PAN-EUROPEAN UNIVERSITY

Report no. 2.2

Guidelines for creation of intelligent avatars

Version: 2.0

Language: English

Project name: **Intelligent avatars for digital heritage applications**

Project acronym: **IntAvaVR**

Project no: **09I03-03-V04-00495**

Authors:

Lead researcher: **Prof. Dr. Selma Rizvić**

Other authors: **Danijel Briski, RNDr. Ján Lacko, PhD., doc. RNDr. Eugen Ružický, CSc., Ing. Juraj Štefanovič, PhD.**

Date: **26.9.2025**

Financované Európskou úniou NextGenerationEU

PLÁN OBNOVY

ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU

VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA

VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495.”

Guidelines for creation of intelligent avatars

Creating 2D intelligent AI-based avatars in virtual reality (VR) applications requires careful consideration of both technical and user experience guidelines. First, the avatar's visual design must be consistent with the VR environment and the application's purpose. This includes selecting appropriate art styles, color schemes, and animations that ensure the avatar appears engaging and expressive without causing visual fatigue. Developers should also consider scalability, ensuring avatars can perform a wide range of expressions and gestures while remaining lightweight enough to run smoothly on different VR hardware. Optimization techniques, such as sprite sheets and vector-based animations, can help maintain high performance without compromising visual quality.

Equally important are the AI-driven behavioral and interaction guidelines. Avatars should possess adaptive intelligence capable of responding to user inputs, environmental cues, and contextual scenarios in real-time. This involves implementing natural language processing for verbal communication, emotion recognition to adjust expressions, and procedural behavior models that guide autonomous actions. To maintain immersion, interactions should feel intuitive and human-like, with responses that are contextually appropriate and timely. Developers should also ensure that the AI respects ethical and privacy considerations, avoiding behaviors that may seem intrusive or manipulative to users.

Usability and accessibility guidelines are crucial for maximizing the effectiveness of 2D AI avatars in VR. Developers should design avatars to accommodate diverse user needs, including options for adjusting visual clarity, speech output, and interaction complexity. Testing with a broad user base helps identify potential issues in communication, comprehension, and emotional engagement. Furthermore, avatars should support modularity, allowing developers to update or customize behaviors and appearance over time without disrupting the user experience. By combining visual coherence, intelligent responsiveness, and inclusive design, developers can create 2D AI avatars that enhance immersion and foster meaningful interactions within VR applications.

1. 2D Data acquisition

Step 1: Define Data Requirements

Clearly specify the objectives of the video dataset, including the types of actions, gestures, or expressions to capture. Determine the required resolution, frame rate, and duration of recordings. Consider factors such as the target application (e.g., AI training for avatar motion or gesture recognition) and the diversity of subjects, backgrounds, and lighting conditions needed to ensure robust model generalization.

Step 2: Setup of Green Screen Environment

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

Prepare a controlled environment with a uniform green backdrop. Ensure the green screen is evenly lit to minimize shadows, reflections, and color spill. Use diffused lighting from multiple angles to achieve consistent illumination across the subject and the background. Calibrate the camera to maintain accurate color capture and avoid motion blur or frame drops.

Step 3: Subject Preparation

Ensure subjects wear clothing and accessories that do not match the green screen color to prevent chroma keying errors. Instruct subjects on desired movements, gestures, or expressions according to the dataset plan. Maintain consistent distance from the camera and the green screen to avoid perspective distortion.

Step 4: Video Capture

Record the video at the pre-determined resolution and frame rate. Use multiple takes if necessary to capture variations in motion or expression. Monitor real-time video for issues such as shadows, color spill, or occlusions that could interfere with masking.

Step 5: Masking and Background Removal

Apply chroma keying to isolate the subject from the green screen. Generate accurate masks for each frame using automated or semi-automated methods, followed by manual refinement to handle complex edges (e.g., hair, clothing folds). Store the resulting foreground masks along with the original videos for downstream processing.

