provadis Hochschule

Cook Book

Human Presentator: From Voice to Animated Avatar

Markus Siegert & Lucas Buchholz

Student ID: D533 & D???

Bachelor of Science Informatik

Primary Examiner: Richard Beetz

Deadline: 27.08.2025

Contents

List of Figures

List of Tables

List of Listings

Acronyms

API	Application Programming Interface.
CLI	Command-Line Interface.
DTIT	Deutsche Telekom IT GmbH.
memo	memoavatar/memo.
Zonos	Zyphra/Zonos.

1 Introduction

This cookbook presents a fully local pipeline for generating talking-head animations from static images and arbitrary text, leveraging Zyphra/Zonos (Zonos) for voice cloning and text-to-speech and memoavatar/memo (memo) for precise lip-synchronization. Designed to run entirely on your own hardware, this application ensures data privacy, offline functionality, and the flexibility to customize every component.

In today's media landscape, immersive and personalized audiovisual content drives engagement across educational platforms, virtual assistants, and marketing campaigns. By combining high-fidelity voice cloning with accurate mouth animation, creators can produce dynamic videos without relying on cloud services or external APIs. This guide walks you through setting up the environment, preparing your data, and executing each stage of the pipeline, from extracting a speaker's voice characteristics to animating a still image's mouth movements to match the synthesized speech.

At the core of this pipeline are two specialized tools: Zonos and memo. Zonos provides an end-to-end solution for voice cloning and speech synthesis, enabling you to input a short audio sample and generate natural-sounding speech for any text. memo handles the visual side, mapping the generated audio waveform onto a static image to produce lipid-accurate mouth movements. You'll find detailed instructions for installing dependencies, configuring models, and running inference scripts, alongside code snippets and troubleshooting advice.

While several open-source and commercial solutions address parts of this workflow, most depend on cloud infrastructure or separate services. Projects like Real-Time-Voice-Cloning and Mozilla TTS excel at local voice synthesis, and Wav2Lip and SyncNet offer speaker-agnostic lip-sync capabilities. Commercial tools such as Adobe Character Animator or CrazyTalk let users animate images without code, whereas platforms like Synthesia and D-ID provide end-to-end video creation online. This cookbook demystifies the underlying algorithms, empowering you to implement a cohesive, offline solution that you can extend for research or production purposes.

2 Methodology

The human-presentator project was initially conceived as a streamlined, single-script Python application designed to generate talking-head animations from static images and text input. The original approach emphasized simplicity and cohesion, with all functionality consolidated into one comprehensive Python script that would handle the entire pipeline from text input to final video output.

Rather than developing proprietary solutions from scratch, the project methodology centered on leveraging established open-source software for the core voice generation and image animation

components. This approach was chosen to reduce development time by building upon proven, well-tested codebases, ensure compatibility with existing workflows and standards and maintain transparency and auditability through open-source implementations.

An integral component of the original methodology included implementing automated validation mechanisms for the generated MP4 output files. The automated validation approach aimed to ensure consistent output quality while reducing manual review overhead during the development and testing phases.

The initial architectural approach emphasized a monolithic design pattern, consolidating all processing stages within a single executable unit. This methodology was selected to minimize deployment complexity, reduce inter-component communication overhead, and provide a clear, linear processing flow that could be easily understood and modified by developers working on the project.

3 Results

The implementation of the human-presentator project deviated significantly from the initially planned monolithic approach described in the methodology section. During development, several technical constraints and compatibility issues necessitated a fundamental architectural redesign that resulted in a modular, service-oriented system.

The most significant change from the original methodology was the transition from a single-script Python application to a distributed 'microservice' like architecture. This shift was primarily driven by Python version incompatibilities between the core dependencies, specifically Zonos for voice synthesis and Memo for face animation. Rather than attempting to reconcile these version conflicts within a monolithic structure, the implementation adopted a containerized approach using Docker Compose to isolate each service in its own environment with appropriate Python versions and dependencies.

The final implementation consists of three primary services: a Vue.js frontend (presentor-vue) serving as the user interface, a Zonos API service for voice recognition and synthesis running on port 8001, and a Memo API service for face animation processing on port 8002. This modular architecture allows each component to operate independently while maintaining clear communication channels through RESTful APIs.

