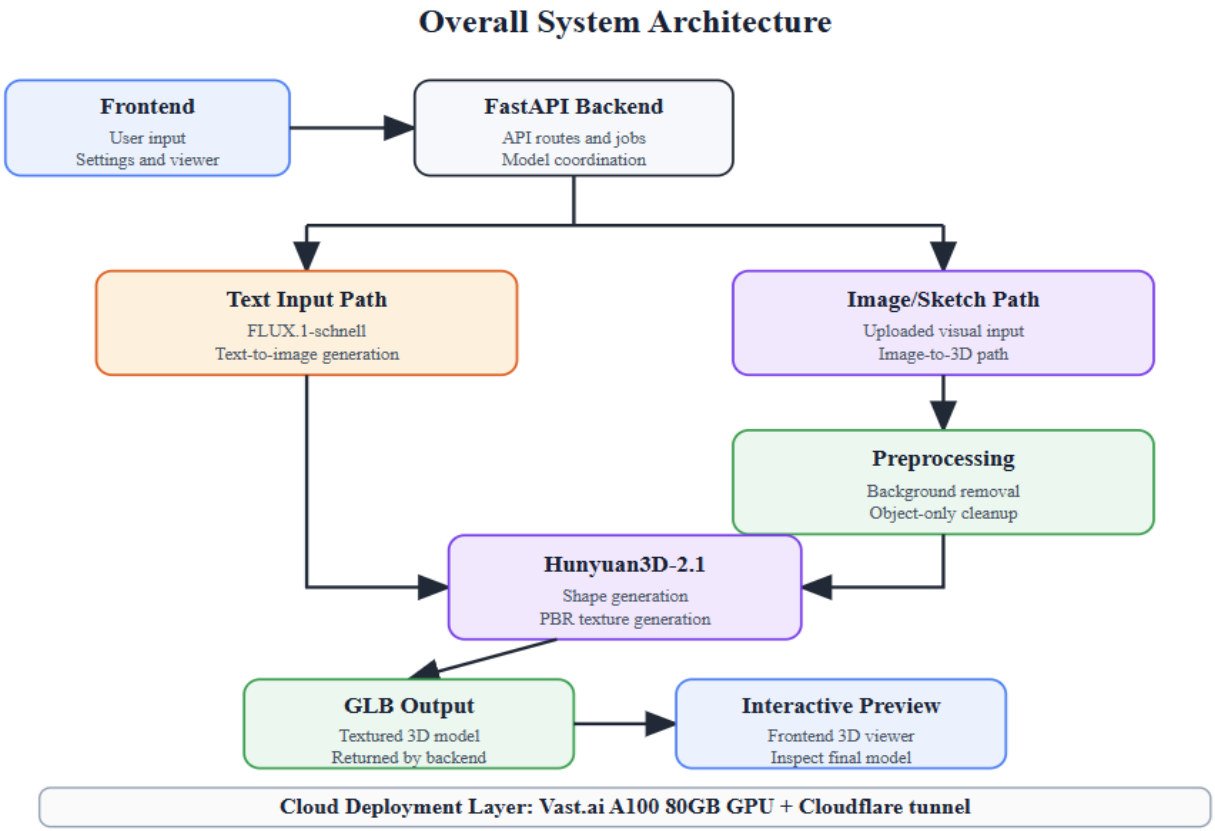


Figure 1. Overall architecture of the proposed system

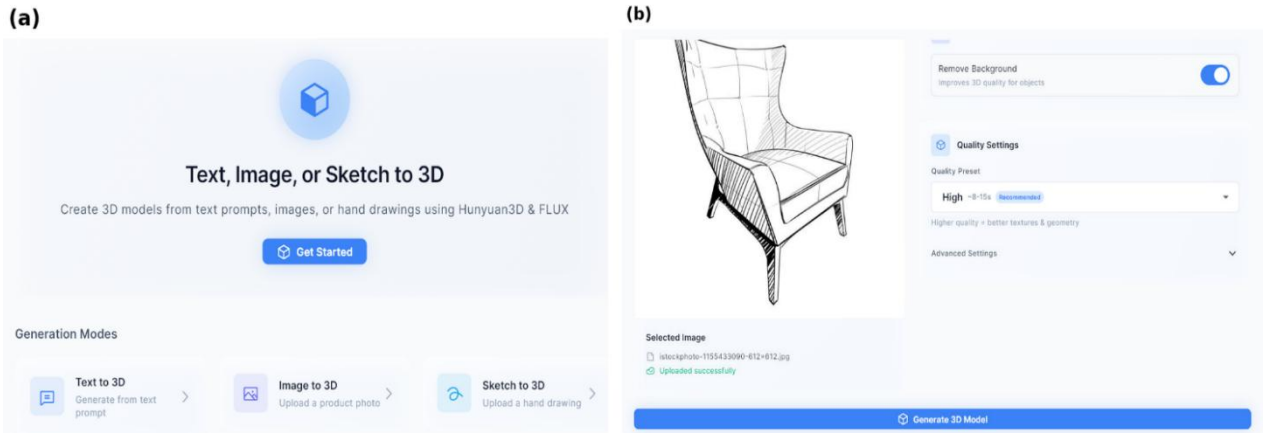


Figure 2. User interface of the proposed 3D generation application: (a) main application interface; (b) sketch-to-3D settings screen