Step 6: Quality Control and Annotation

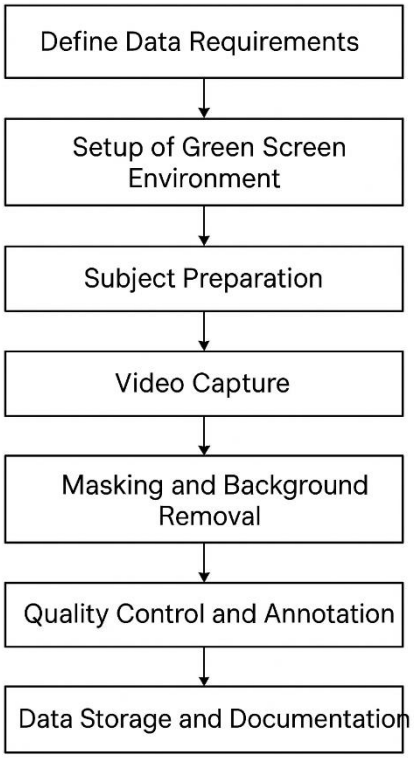
Inspect the masked videos for artifacts, inconsistencies, or missing data. Annotate frames with relevant metadata, including subject ID, action type, or temporal markers. Ensure the dataset meets statistical diversity requirements in terms of subjects, motions, and lighting conditions.

Step 7: Data Storage and Documentation

Organize the processed videos and masks systematically with clear file naming conventions. Document acquisition conditions, camera settings, lighting setup, and masking procedures to facilitate reproducibility and model training transparency.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."



1.1 Scene setup

Green Screen Setup:

A flat, seamless green screen backdrop was used, positioned vertically behind the subject. The surface was stretched tightly to eliminate wrinkles and texture that could interfere with chroma keying. The green screen extended beyond the subject's movement boundaries both horizontally and vertically to ensure full coverage during recording.

- A 3 m (width) × 2.5 m (height) chroma green backdrop was mounted on a support frame, placed 1.5 m behind the subject.
- The bottom edge of the screen extended to the floor to allow full-body capture without floor reflections.
- The green surface was illuminated separately to maintain consistent luminance between 40–50% on the waveform monitor, avoiding hotspots or shadows.

Camera Position:

A single high-resolution digital video camera was placed at eye-level with the subject, centered along the frontal axis. The distance between the camera and the subject was set to approximately 2.5–3 meters, ensuring a clear field of view that captured the subject's full body without distortion.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

The focal length was adjusted to minimize lens warping, and the camera was stabilized on a tripod to maintain consistent framing throughout the capture session.

- Camera: Single 4K digital video camera (e.g., DSLR or mirrorless with HDMI output).
- Tripod height: 1.6 m (aligned with the subject's eye level).
- Distance from subject: 2.8 m to achieve a full-body frame with a 50 mm focal length lens (minimizing perspective distortion).
- Angle: 0° (directly facing subject, perpendicular to green screen plane).
- Frame rate & resolution: 30 fps at 3840×2160 pixels.
- White balance: Fixed at 5600 K to match daylight-balanced lights.
- Shutter speed: 1/60 s, Aperture: f/4 for depth of field sufficient to keep subject sharp while slightly softening the green screen.

Lighting Arrangement:

A three-point lighting system was employed to ensure uniform subject illumination and minimize shadows on the green screen. Two softbox key lights were positioned at 45-degree angles to the subject from the front-left and front-right sides, providing even frontal illumination. A fill light was placed directly in front at a lower intensity to reduce harsh shadows on the face and body. Additionally, two backlights were aimed at the green screen to ensure even chroma saturation, and a rim light was placed behind the subject to separate the subject from the background and reduce color spill. All lights were diffused to prevent glare and maintain natural color rendering.

- Key Lights: Two softboxes with LED panels, each 100 W, placed at 45° angles to the subject (left and right), 2 m in front, at a height of 1.8 m, tilted downward at ~30°.
- Fill Light: 50 W LED panel with diffuser, directly in front of the subject, 2.5 m away, at chest height, providing ~50% intensity of the key lights to reduce shadow contrast.
- Back/Rim Light: A 60 W LED spotlight behind the subject, elevated at 2.2 m, pointing downward at 45°, creating a light outline around hair and shoulders to separate subject from the screen.
- Background Lights: Two 80 W LED panels directed at the green screen from each side, positioned 1 m from the screen edges, at a 45° inward angle. Light intensity adjusted to achieve uniform chroma saturation without exceeding 60% luminance.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

1.2 Makeup and costumes

Costumes and Clothing Selection

Costume design plays a critical role in ensuring accurate chroma keying and high-quality video capture. Subjects must avoid clothing that matches or contains shades close to the green screen, as this produces transparency artifacts during masking. Similarly, reflective fabrics (e.g., sequins, metallic textures, silk) should be avoided since they reflect green light back into the camera sensor, creating color spill and false edges. Optimal clothing consists of matte, solid colors that contrast strongly with the green screen, such as dark blues, reds, or neutral tones. For full-body capture, shoes should also be non-reflective and distinct from the green floor extension to prevent edge blending.

