International Journal
Information Technology Applications (ITA)

Volume 15, Number 1, July 2026

AI-BASED DEEP FAKE AVATARS USING GAUSSIAN SPLATTING

Ondrej Hudcovič

Abstract:

This paper investigates the technical feasibility of creating an interactive conversational avatar using Gaussian Splatting technology in the Unreal Engine environment. The study focuses on verifying the possibility of implementing a talking avatar without using parametric face models. The result of the study is a prototype consisting of a local FastAPI backend and a client application in Unreal Engine 5.3.2, which integrates speech recognition, response generation by a local large language model, speech synthesis and data generation for lip synchronization. The avatar is rendered using the 3D Gaussian Splatting plugin from Luma AI. The paper includes the implementation of the application in the VR environment of Unreal Engine. The implementation section of the study also introduces comparative testing of four large language models and latency measurement of the entire conversational data pipeline. The study confirms the technical feasibility of the combination of Gaussian Splatting and a conversational AI application only partially, as the main open problem remains the implementation of the avatar's lip animation.

Keywords:

Gaussian Splatting, conversational avatar, virtual reality, large language models, Unreal Engine.

Introduction

Virtual reality applications have become increasingly popular in recent years, not only in the fields of entertainment and gaming, but also in the fields of tourism, education, healthcare, schooling, remote communication and many others. One way to achieve high-quality and immersive VR experiences is to have avatars with which users can interact naturally. Traditional approaches to creating avatars involve time-consuming manual modeling or the use of expensive motion capture systems. This naturally poses a problem for small developers and teams of people with limited resources.

Recent advances in neural rendering, especially Gaussian Splatting, open up new possibilities for creating realistic 3D representations from simple photographic inputs. At the same time, rapid developments in the field of large language models enable natural speech interactions. The overlap of these technologies thus presents an opportunity to create intelligent, visually realistic avatars that are capable of conducting meaningful conversations.

1 State of the Art

3D Gaussian Splatting represents a revolutionary change in the neural scene model. Unlike NeRF, which uses implicit functions, Gaussian Splatting directly represents the scene as a collection of 3D Gaussian elementary graphics elements.

Each Gaussian is defined by several parameters: its position in 3D space; a covariance matrix that defines its shape and orientation; a color that is expressed using spherical harmonics for angle-dependent effects; and a transparency value.

Rendering a 3DGS scene involves mapping each 3D Gaussian onto the image plane, where they are subsequently converted into 2D Gaussian points. These 2D Gaussian points are then combined using alpha blending in conjunction with a depth sequence. This process, called splatting, is highly parallelizable and can be efficiently used with modern graphics processors. The data pipeline for creating a 3DGS model begins with capturing a series of photographs of the subject we want to display from multiple angles. These photographs must overlap sufficiently so that the "structure from motion" algorithms can estimate the photographic poses and reconstruct the underlying scattered point cloud [1].

The result is a representation that achieves quality comparable to NeRF while also offering faster rendering. Real-time frame rates are thus achievable even for complex scenes, enabling the use of 3DGS in VR applications [2].

The most commonly used tool for this purpose is COLMAP, which extracts features using SIFT features, assigns individual features across all images, and uses bundle adjustment, a technique that optimizes camera parameters and 3D point positions. The scattered point cloud from COLMAP serves as the starting point for Gaussian Splatting optimization. Each such point becomes the center of a Gaussian, with the underlying covariance and color parameters estimated based on local image statistics. The optimization then refines these parameters and adaptively adds or removes Gaussian points to improve the quality of the image reconstruction [3].

Several tools can be used to simplify the data flow for end users. Polycam is an application that automates the process of capturing photos. It guides users to capture photos from appropriate angles and then automatically creates an image reconstruction [4].

Talking head models are a rapidly developing field that connects several fields such as computer vision, deep learning, and speech synthesis. These models are animated, human heads that speak and express emotions with high visual realism [5].

Lip sync is a field that deals with the production of speech in animated characters so that their lip movements match the phonemes in audio recordings [6].

To overcome the limitations associated with specific languages, auxiliary representations based on phonemes and visemes have been proposed. An example is the framework The Multilingual Experts (MuEx), which uses an architecture based on phoneme-driven expert mixing to ensure strong audiovisual matches [7].

Existing approaches to talking head avatar synthesis can be categorized into five main groups based on their underlying methodology. Each approach differs in terms of visual quality, computational complexity, and suitability for real-time applications, as summarized in the (Tab.1) [8].