There are four quality presets—fast, balanced, high and ultra—with high chosen as the default. A seed value, which is provided by the advance setting and is used to enhance the repeatability with the same experimental configuration, is available. There are 2 options for pre-processing available: removing background and cleaning up only objects – detailed in Section 3.6. Images that are uploaded must be PNG, JPEG or WebP format, no larger than 10 MB and a long edge with a resolution of no more than about 512 pixels is recommended to ensure consistency with the resolution used by the text-to-image stage.

3.3 Backend Service and Job Model

The backend is built using FastAPI framework [29] and it provides versioned endpoints on /api/v1 for image upload, preprocessing, generation, for image upload, preprocessing, generation, status monitoring, and result retrieval. To check backend availability before submitting a generation request additional /health and /ping endpoints are supplied. The

asynchronous approach was used for generation because individual requests can take over 1 minute to process on the GPU, and a continually open HTTP connection is undesirable when accessing the service over a tunnelled network connection. On submitting a generation request, the back end validates this request, provides a unique job identifier, schedules the processing and returns control to the client without delay.

The client then checks the status endpoint every two seconds. The process of each job goes through a series of states: queued, generating_image (if applicable), generating_3d, completed, or failed. A rough progress indicator is also kept in the UI. The job state is kept in memory, which works well for the one-instance research prototype, but does not support persistence or scalability, as described in Section 7.

Figure 3 illustrates the backend execution process. Figure 3(a) shows service startup and model loading, whereas Figure 3(b) shows diffusion sampling and volume decoding during 3D generation.

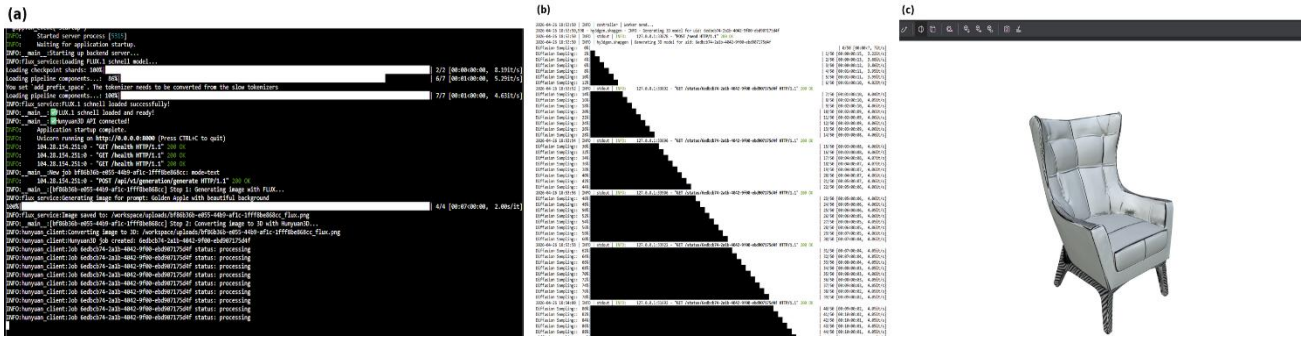


Figure 3. Backend execution of the proposed system: (a) service startup and model loading; (b) diffusion sampling and volume decoding during 3D generation

3.4 Text-to-Image Stage

FLUX.1-schnell [3] is used to process the user prompt for text input via the Diffusers FluxPipeline. Weights are loaded in bfloat16 while sequential CPU offloading is enabled to minimize peak GPU-memory usage and to facilitate the text and 3D generation parts being kept on the same device.

The text-to-image stage has four sampling steps, 512 × 512 image size, and a maximum length of text sequence of 256 tokens. The guidance value is to be set to zero, as is the case for the distilled FLUX.1-schnell configuration. The image created is saved to disk and fed directly into the image-to-3D stage. Therefore, in image synthesis, the resulting text to 3D follows the same pipeline from the 3D input, which is the downstream pipeline.