Makeup Considerations

Makeup application is important for facial data capture, particularly when expressions are part of the dataset. Glossy or shimmery cosmetics introduce unwanted highlights that interfere with consistent chroma keying and may produce variable skin tones across frames. A matte foundation helps create uniform skin texture and reduces light reflection. Eyeliner and eyebrow definition may improve landmark detection for facial expression analysis, while lip color should be chosen to contrast naturally with skin tone without oversaturation. Excessive makeup alterations should be avoided to maintain a natural baseline representation of the subject, unless variations in appearance are explicitly required by the dataset design.

Influence on Data Quality and Masking

Improper costume or makeup choices can lead to chroma spill, blurred segmentation edges, or unstable mask boundaries, reducing the accuracy of the extracted subject silhouette. These artifacts require additional post-processing, increasing the time and computational resources needed for dataset preparation. Conversely, when clothing and makeup are optimized for green screen environments, segmentation accuracy improves significantly, reducing false positives in masking and maintaining fine details such as hair strands or hand gestures. Proper preparation

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495.”

ensures the dataset is more consistent, reducing variability across subjects and enhancing the robustness of machine learning models trained on the video data.

Makeup and Costume Checklist for Green Screen Recording

Costumes / Clothing

- ✔ Wear matte fabrics in solid colors (dark blue, gray, red, or neutral tones).
- ✔ Ensure full-body clothing is non-reflective (including shoes).
- ✔ Choose outfits that contrast strongly with green backgrounds.
- ❌ Avoid green, neon, or pastel tones that resemble the screen.
- ❌ Avoid reflective or glossy materials (silk, sequins, metallics).
- ❌ Avoid complex patterns or stripes that cause visual aliasing.

Makeup

- ✔ Apply matte foundation to reduce shine and even out skin tone.
- ✔ Use natural, non-reflective lipstick or lip tint for mouth clarity.
- ✔ Light eyeliner and eyebrow definition to improve face tracking.
- ❌ Avoid glossy, glittery, or metallic makeup products.
- ❌ Avoid extremely bright or neon colors that could distort segmentation.

Accessories and Hair

- ✔ Keep hair neat and separated from the green screen (use subtle backlighting).
- ✔ Use matte hair products (avoid sprays or gels that create shine).
- ❌ Avoid reflective jewelry, glasses, or metallic accessories.
- ❌ Avoid hats, scarves, or loose items that obstruct facial capture unless required.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

1.3 Data processing

Preprocessing of Raw Video Data

The first step in processing is the conversion of raw camera footage into a standardized digital format suitable for frame-by-frame analysis. Videos are typically transcoded into lossless or minimally compressed formats (e.g., ProRes, DNxHR, or PNG/TIFF image sequences) to prevent degradation of chroma information. Metadata such as frame rate, resolution, and color space are preserved for reproducibility. Color correction may be applied at this stage to ensure uniform green screen saturation and eliminate variations caused by lighting.

Chroma Keying and Foreground Extraction

Chroma keying techniques are applied to separate the subject from the green screen background. This process involves identifying pixels within a defined chroma range corresponding to the green screen and removing or masking them. Advanced algorithms (e.g., spill suppression, color difference keying, or machine learning–based segmentation) are used to handle fine details such as hair, semi-transparent materials, and motion blur. The result is a binary or alpha mask representing the subject's silhouette in each frame. The mask is then combined with the original video to generate masked video frames, where only the subject remains visible and the background is transparent.

Post-processing and Mask Refinement

Post-processing ensures that the masked frames are accurate and consistent across the entire video sequence. Common refinements include edge feathering to smooth jagged boundaries, morphological operations (dilation/erosion) to correct mask leakage, and temporal smoothing to reduce flickering between frames. In cases where chroma keying fails due to clothing, lighting artifacts, or shadows, manual correction may be applied on a subset of frames, followed by interpolation to minimize workload. The final dataset consists of synchronized pairs of masked frames and their corresponding raw frames, stored in an organized format with metadata annotations. These processed frames form the foundation for training and evaluation of AI-based 2D avatars, ensuring high fidelity and minimal noise in downstream applications.