Table 1. Overview of methods for synthesizing talking avatars

Type	Description	Advantages	Limitations
2D GAN	Generation via adversarial networks	Visual realism, speed	Training instability, limited to 2D
Diffusion models	Iterative denoising process	Identity preservation, temporal consistency	Slow, expensive
3D morphable models	FLAME, MetaHuman	Geometric consistency, full facial control	Requires 3D data, lower photorealism
NeRF	Neural 3D scene representation	High perceptual quality	Slow rendering, not for real-time
Gaussian Splatting	Explicit 3D Gaussian representation	Real-time rendering, photorealism	No native facial animation support

2 Design of Application

The aim of this paper is to explore the possibilities of creating a VR application implementing a talking avatar created using Gaussian Splatting technology in the Unreal Engine environment. We want to achieve this goal by creating an interactive application that allows the user to communicate with an avatar connected to a large language model. This application is to be integrated into Unreal Engine and into virtual reality using the Meta Quest device.

2.1 Methodology

The system was designed iteratively, rather than comprehensively at the beginning. The application was created by gradually adding components and each iteration of the development had a firmly established and functionally defined goal. Decisions on the selection of individual technologies were made based on the results of previous iterations. This procedure allowed for the acceptance or rejection of individual tools and technologies after practical verification, like detected incompatibility with particular components.

2.2 System Architecture

The development environment consists of several key elements. The entire prototype is developed on macOS Tahoe 26.2 based on Apple Silicon M1 Max. The graphical output of the application is provided by the Unreal Engine 5 game engine. A local Python FastAPI server was used for the backend. The physical availability of the VR headset was limited during development, and therefore the functionality was primarily developed and verified on the desktop in Unreal Engine 5.3.2. VR output is planned as a final validation, not an ongoing part of development.

The designed system consists of two main parts, namely the frontend implemented in Unreal Engine and the backend as a locally running server. Communication between these two entities is carried out using the HTTP protocol.

The frontend provides for the display of the GS avatar, the capture of audio input from the user and the playback of audio output. The backend processes the entire conversational part of the application, namely speech transcription, response generation, speech synthesis and data for lip synchronization. The final visual output of the imported GS avatar can be seen in (Fig.1).

Fig.1. GS avatar rendered in Unreal Engine 5.

The data flow of the conversational application begins when the user presses the K button in the Unreal Engine application. The entire system architecture is shown as a diagram in (Fig.2).

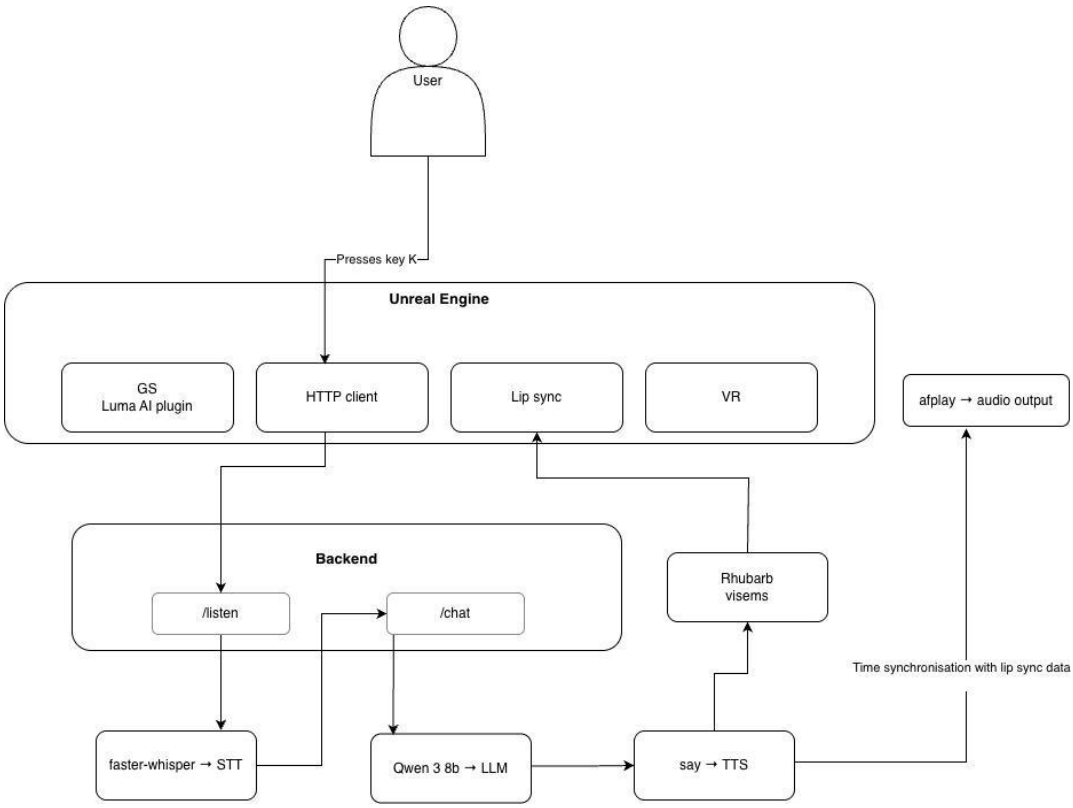

Fig.2. Architecture diagram of the application.