3.5 Image-to-3D Stage

Geometry and texture synthesis are done with Hunyuan3D-2.1 [2] as an independent API service on the same remote machine. This service is contacted by a lightweight client by the FastAPI backend. The image is encoded in Base64 and sent to the generation service that returns a unique job identifier. The backend will then query the service repeatedly until it has been processed and the resulting GLB asset will be returned as a Base64 payload, then stored in the output directory.

All requests can generate texture. This means that a generated mesh can include physically based rendering (PBR) material maps and not just vertex colours. The Hunyuan3D-2.1 service is not directly imported into the main FastAPI application, but is executed as a separate process. This separates its internal components for mesh processing and its native dependencies from its client facing API service. The cloud machine doesn't have Blender Python installed, so a compatibility shim is used and the OBJ-to-GLB conversion is passed to trimesh. Process isolation also ensures that the 3D conversion stage doesn't terminate the main backend service if there is a failure.

3.6 Input Preprocessing

Initial experiments using images showed that pictures with a floor, a road, cast shadows or a large background area might lead to a planar geometry under the reconstructed object. This behaviour implies that when separation between object and background is not complete, irrelevant parts of an image could be perceived as belonging to the object. To isualiz this effect, an explicit pre-processing step was added prior to feeding the image into the 3D generation model.

Let \(I\) represent the input image, and its representation in RGBA be

$$(R, G, B, A),$$

where A denotes the alpha channel. Background removal is done with rembg library, resulting in an image with an alpha channel—any pixel found to be background is given transparency. The resulting image is an image with an alpha channel – the background pixels that were detected are set to transparency. If rembg is not available, then a simple default that removes near-white background pixels is applied to the image by the backend. It is a fall-back function and used only for easy studio-type images.

With object-only cleanup, the object alpha channel contains an extra hard threshold:

$$A'(x,y) = \begin{cases} 255, & A(x,y) \geq \tau, \\ 0, & A(x,y) < \tau, \end{cases} \quad (1)$$

where the threshold is fixed at

$$\tau = 96$$

on the 0–255 alpha range. This operation suppresses low-alpha residual regions that may remain around shadows and soft object boundaries after background segmentation.

The remaining non-zero alpha region is then cropped. Let

$$B = (x_{\min}, y_{\min}, x_{\max}, y_{\max})$$

denote the bounding box of the non-zero region of A', with width w and height h. A margin is computed as

$$m = \max(8, \lfloor \rho \cdot \max(w, h) \rfloor), \quad (2)$$

where ρ is the padding ratio. The crop rectangle is expanded by m pixels on each side and constrained to the image boundaries.

The cropped object is then centred on a transparent square canvas with side length

$$s = \max(w_c, h_c), \quad (3)$$

where w_c and h_c denote the cropped width and height after the margin has been applied. Using a square canvas before model input reduces anisotropic distortion caused by directly resizing a non-square crop to the square resolution expected by the geometry model.

Two preprocessing configurations are available. The background removal configuration disables Equation (1) and applies the cropping and square-padding operations of Equations (2) and (3) with

$$\rho = 0.16.$$

The object-only cleanup configuration applies all three operations with

$$\tau = 96$$

and

$$\rho = 0.12.$$

The qualitative effects of these two configurations are evaluated in Section 4.3. The no-preprocessing condition is retained only as a historical baseline from early system development and was not re-run under matched experimental conditions.

3.7 Deployment Environment

The backend was deployed on a Vast.ai cloud instance equipped with an NVIDIA A100 80 GB PCIe GPU. Remote cloud execution has been adopted in prior deep-learning systems to offload computationally intensive inference from local devices[30]. The environment consisted of PyTorch 2.5.1, CUDA 12.4, and about 150 GB of storage (most of which was needed for model weights). Hunyuan3D-2.1 requires Python 3.10, and to prevent dependency issues with the host image, it was installed in a separate image.To minimize the fragmentation of the GPU memory, the memory allocator option to enable expandable segments was used.

The FastAPI service runs on port 8000 and the Hunyuan3D-2.1 service runs locally on port 8081. The FastAPI service is the only service that is made visible externally. In testing, public access was made possible by a Cloudflare Quick Tunnel which generates a temporary hostname that can be set up in the client settings.

Because Cloud flare Quick Tunnels are not intended for long-running requests, occasional connection interruptions were observed during lengthy textured-generation tasks. This deployment constraint further motivated the asynchronous job architecture and periodic polling mechanism described in Section 3.3.

3.8 Experimental Design and Evaluation Protocol

The primary quantitative measure used in this study is end-to-end wall-clock latency, defined as the elapsed time between submission of a generation request by the client and availability of the completed GLB asset for download. This measurement includes preprocessing, generative inference, texture synthesis, file writing, and job-status polling. It excludes the initial upload of the source image and the one-time model-loading cost incurred during backend startup.