Step-by-Step processing pipeline

1. **Raw Video**
    - Input video with subjects and background (e.g., green screen).
2. **Chroma Keying**
    - Detect and separate the background using color-based keying.
    - Outputs preliminary foreground mask.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

3. **Mask Extraction**
    - Convert keying results to a binary mask (foreground vs. background).
    - May involve thresholding or alpha extraction.
4. **Refinement**
    - Clean edges, remove noise, smooth transitions (e.g., morphological operations, matting algorithms).
5. **Masked Frames**
    - Final output frames where the foreground is cleanly separated and ready for compositing or analysis.

2. Gaussian splatting data acquisition

Creating high-quality 3D AI-based avatars for virtual reality (VR) applications requires a careful balance between realism, computational efficiency, and user interactivity. One emerging technique, Gaussian splatting, represents complex 3D geometry and textures as a cloud of Gaussian primitives rather than traditional polygonal meshes. This approach allows for smoother rendering and easier integration of neural network-based enhancements, such as AI-driven facial expressions or dynamic clothing simulations. When developing VR avatars using Gaussian splatting, it is essential to define the purpose and context of the avatar, including whether it will be used in social VR, training simulations, or gaming, as these use cases impose different requirements on visual fidelity, animation complexity, and interaction responsiveness.

The first step in avatar creation involves data acquisition and preprocessing. High-quality 3D scans or multi-view images of the intended subject provide the foundation for Gaussian splatting representations. Data should be carefully cleaned and normalized to remove noise, correct lighting inconsistencies, and align the scans to a canonical pose. AI models, such as neural radiance fields or generative models, can then be trained to convert these inputs into Gaussian splatting clouds, optimizing for both geometric accuracy and realistic texture reproduction. Guidelines suggest maintaining a balance between the number of Gaussian primitives and rendering performance, as excessive detail can lead to VR latency issues.

Character design and aesthetic consistency are crucial when generating AI-based avatars. Designers should establish clear rules for proportions, style, and color palettes to ensure avatars remain visually coherent in the VR environment. Gaussian splatting enables subtle blending of surface features and realistic soft shadows, but these effects must be carefully calibrated to avoid uncanny valley artifacts. It is also important to consider modularity, allowing the avatar to adopt

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

multiple outfits, accessories, or hairstyles without requiring a complete rescan. AI-driven procedural generation can assist in creating variations while maintaining consistent visual style.

Animation and interactivity guidelines focus on integrating the avatar into a responsive VR environment. Gaussian splatting supports real-time deformation, making it suitable for facial expressions, hand gestures, and body movements driven by motion capture or AI pose estimation. Developers should prioritize efficient skinning and blending algorithms to ensure smooth motion, especially in multiplayer scenarios where multiple avatars are rendered simultaneously. Haptic feedback, gaze tracking, and voice-driven lip synchronization can further enhance the immersion, but must be optimized to prevent latency and motion sickness. Rigging and weight assignments should be tested thoroughly to ensure that Gaussian-based avatars deform naturally under a wide range of motions.

Optimization and deployment guidelines emphasize performance, scalability, and cross-platform compatibility. Gaussian splatting avatars can be computationally intensive, so developers should employ level-of-detail techniques, dynamic Gaussian culling, and hardware acceleration to maintain high frame rates in VR. Networked VR applications require compression strategies for avatar data while preserving visual fidelity, which may involve AI-based encoding or streaming of Gaussian primitives. Regular user testing is recommended to validate the visual realism, interactivity, and comfort of the avatars, ensuring they meet the immersive expectations of VR users. By following these guidelines, developers can leverage Gaussian splatting to create AI-powered avatars that are both visually compelling and operationally efficient in virtual reality environments.

2.1 Scene setup

Step 1: Define the Capture Space

1. Choose a **well-lit, uncluttered environment** to minimize shadows and background noise.
2. Ensure the area is **large enough for the subject to move** if you plan to capture multiple poses.
3. Mark the positions for cameras and the subject with tape or markers for **precise alignment**.