A critical implementation decision involved forking both the Zonos and Memo projects to adapt their system APIs for the specific requirements of the human-presentator use case. The original Zonos and memo implementation was modified to better integrate with the distributed architecture. The Memo fork in particular was adapted to try optimize face animation processing but without real achievements.

The containerization strategy employed Docker with NVIDIA GPU runtime support to leverage hardware acceleration for the computationally intensive AI operations. The memo-api service specifically utilizes NVIDIA CUDA capabilities for face animation processing, while the zonos-api service can optionally leverage GPU acceleration when sufficient VRAM is available. As for now it is disabled because it then would take up to 32GB VRAM.

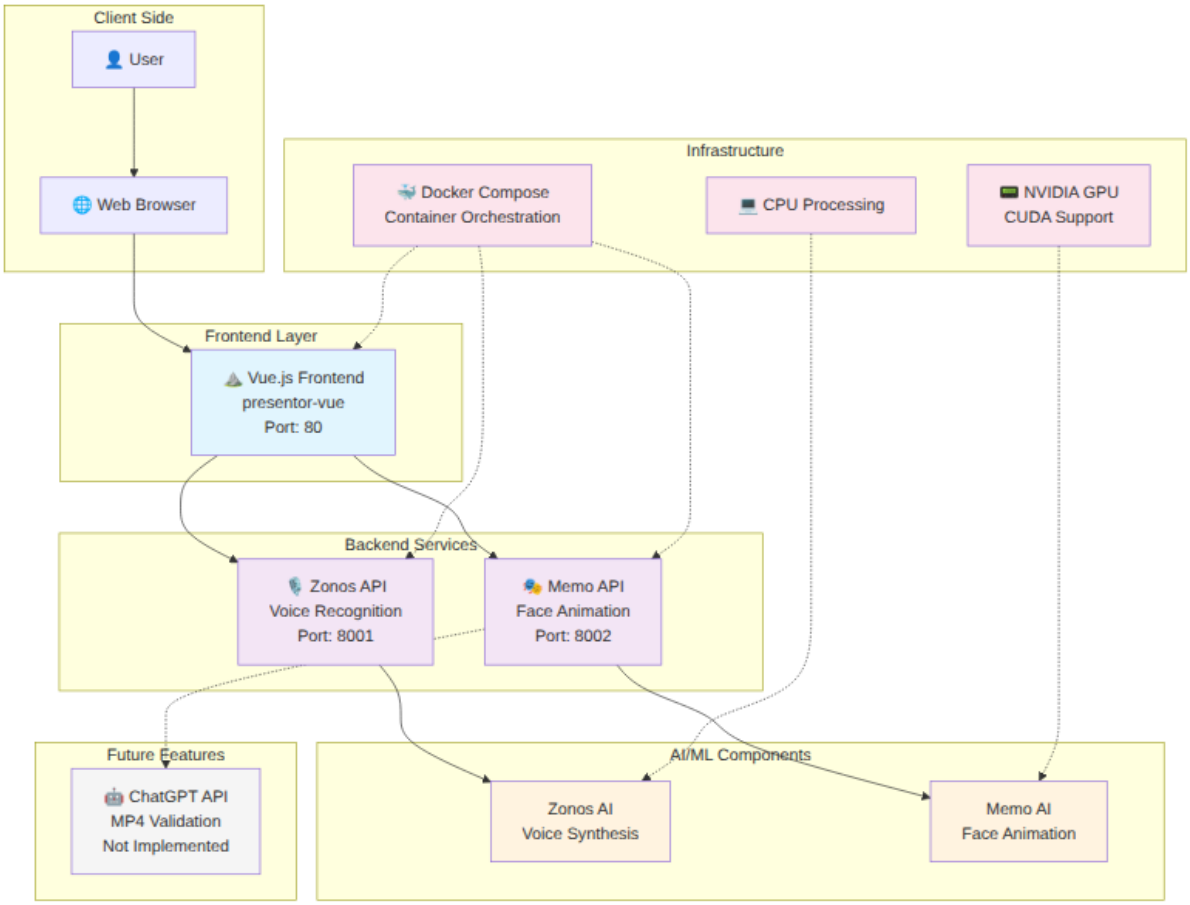

Figure 1: Architecture of the Human Presentor

4 Discussion

The Human Presenter Generator represents a sophisticated pipeline for creating realistic talking-head animations, combining Zonos for voice synthesis and memo for facial animation. While technically impressive, the system's practical usability presents both opportunities and significant challenges for end users.