All measurements were obtained using the NVIDIA A100 80 GB deployment described in Section 3.7, with the default high quality preset and a fixed seed of 1. Each reported time in Section 4 represents an individual observation recorded during functional testing rather than a mean across repeated trials. Consequently, no variance or standard deviation is reported. The latency values therefore isualized d the approximate response time of the implemented system under the evaluated deployment conditions and are not intended to support fine-grained statistical comparisons between workflows.

Output quality was evaluated qualitatively through visual inspection of the generated meshes using the application's integrated 3D viewer. No quantitative geometric metric, such as Chamfer distance, Intersection over Union, or F-score, was computed because ground-truth 3D meshes were not available for the photographs and sketches used in the evaluation.

3.8.1 Evaluation Scope

The evaluation was not performed on the isualized d 3D reconstruction benchmark, but created as a prototype level system study based on the evidence provided by the implemented application. To evaluate the three workflows, functional execution of the text-to-3D, image-to-3D and sketch-to-3D processes were measured, an evaluation of representative generated outputs and preprocessing configurations was conducted qualitatively, and observed end-to-end generation latencies were evaluated. Due to the lack of ground-truth 3D meshes and multiple experimental runs, the study does not present quantitative measures of reconstruction, statistical variability, confidence intervals, or a formal evaluation of user-study measures.

3.8.2 Functional Test Cases

Functional tests were carried out for all the three kinds of input procedures. In the process of creating sketches to 3D objects, a hand drawn sketch went through the visual 3D generation pipeline. In image to 3D test, object images were used, and the vehicle image that was to be used later in preprocessing ablation was one of them. The procedure for text to 3D generation started with conversion of the text to an intermediate image using FLUX.1-schnell and then the image was processed by Hunyuan3D-2.1.

3.8.3 Timing Measurement

The end to end wall clock time was computed between a generation request is submitted and the generated GLB is available for download. This measurement covers only the steps following preprocessing, model inference, texture synthesis, writing files and polling job status; it does not include the initial upload of the source image or the cost of loading a model just one time when the backend is started. These measurements were taken with the default high quality preset and with a fixed seed of 1 in the deployed NVIDIA A100 80 GB environment. The figures reported are not necessarily means of several observations, and so there are not variance or standard deviation reported.

3.8.4 Preprocessing Ablation Protocol

The preprocessing analysis considered two comparable setups involving the same source image, best quality preset, seed 1, and deployment setting. Setup B involved background elimination followed by image cropping with square padding and $\rho = 0.16$, while Setup C incorporated alpha thresholding using $\tau = 96$ and $\rho = 0.12$. The previous case, where no preprocessing was conducted, was kept only for historical purposes and was not run again as a comparable experiment setup. This comparison was made qualitatively since there was no ground-truth 3D mesh or geometric metric available.

4. Results

4.1 Functional Results

All three input methods (text-to-3D, image-to-3D and sketch-to-3D) were able to generate textured GLB assets that could be isualized in the application. A 2D image to 3D reconstruction example is presented in figure 4, with an image of a house as the input and the resulting 3D reconstruction. The overall architectural form, proportions and dominant visual elements of the source image are preserved in the resulting model, and some fine architectural elements may be simplified during reconstruction.