Step 2: Select and Prepare Cameras

1. Use **three high-resolution cameras** with adjustable focal lengths and manual exposure control.
2. Ensure all cameras support **synchronized triggering** or remote control to capture frames simultaneously.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

3. Mount the cameras on **tripods or fixed rigs** to prevent movement during capture.

Step 3: Arrange the Cameras

1. Position the **primary camera** directly in front of the subject to capture frontal details.
2. Place the **second camera at ~45° to the left** and the **third camera at ~45° to the right**, forming a semi-circular layout around the subject.
3. Ensure the cameras are at **similar height**, ideally at the subject's chest or eye level, to maintain consistent perspective.
4. Verify that each camera **overlaps slightly in their fields of view** to allow for accurate 3D reconstruction.

Step 4: Synchronize and Calibrate

1. Use a **synchronization signal or hardware trigger** to ensure all three cameras capture frames at exactly the same time.
2. Perform **camera calibration** using a checkerboard or calibration pattern to obtain intrinsic parameters (focal length, lens distortion) and extrinsic parameters (relative positions and angles).
3. Test a few frames to confirm **alignment, focus, and exposure** are consistent across all cameras.

Step 5: Capture Data

1. Have the subject stand in the marked position and maintain the desired pose.
2. Trigger all cameras simultaneously to capture **multiple angles at the same time**.
3. Repeat the process for **different poses or expressions**, ensuring consistent lighting and positioning.
4. Store the captured images or video sequences in **organized folders** labeled by pose, timestamp, or camera angle.

Step 6: Post-Processing Check

1. Verify that all frames are **sharp, properly exposed, and synchronized**.
2. Remove frames with motion blur or occlusions.
3. Prepare the data for **3D reconstruction** using Gaussian splatting or other neural rendering methods.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

2.2 Data processing

Creating a Gaussian-splatted 3D avatar begins with **capturing high-quality video** from multiple cameras or a moving single camera. Each video should cover the subject from various angles with consistent lighting and high resolution to preserve fine details. Preprocessing includes extracting frames, normalizing color and brightness, and segmenting the subject from the background to remove noise. This ensures that the subsequent 3D reconstruction accurately represents the subject without artifacts from shadows or background clutter.

Once the frames are ready, **camera calibration and pose estimation** are performed. Intrinsic parameters like focal length and lens distortion are determined for each camera, while extrinsic calibration defines their relative positions and orientations. Pose estimation identifies key landmarks on the subject, enabling accurate tracking across frames. These steps allow the 2D video data to be mapped into a coherent 3D space, which is crucial for generating a realistic avatar.

The core of the process is **Gaussian splatting-based reconstruction**. A dense point cloud is generated from the aligned frames, representing the surface of the subject. Each point is replaced by a Gaussian primitive that encodes position, color, and spatial uncertainty. Overlapping Gaussians are blended and smoothed to create a continuous 3D volume, avoiding the need for explicit mesh surfaces. AI-based refinement can further enhance texture detail, surface normals, and shading, resulting in a visually rich and realistic avatar suitable for VR applications.

Finally, the Gaussian avatar is prepared for **real-time VR integration**. Optimization steps include creating multiple levels of detail, compressing the number of Gaussians, and applying animation rigs for movement. The model is exported in a format compatible with the VR engine, preserving both geometric and color data. This pipeline transforms raw video footage into a fully functional, high-fidelity 3D avatar capable of dynamic interaction in immersive virtual environments.

Step 1: Camera Calibration and Pose Estimation

Before combining frames into a 3D representation, it's crucial to understand **where each camera is and how it views the subject**. This involves:

- **Intrinsic calibration:** Determines focal length, optical center, and lens distortion for each camera.
- **Extrinsic calibration:** Establishes the relative position and orientation of all cameras.
- **Pose estimation:** Detect keypoints or landmarks on the subject (e.g., joints, face landmarks) to track motion across frames.
  These steps allow the reconstruction algorithm to accurately map 2D pixels to 3D space.

Step 2: Multi-View 3D Reconstruction

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

Once frames are aligned in 3D space, the core Gaussian splatting process begins:

1. **Point cloud generation:** From multiple synchronized frames, generate a dense point cloud representing the subject's surface.
2. **Gaussian fitting:** Replace each point in the cloud with a Gaussian primitive, which encodes position, color, and uncertainty (size/shape).
3. **Blending and smoothing:** Merge overlapping Gaussians and apply smoothing to remove noise, producing a continuous 3D representation.