2.3 Testing methodology

System testing was divided into two separate parts. The first part focused on measuring the latency of individual components of our application. Each component was measured separately in order to identify dominant sources of high latency. The overall response was measured by repeatedly executing the data pipeline, with the measurement starting from the moment the user launched the data flow and ending at the moment when both audio and lip sync data were available.

Table 2. Latency measurements

Component	Average	Range	Note
HTTP round-trip	7ms	3 - 15ms	Local, negligible
LLM	2.5s	2 - 3,5s	Depends on the length of text
TTS	0.8s	0,5 - 1,2s	Depends on the length of text
FFmpeg AIFF -> WAV/MP4	0.25s	0.1 - 0.35s	
Rhubarb	0.6s	0,4 - 0,9s	Depends on the length of audio
Total	4.3s	3,2 - 6,2s	

The second part of the testing focused on comparing language models. Since the choice of model directly affects the quality of the user experience, manual comparative testing on a fixed set of prompts was designed. To make the results comparable across models, each model was given an identical system prompt and the same set of test questions. The questions were designed to cover various aspects of the model's behavior relevant to the given use case, in particular the quality of answers in the domain, language support, persona preservation and the model's resistance to prompt injection. The models were tested via the native interfaces of each individual model, either via a web interface or a terminal.

ChatGPT performed the worst in the test, which could be due to the fact that it was a free version, while the Claude models were paid versions. Both Claude models were flawless in the test. The Qwen3-8B model was identical in quality in English questions, but in the case of Slovak questions there was a slight decrease in the quality of the answers. One example is an answer in English to a question in Slovak. The assumption is that a better result could be achieved with appropriate prompt engineering. Since local use was preferred due to the reduction of the overall latency of the application, the local Qwen3-8B model was chosen for the purposes of this work. The summary of the results can be seen in (Tab.3).

Table 3. LLM model comparison.

Model	Quality of answers	Persona preservation	Open-source
Qwen3	5	yes	4
Claude Haiku	5	yes	5
Claude Sonnet	5	yes	5
ChatGPT (free)	4	no	3

3 Discussion

The tool selection for this project was achieved by searching for standard publicly available tools and then testing the functionality locally. For several tools, alternatives had to be chosen, as they were incompatible versions, either in relation to the operating system or a specific version of the Unreal Engine.

Several plugins were tested for rendering the Gaussian Splat avatar in the Unreal Engine. The XV3DGS plugin did not display the UI import after installation in UE 5.5, making it practically unusable. Although UnrealSplat supported UE 5.5, it only contained Windows binaries - this plugin could not be compiled on macOS. The only functional solution was the Luma AI plugin, which officially supports macOS and Unreal Engine 5.3. It was precisely because of this compatibility that the project was moved from UE 5.5 to version 5.3.2. No available GS plugin was able to work on macOS in a newer version of the engine, so a downgrade was necessary.

The backend is implemented as a FastAPI application running on Python 3.11.10. The Python version is managed via pyenv and each project runs in its own virtual environment, which prevents incompatibility between individual libraries.

Local inference via Ollama with the Qwen 3 8B model using a local REST API was chosen as the language model. Compared to the tested models, it showed sufficient quality of responses while allowing local operation without dependence on cloud services.

Piper TTS was originally considered for speech synthesis. However, it was rejected because its repository was archived and its macOS compatibility was broken. The native say tool, which is part of macOS, was used as a replacement. The voice quality is lower than with dedicated TTS tools, but the solution does not require any external dependencies. This compromise was acceptable for the purposes of the paper.

The measured latency is on the edge of acceptability for an interactive VR experience. Improvements would be achieved by replacing the local model with a larger or faster model, more powerful HW, or optimizing Rhubarb generation with parallel processing.

The most serious problem of the resulting solution is related to the limitation of GS representation for animation in Unreal Engine through available plugins. These do not allow graphical modification of the inserted model, but consider the given model as a whole. Since some of the available plugins are open-source projects, one solution could be to modify the code. This modification would have to ensure that relevant parameters are exposed and modifiable in Unreal Engine, which would allow graphical manipulation. However, this step is beyond the scope of this thesis.

Another approach to animating avatars is described in the GaussianAvatars method, where Gaussians are bound to a FLAME mesh and trained together from scratch on a video taken from multiple angles [9]. However, this method is outside of the scope for our paper, as the implementation was carried out exclusively in Unreal Engine without access to similar training infrastructure.