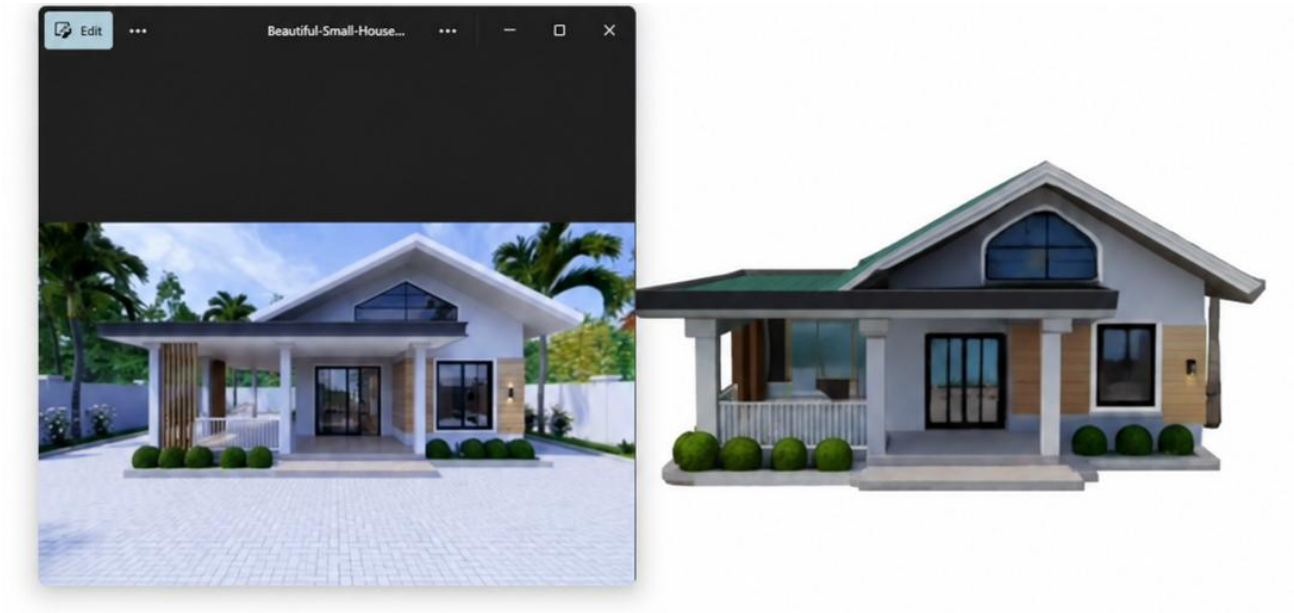


Figure 4. Representative image-to-3D result: source image of a house and the corresponding generated 3D reconstruction

The text-to-3D workflow was successful as intended. In the first instance, the textual prompt was converted to an image using FLUX.1-schnell, after which it was passed to Hunyuan3D-2.1, making use of the common 3D generation from images step. It is evident that the text-based inputs can be incorporated into the downstream 3D generation pipeline used by the other visual modalities.

4.2 End-to-End Latency

Table 2. Observed end-to-end generation time for each workflow on an NVIDIA A100 80 GB GPU

Workflow	Processing pipeline	Observed time
Sketch-to-3D	Sketch input → preprocessing → Hunyuan3D-2.1 → textured GLB	80.72 s
Image-to-3D	Image input → preprocessing → Hunyuan3D-2.1 → textured GLB	103.89 s
Text-to-3D	Text prompt → FLUX.1-schnell → Hunyuan3D-2.1 → textured GLB	≈105 s

Sketch to 3D showed the minimum end to end time of 80.72 seconds, with Image to 3D at 103.89 seconds, and Text to 3D close to 105 seconds. The above values are based on the observations of the entire workflow, not averages resulting from multiple runs. It is important to note that the difference between Image to 3D and Text to 3D is relatively small, but it does not represent the cost of computation for the FLUX.1-schnell phase alone. There was no attempt made to perform measurements separately per stage, nor were there any replication tests conducted.

4.3 Preprocessing Ablation Study

4.3.1 Experimental Comparison

Preprocessing Ablation examined the effect of progressive enhancement of object isolation before the image-conditioned 3D generation step. There were three configurations used for pre-processing. For the purposes of this report,

a matched evaluation of Configurations B and C was conducted in the same image, quality preset, seed and deployment environment, with the exception of Configuration A, which was retained as a historical baseline.

Configuration A—No preprocessing: the source image was passed directly to the 3D generation stage. This condition represents the historical baseline and was not re-run as a matched experiment.

- Configuration B—Background removal: segmentation was followed by cropping and square padding using Equations (2) and (3) with \(\rho = 0.16\), while the alpha threshold was disabled.
- Configuration C—Object-only cleanup: segmentation was followed by alpha thresholding with \(\tau = 96\), cropping, and square padding with \(\rho = 0.12\).

The comparison was qualitative and was intended to examine the visible effect of preprocessing on the observed ground-plane artefact rather than to establish a quantitative reconstruction improvement.

4.3.2 Qualitative Observations

Configuration B vs Configuration C is compared in Figure 5. In configuration B, background removal was successful in getting rid of most of the scene; however, there was an artifact of a flat ground plane underneath the car, indicating that background data still existed in the reconstructed geometry.