Gaussian splatting differs from mesh-based reconstruction because it doesn't require explicit surfaces; instead, **volumetric Gaussians approximate geometry and texture simultaneously**, making it ideal for neural rendering and dynamic lighting in VR.

Step 3: Neural Refinement and Texture Enhancement

AI can improve the visual fidelity of the Gaussian avatar:

- **Texture synthesis:** Neural networks can enhance surface detail, filling in regions that were poorly captured.
- **Normal and lighting inference:** Predict surface normals and reflectance for realistic shading.
- **Compression and optimization:** Reduce the number of Gaussians while preserving visual quality to maintain real-time VR performance.

At this stage, the avatar is **ready for animation and integration** in VR environments.

Step 4: Exporting and Integration

The final step involves preparing the Gaussian-splatted avatar for use in VR:

- **Level-of-detail (LOD) creation:** Generate multiple LODs to optimize rendering depending on the viewer's distance.
- **Animation rigging:** Apply skeletal or blend-shape rigs if real-time movement is needed.
- **File export:** Save the Gaussian splatted model in a format compatible with your VR engine, ensuring all positional, color, and shading data are preserved.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

Quality of data

Data quality is critical for generating accurate and realistic Gaussian-splatted 3D avatars. High-resolution, well-lit, and stable video or image captures provide the foundation for precise reconstruction. Inconsistent lighting, motion blur, or occlusions can introduce noise into the point cloud, leading to artifacts or incomplete geometry. Multi-angle coverage is essential, as gaps in the subject's surface can reduce the fidelity of the final avatar. Additionally, consistent color calibration across all cameras ensures uniform textures, preventing mismatched shading or color discontinuities in the 3D model. Proper background segmentation also contributes to clean data, reducing errors when the Gaussian splatting algorithm maps points to the subject's surface.

Beyond raw capture, **temporal and spatial consistency** in the data further improves avatar quality. Videos should be synchronized when using multiple cameras, ensuring that all frames represent the subject in the same pose. Camera calibration and pose estimation accuracy directly impact the alignment of 2D frames in 3D space, so even minor calibration errors can distort the avatar. Noise reduction, artifact removal, and careful preprocessing maintain data integrity, allowing AI-based refinement to enhance rather than compensate for flawed inputs. High-quality, consistent data ultimately leads to smoother Gaussian splatting, better texture reproduction, and more immersive VR avatars.

3. Semi-automatic scene creation based on existing historical sites

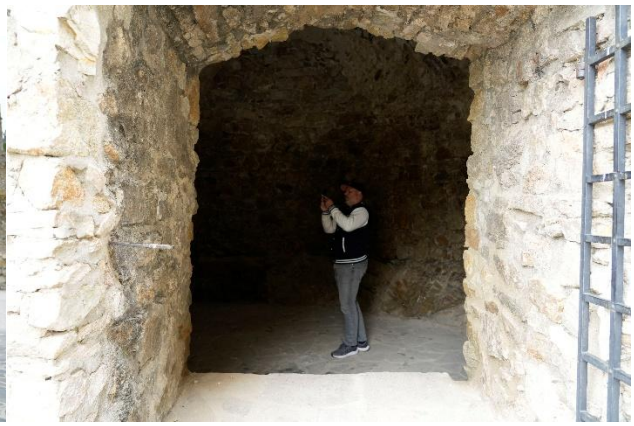
Semi-automatic scene creation leverages photogrammetry to reconstruct historical sites into highly detailed 3D models with minimal manual intervention. The process begins with image acquisition, where multiple overlapping photographs of the historical site are captured from various angles. High-resolution images with consistent lighting and coverage are essential to

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

capture fine architectural details. Drones or handheld cameras can be used to capture large or hard-to-reach areas. Tools like RealityScan (mobile) or RealityCapture (desktop) then automatically process these images to detect features, match points across images, and generate dense point clouds representing the surfaces of the site.

Once the initial point cloud is generated, the software provides semi-automatic alignment and optimization features. RealityCapture can automatically create meshes, optimize geometry, and generate textures, while allowing manual corrections where needed—such as filling gaps, removing noise, or refining misaligned sections. The semi-automatic workflow significantly reduces the labor-intensive steps of traditional 3D reconstruction, allowing users to focus on enhancing detail and correcting errors. The final output includes textured 3D meshes, which can be exported in standard formats for use in VR simulations, architectural studies, or historical preservation projects, effectively turning real-world historical sites into immersive, interactive digital environments.



Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495.”

4. Deepfake Avatars

4.1 Topic Rationale

Deepfake technology is one of the most prominent fields within artificial intelligence (AI), with a growing number of applications in both creative industries and content manipulation. This technology utilizes advanced deep learning methods to generate video or audio depicting fictitious events and statements. In its broadest sense, deepfake refers to the creation of synthetic media where an individual's likeness is superimposed onto another person's using sophisticated image processing algorithms. This is accomplished using Convolutional Neural Networks (CNNs), with a particular emphasis on Generative Adversarial Networks (GANs), which excel at creating hyper-realistic visual content.[1]

The primary objective of this thesis is to investigate the feasibility of developing a deepfake video chatbot using footage of an actor recorded against a green screen. This approach enables the creation of interactive avatars capable of realistically simulating human communication, which presents numerous opportunities for application in education, entertainment, digital marketing, and simulations. This work will detail the end-to-end process of deepfake video creation, including video segmentation, voice-driven lip synchronization, and the generation of a final video composite with an alpha channel for transparency. Particular focus is placed on creating a video chatbot—an avatar that leverages pre-recorded video for interactive communication. A case study will demonstrate the application of this process, enabling the dynamic synthesis of facial expressions and speech driven by audio input generated from user queries.

Deep learning, a subfield of AI, emulates the neural processing of the human brain for tasks like speech recognition, object detection, language translation, and decision-making. Among the most significant architectures in deep learning are Generative Adversarial Networks (GANs), introduced by Goodfellow et al. in 2014. GANs, which typically leverage CNNs as their core components, enable the generation of highly realistic images and videos, including the synthesis of novel photorealistic faces or the high-fidelity replacement of existing ones.[2]

4.2 Research Motivation

The development of real-time deepfake chatbots marks the next stage in the evolution of human-machine interaction, blurring the line between digital assistants and human interlocutors. By integrating Large Language Model (LLM) based response generation, Text-to-Speech (TTS) synthesis, and real-time facial animation, it is possible to create highly realistic virtual entities capable of engaging in fluid, human-like conversation. These innovations are poised to revolutionize sectors such as customer support, digital marketing, and education by offering more interactive and personalized user experiences. Research in this domain facilitates the

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

optimization of these systems for effective deployment and aids in identifying and addressing the challenges inherent in their implementation.

4.3 Primary Objectives and Research Plan

The primary objective of this research is to develop and evaluate a prototype real-time deepfake chatbot that integrates the aforementioned technologies: LLM-based response generation, speech synthesis, and real-time facial synthesis. The research is focused on understanding the technical capabilities and limitations of such a system and its potential for practical, real-world applications. Particular emphasis will be placed on system performance, the quality of user interaction, and the preservation of a transparent video background—a critical feature for seamless integration into diverse virtual environments.

The research plan is structured in three phases:

- **Phase One:** A theoretical analysis and a literature review of relevant technologies.
- **Phase Two:** The development of a functional prototype that integrates the selected components into a coherent system.
- **Phase Three:** Comprehensive testing and measurement of the system's efficiency, stability, and the quality of the user experience.

The findings will inform recommendations for the future development and application of such systems, emphasizing the need for safe and ethical deployment.

4.4 Expected Results

This thesis is expected to yield a detailed insight into the techniques for creating interactive deepfake avatars from green-screen video footage. The practical implementation and subsequent testing will demonstrate how a functional video chatbot, capable of realistic audiovisual communication, can be successfully implemented using the Wav2Lip model alongside other contemporary image and video processing tools.

5. Theoretical Framework

5.1 Historical Development

In the late 1990s and early 2000s, face detection emerged as a critical research area, driven by its potential military and security applications. During this period, military organizations and national security agencies heavily invested in facial recognition technologies to identify and track persons of interest. The objective was to develop tools for rapid and reliable facial recognition within massive datasets or in real time from sources like surveillance cameras and satellite imagery. However, the technology's application was not confined to the military; the commercial security sector also showed significant interest. There, face detection was applied to surveillance and

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

access control systems, such as unlocking devices or securing sensitive areas. Subsequent advances in machine learning and deep learning led to the development of highly sophisticated face detection algorithms. As deep learning techniques matured, the precision and reliability of face detection improved significantly. Today, this technology is widely accessible through open-source libraries and development tools, such as OpenCV and Python's face-recognition library.