Conclusion

The aim of this study was to investigate whether it is possible to create a functional real-time conversational avatar using Gaussian Splatting in the Unreal Engine environment. The result of the work is a functional prototype that verifies this question in practice.

A system consisting of two main parts was designed and implemented, namely a backend built on FastAPI, as well as a client application in Unreal Engine 5. The backend provides the entire conversational data stream: speech recognition via the faster-whisper tool, response generation using the local language model Qwen 3 8B via Ollama, speech synthesis, audio format conversion and lip sync data generation using the Rhubarb tool. The Gaussian Splatting avatar is rendered in real-time using the Luma AI plugin, while lip synchronization could not be solved. The system was successfully integrated into the VR environment using the UE plugin OpenXR.

The work verifies the technical feasibility of combining a Gaussian Splatting avatar and a real-time conversational LLM data stream. This is an area for which there were no established and standardized solutions at the time of this work.

The resulting prototype can serve as a basis for future interactive installations combining photorealistic representation of characters with conversational AI systems. A positive contribution of this work is also the test of large language models, the more detailed results of which can be found in Appendix 1. These results can also be used outside the domain of interactive conversational applications.

The most significant limitation remains the inability to easily implement lip animations. The overall pipeline latency, averaging around 4.3 seconds, is on the limit of acceptability for a smooth conversational experience. The system was developed and tested primarily on a desktop display, with VR testing limited by the availability of a headset. Due to incompatibility, audio playback was handled on the local server side, which represents an architectural limitation when deployed on another platform, and for future development it would be more appropriate to implement audio playback directly in Unreal Engine.

The immediate next step is to complete the lip sync integration. A feasible solution is to implement sprite overlay visemes directly in the Unreal Engine Blueprint logic based on data generated by the Rhubarb tool. Consequently, the greatest potential for improvement is to replace sprite overlay with a more sophisticated method. This could be, for example, an extension or modification of available open-source plugins that would allow Gaussian Splatting avatar deformation and animation directly in Unreal Engine, or it could be the use of one of the currently developed or researched solutions for animated GS avatars.

Acknowledgement

The author would like to thank RNDr. Ján Lacko, PhD. for his support during the development of this paper. Additionally, the author wishes to express gratitude to the Faculty of Informatics at Pan-European University for providing the resources and academic environment to complete the study. This paper was conducted as part of the diploma thesis titled AI-based deep fake avatars using Gaussian Splatting.

References

[1] Bernhard Kerbl, Georgios Kopanas, T. Leimkühler, and G. Drettakis, (2023). 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, vol. 42, no. 4, pp. 1–1, doi: https://doi.org/10.1145/3592433.

[2] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, (2020). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. arXiv:2003.08934.

[3] J. Schönberger and J.-M. Frahm, (2016). Structure-from-Motion Revisited. Available: https://demuc.de/papers/schoenberger2016sfm.pdf.

[4] Polycam, (2024). Polycam - LiDAR 3D Scanner. Available: https://poly.cam/.

[5] H. Nisar, S. Masood, Z. Malik, and A. Abid, (2026). Talking Head Generation Through Generative Models and Cross-Modal Synthesis Techniques, *Journal of Imaging*, vol. 12, no. 3, p. 119, doi: https://doi.org/10.3390/jimaging12030119.

[6] N. H. Loh, (2014). Development of Real-Time Lip Sync Animation Framework Based On Viseme Human Speech, *Archives of Design Research*, vol. 112, no. 4, p. 19, doi: https://doi.org/10.15187/adr.2014.11.112.4.19.

[7] Caglar, E., Oskooei, A. R., Kutanoglu, M., Keles, M., & Aktas, M. S. (2025). Asynchronous Pipeline Parallelism for Real-Time Multilingual Lip Synchronization in Video Communication Systems. arXiv preprint arXiv:2512.18318.

[8] H. Nisar, S. Masood, Z. Malik, and A. Abid, (2026). Talking Head Generation Through Generative Models and Cross-Modal Synthesis Techniques, *Journal of Imaging*, vol. 12, no. 3, p. 119., doi: https://doi.org/10.3390/jimaging12030119.

[9] Qian, S., Kirschstein, T., Schoneveld, L., Davoli, D., Giebenhain, S., & Nießner, M. (2024). Gaussianavatars: Photorealistic head avatars with rigged 3d gaussians. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 20299-20309).

Author

Bc. Ondrej Hudcovič
Faculty of Informatics
Pan-European University in Bratislava, Slovakia
xhudcovic@paneurouni.com
Data Platform Engineer with background in Information Systems and AI. Currently studying Applied Informatics at the Faculty of Informatics at Pan-European University in Bratislava.