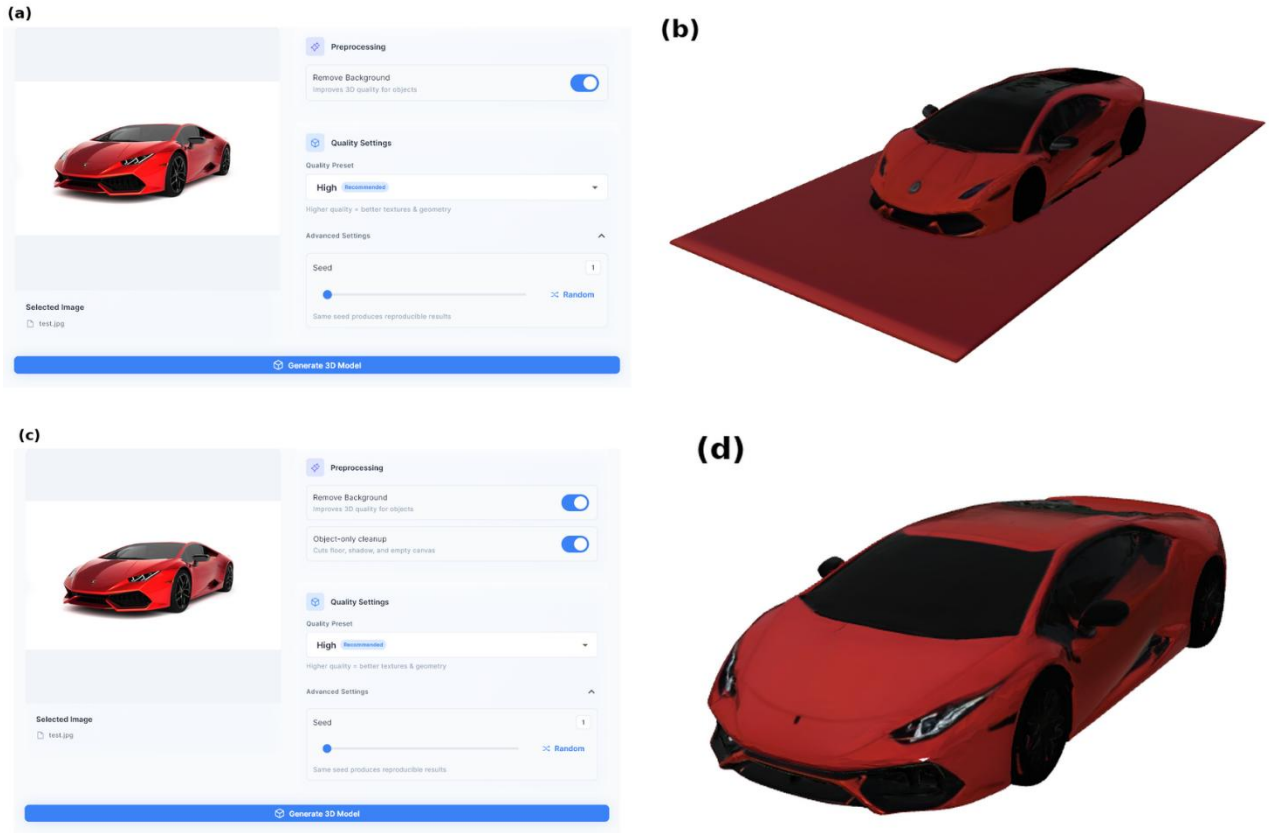


Figure 5. Qualitative preprocessing ablation: (a) Configuration B settings with background removal only; (b) resulting model showing the ground-plane artefact; (c) Configuration C settings with object-only cleanup; (d) resulting model with the ground-plane artefact removed

In the case of Configuration C, the additional step of alpha-thresholding and stronger cropping removed the ground plane artefact present, and the generated mesh was kept within the boundary of the car without losing the silhouette and major structures of the car.

The observed effects of the preprocessing configurations are summarised in Table 3.

Table 3. Preprocessing configurations and observed effects on the image-to-3D workflow

Configuration	Operations applied	Observed effect on the test case
A. No preprocessing	Source photograph passed through unchanged.	Ground-plane artefact observed during early testing, before preprocessing was implemented. Not re-run as a matched trial.
B. Background removal	Segmentation, then crop and square pad, Eqs. (2) and (3) with $\rho = 0.16$.	Most background removed, but a flat slab still appeared beneath the object (Figure 5(b)).
C. Object-only cleanup	Segmentation, alpha threshold Eq. (1) with $\tau = 96$, then crop and square pad with $\rho = 0.12$.	No ground plane; mesh confined to the object, with the silhouette preserved (Figure 5(d)).

4.3.3 Threshold Considerations

The threshold value for alpha was empirically selected as $\tau = 96$ at system development based on visual inspection rather than by systematically varying parameters. The objective of the threshold value was the suppression of residual areas of low alpha values resulting from the segmentation process, especially those near shadows and soft boundaries of objects. As the process uses a hard threshold, it makes all the retained pixels totally opaque; thus, transparency of semi-transparent materials cannot be maintained using this technique. Hence, no test was carried out for transparent objects (such as glass).

4.3.4 Preprocessing Computational Cost

The steps for the preprocessing are segmentation, alpha-thresholding, cropping, and padding. These steps are performed prior to generative inference. No independent measurements of the time it took to perform these steps have been made; thus, no quantitative assertion is made about the time overhead for the preprocessing step. The timings listed in Table 2 have been done with the preprocessing on.