From a usability perspective, the application provides an intuitive two-step workflow through its Vue.js-based web interface. Users begin by either recording voice samples directly through their microphone or uploading existing audio files to the Zonos component. The interface guides users through entering the desired text and language parameters before generating synthetic speech. Subsequently, the memo component allows users to capture face images via webcam or upload static photographs, which are then animated to match the generated audio. This streamlined

approach makes the complex underlying technology accessible to non-technical users through clear visual feedback and straightforward controls.

However, the system's hardware requirements pose substantial barriers to widespread adoption. The implementation demands significant computational resources, particularly GPU memory, with the documentation indicating that even systems with 20GB of VRAM proved insufficient for running both Zonos and memo simultaneously. The Docker configuration reveals that while the memo component requires NVIDIA GPU acceleration for acceptable performance, the Zonos service had to be configured to run without GPU acceleration due to memory constraints. This hardware limitation forces users to either invest in high-end graphics hardware or accept significantly degraded performance, making the system impractical for typical consumer hardware configurations.

The application's architecture, built around separate FastAPI microservices for each AI component and a Vue.js frontend, demonstrates good separation of concerns but introduces additional complexity for deployment and maintenance. Users must manage multiple Docker containers and ensure proper GPU driver integration through the NVIDIA Container Toolkit. The reliance on local processing, while ensuring privacy and offline functionality, places the entire computational burden on the user's system rather than distributing it across cloud infrastructure.

Despite these hardware challenges, the system offers valuable advantages for users with appropriate computational resources. The fully local processing ensures complete data privacy, as no voice samples or images are transmitted to external services. The modular architecture allows for customization and extension of individual components, making it suitable for research applications or specialized use cases. The continuous workflow from voice synthesis to facial animation within a single application eliminates the need for users to coordinate multiple separate tools and manual file transfers between processing stages.

5 List of AI

No AI was used.

6 Declaration of independence

I hereby confirm that I have written this paper personally and independently and have used no sources other than those indicated. All passages taken literally or in spirit from other sources are marked as such. The drawings, figures and tables in this thesis have been created by me or have been provided with an appropriate source reference. This thesis has not been submitted in the same or a similar form by me to any other university for the purpose of obtaining an academic degree.

Berlin, August 6, 2025 Markus Siegert & Lucas Buchholz

Appendix

Contents

A Source Code

```
> Eine Nudel is ein - also n ganz‘ Tach.
> Man kann im Winter Sommer,
> Man kann morgens abends essen,
> ein Nudel is wirklich ein sehr leckeres Gericht.
>
> Man kann Nudeln machen warm,
> man kann Nudeln machen kalt.
> Also man kann in den Urlaub,
> man kann nen Picknick machen,
> man kann abends morgens essen,
> wie man gerade Hunger hat.
>
> Dawegen spielt’s keine Tageszeit,
> keine Sonnenzeit, kein Winter oder so -
> die Italiener essen es auch jeden Tach.
> Also ist eine Nudel ein ein ein ewig’s Essen.
>
> Egal aus was ist die Nudel gemacht ist,
> egal wie Du Nudel zubereitet wird.
> Jeder hat seinen Geschmack,
> einer ist gerade morgens aufgestanden
> ist - äh - gerne Nudeln
> der macht sich auch morgens Nudeln -
> kein bisschen...
>
> Dawegen lass ich auf Nudeln nix kommen
> Humor muss sein.
> Dafür ist ja die humorvolle italienische Nudeln,
> der hat ja auch Humor der Italiener.
>
> Eine Nudel kräftigt -
> eine Nudel kräftigt nicht nur,
> die hat Energie bringt sie,
> auch nich dann noch dafüchtüch
> wenn man die gegessen hat,
> hat man auch Freude an den ganzen Tag
> um so schneller ist man, oder -
> umso mehr baut man seine Kraft -
> und seine Freude auf.
>
> Das ist das Wichtige, ha haaa!
>
> Die Nudel schmeckt, dadrauf kommts an.
```

Listing 1: Peter Ludolfs "Ode an die Nudel"