5. Discussion

With the observed range of end-to-end generation times of 80.72 s to approximately 105 s, the developed framework proved to be capable of supporting individual 3D asset generation, although the proposed implementation is not ready to support any highly interactive or fast iterative workflow. The provided values are complete system-level observations as they account for the total amount of time spent on preprocessing, generative inference, texture synthesis, file processing, polling, and remote communications. Since each workflow was observed only once, the provided times must be regarded as prototype-level observations rather than statistically stable performance indicators.

With multi-input architecture, there is a single generation pipeline for text, image, and sketch-based workflows. Text inputs are first transformed into intermediate images before proceeding to the shared image-conditioned 3D generation stage, whereas sketches are considered visual inputs and go through the same Hunyuan3D-2.1 pipeline as photos. This modular approach eliminates redundancy between workflows and allows replacing or updating particular stages without redesigning the whole system.

One of the most important results of practical interest concerns the impact of input conditioning on the generated geometry. In the tested vehicle case, simply removing the background was not enough to remove the ground plane artefact, while the object-only input preparation was able to eliminate it via alpha thresholding along with stricter cropping and padding. This observation shows the practical significance of the object isolation for image-conditioned 3D generation. However, since the comparison was qualitative and carried out only in a few cases, it cannot be seen as proof of general numerical advantage.

The deployment aspect of remote GPU execution is the possibility of performing computationally expensive generation process without the need for high-end hardware on the client side. This approach is complemented by the dependencies on backend availability, network connectivity, and the cost of cloud resources usage. The choice of GLB as the output format contributes to compatibility with 3D- and web-capable applications and in-app preview functionality. Thus, the

study has shown that the combination of existing generative models, preprocessing, remote inference, and standardised output is feasible in the context of application-level 3D generation workflow.

6. Limitations

The present study is limited by a small number of test cases, the absence of standardised 3D benchmarks with ground-truth meshes, and single-run latency measurements; therefore, geometric accuracy, statistical variability, and generalisation cannot be established. The preprocessing evaluation was qualitative, and its parameters were selected empirically rather than through systematic optimisation. In addition, the prototype depends on remote GPU and network availability, uses in-memory job storage, supports only GLB export, and inherits the limitations of the underlying generative models, particularly for thin, reflective, transparent, and occluded structures. No formal user study or direct AR/VR runtime evaluation was conducted.

7. Conclusion and Future Work

In this study, a preprocessing-aware multi-input AI architecture has been described for the creation of 3D objects from text, images, and sketches in a single client-server application. This includes FLUX.1-schnell, Hunyuan3D-2.1, Flutter frontend, FastAPI backend, remote GPU computation, control of the preprocessing phase, and GLB format of 3D output. Functional tests revealed that all three workflows are executed successfully, with generation time of 80.72 s for sketch-to-3D, 103.89 s for image-to-3D, and 105 s for text-to-3D. The qualitative preprocessing ablation revealed that the observed artefact on the ground plane in the analysed vehicle case was successfully reduced by the object-only cleaning method which enhanced the object isolation prior to 3D generation. Thus, the potential of incorporating existing generative models, preprocessing, and cloud-based inference into a single 3D generation workflow at the application level has been shown.

Future work may consider more extensive and diverse testing, repeated timing tests, stage-based profiling, and quantitative reconstruction measures in cases when the ground truth 3D data is available. The preprocessing approach needs to be investigated in detail through controlled ablations and systematic exploration of its threshold and crop parameters. Possible improvements might include persistent jobs storage, enhanced backend communication, different export formats, post-generation 3D editing capabilities, user testing, and AR/VR evaluation.

References

[1] A. Nichol, H. Jun, P. Dhariwal, P. Mishkin, and M. Chen, “Point-e: A system for generating 3d point clouds from complex prompts,” *arXiv Prepr. arXiv2212.08751*, 2022.

[2] T. Hunyuan3D *et al.*, “Hunyuan3d 2.1: From images to high-fidelity 3d assets with production-ready pbr material,” *arXiv Prepr. arXiv2506.15442*, 2025.

[3] H. Qin *et al.*, “AViTS: Adaptive Spatiotemporal Token Selection for Efficient Dynamic-Resolution Generation,” in *European Conference on Computer Vision*, Springer, 2026, pp. 257–273.

[4] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, “Dreamfusion: Text-to-3d using 2d diffusion,” *arXiv Prepr. arXiv2209.14988*, 2022.

[5] C.-H. Lin *et al.*, “Magic3d: High-resolution text-to-3d content creation,” in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2023, pp. 300–309.

[6] Z. Wang *et al.*, “Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation,” *Adv. Neural Inf. Process. Syst.*, vol. 36, pp. 8406–8441, 2023.

[7] Y. Shi, P. Wang, J. Ye, L. Mai, K. Li, and X. Yang, “Mvdream: Multi-view diffusion for 3d generation,” in *International conference on learning representations*, 2024, pp. 39838–39859.

[8] H. Jun and A. Nichol, “Shap-e: Generating conditional 3d implicit functions,” *arXiv Prepr. arXiv2305.02463*, 2023.

[9] T. Yi *et al.*, “Gaussiandreamer: Fast generation from text to 3d gaussians by bridging 2d and 3d diffusion models,” in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2024, pp. 6796–6807.

[10] H. Lee, M. Savva, and A. X. Chang, “Text-to-3D Shape Generation,” in *Computer Graphics Forum*, Wiley Online Library, 2024, p. e15061.

[11] J. Liu *et al.*, "A comprehensive survey on 3d content generation," *arXiv Prepr. arXiv2402.01166*, 2024.
[12] R. Liu, R. Wu, B. Van Hoorick, P. Tokmakov, S. Zakharov, and C. Vondrick, "Zero-1-to-3: Zero-shot one image to 3d object," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, 2023, pp. 9264–9275.
[13] Y. Hong *et al.*, "Lrm: Large reconstruction model for single image to 3d," in *International Conference on Learning Representations*, 2024, pp. 50678–50702.
[14] J. Xu, W. Cheng, Y. Gao, X. Wang, S. Gao, and Y. Shan, "Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models," *arXiv Prepr. arXiv2404.07191*, 2024.
[15] X. Long *et al.*, "Wonder3d: Single image to 3d using cross-domain diffusion," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2024, pp. 9970–9980.
[16] D. Tochilkin *et al.*, "Triposr: Fast 3d object reconstruction from a single image," *arXiv Prepr. arXiv2403.02151*, 2024.
[17] L. Chen *et al.*, "3D-ISRNet: Generating 3D point clouds through image similarity retrieval in a complex background from a single image," *Image Vis. Comput.*, vol. 150, p. 105203, 2024.
[18] Y. Tong, H. Chen, N. Yang, M. I. Menhas, and B. Ahmad, "3D-CDRNet: Retrieval-based dense point cloud reconstruction from a single image under complex background," *Displays*, vol. 78, p. 102438, 2023.
[19] Y. Yang and B. Yang, "Benchmarking and analysis of unsupervised object segmentation from real-world single images," *Int. J. Comput. Vis.*, vol. 132, no. 6, pp. 2077–2113, 2024.
[20] W. Cai *et al.*, "Instruct Pix-to-3D: Instructional 3D object generation from a single image," *Neurocomputing*, vol. 600, p. 128156, 2024.
[21] S. K. Ramadhan, N. R. Najem, A. Sulaiman, H. A. Rammo, and F. A. Shaban, "Beyond Chronological Age: Explainable Multi-Task Swin Transformers for Biological Age and Gender Estimation from Dorsal Hand Images.," *Int. J. Comput. Electron. Asp. Eng.*, vol. 7, no. 2, p. 227, 2026.
[22] J. Xiang *et al.*, "Structured 3d latents for scalable and versatile 3d generation," in *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2025, pp. 21469–21480.
[23] Z. Zhao *et al.*, "Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation," *arXiv Prepr. arXiv2501.12202*, 2025.
[24] T. Chen *et al.*, "Deep3dsketch+: Rapid 3d modeling from single free-hand sketches," in *International conference on multimedia modeling*, Springer, 2023, pp. 16–28.
[25] P. Esser *et al.*, "Scaling rectified flow transformers for high-resolution image synthesis," in *Forty-first international conference on machine learning*, 2024.
[26] C. Wang, H.-Y. Peng, Y.-T. Liu, J. Gu, and S.-M. Hu, "Diffusion models for 3D generation: A survey," *Comput. Vis. Media*, vol. 11, no. 1, pp. 1–28, 2025.
[27] F. Documentation, "Flutter: Build apps for any screen," *Google LLC. Retrieved from https//docs. flutter. dev*, 2026.
[28] P. Pataranutaporn, A. Doudkin, and P. Maes, "OceanChat: The effect of virtual conversational AI agents on sustainable attitude and behavior change," in *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*, 2026, pp. 1–25.
[29] S. Tiangolo, "FastAPI documentation," *URL https//fastapi. tiangolo. com*, 2025.
[30] L. Körösi *et al.*, "From Detection to Action: Implementing Deep Learning Inference in PLC Systems via Docker Services," *IFAC-PapersOnLine*, vol. 58, no. 9, pp. 19–24, 2024.