This accessibility fostered the creation of various applications, including those for swapping faces in images, often with public figures and politicians. FaceApp is a prominent example of such an application. Generative Adversarial Networks (GANs) employ a unique learning paradigm distinct from other neural network architectures. A GAN consists of two competing neural networks: a Generator and a Discriminator. The Generator's objective is to create synthetic, realistic-looking images, while the Discriminator's task is to differentiate between these synthetic images and authentic ones. When both networks perform at a high level of proficiency, the resulting generated images can be indistinguishable from real photographs. Introduced by Ian Goodfellow et al. in 2014, GANs have achieved remarkable success. Their development was motivated by the observation that most neural networks are easily deceived into misclassification by the addition of minimal adversarial noise to the input data. Curiously, a network often exhibits higher confidence in its incorrect prediction after noise is introduced than it did in its original, correct prediction. This vulnerability often stems from models being trained on limited datasets, making them prone to overfitting and to learning nearly linear input-output mappings. In practice, the decision boundaries between data classes are complex and non-linear. Consequently, even minor perturbations to data points can lead to misclassification, highlighting the intricate nature of the problem.[2]

The term "deepfake" was coined in late 2017 by a Reddit user of the same name, who utilized TensorFlow, image search engines, social media, and public videos to create synthetic media. This work garnered significant public attention, and shortly thereafter, the first open-access tools were released, enabling users without expertise in deep learning to create their own deepfake videos.[3]

5.2 Applications and Types of Deepfakes

Deepfakes, most commonly known as manipulated videos featuring face swaps, have diversified significantly in recent years. This chapter details these various types along with their wide spectrum of applications, drawing from a comprehensive 2019 review by Kietzmann et al. that outlined the commercial uses of deepfake technologies. The key classifications from their work are summarized in Table 2.1.[4]

Type	Description	Application
Photo Deepfakes	**Face and Body Swapping**: Altering, replacing, or blending a person's face or body with another's.	Virtual try-on for cosmetics, eyewear, hairstyles, and apparel.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

Audio Deepfakes	**Voice Swapping**: Altering a voice or mimicking that of another person. **Text-to-Speech**: Modifying audio in a recording by supplying new text.	Creating diverse narrator voices for audiobooks (e.g., varying age, gender, dialect). Correcting mispronunciations or script changes in post-production without re-recording.
Video Deepfakes	**Face Swapping**: Replacing a person's face in a video with another's. **Face Transformation**: Seamlessly morphing one face into another. **Full Body Animation**: Transferring the bodily movements of one person onto another.	Placing an actor's face onto a stunt double's body for realistic action sequences. Allowing video game players to map their own faces onto game characters. Enabling public figures to conceal physical ailments during video appearances.
Audio-Visual Deepfakes	**Lip Synchronization**: Altering a speaker's mouth movements in a video to match a new audio track.	Dubbing advertisements and educational content into different languages while retaining the original speaker's voice.

Table 2.1: Types of Deepfakes, Their Descriptions, and Business Applications[5]

Below is an overview of key industries actively applying deepfake technology, highlighting its uses, benefits, and challenges.

1. Gaming and Film Industry

Deepfake technology offers a revolutionary alternative to traditional filming and animation in the film and gaming industries. Producers can generate realistic digital characters by manipulating existing actor footage or creating entirely synthetic ones. This innovation enhances creative freedom and production efficiency, enabling new forms of artistic expression.[6]

1.1 Chatbots

AI-driven chatbots are achieving increasingly human-like interactions, capable of expressing nuanced reactions like confusion or surprise that enhance the illusion of reality. This allows for the creation of non-player characters (NPCs) in games that can engage in spontaneous, natural conversations reflecting their unique personalities and emotions, providing players with a more immersive experience.

1.2 Actor Replacement and VFX

Deepfake technology can significantly lower production costs by providing an alternative to using stunt doubles or actor stand-ins. Instead of hours of filming, existing footage can be manipulated to meet new scene requirements. This technology facilitates the real-time generation of realistic facial animations synchronized to dialogue, creating vivid characters without laborious manual animation. The adoption of this technology is evidenced by studios like Lucasfilm, which has hired deepfake artists.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

2. Medicine

A notable medical application is the Revoice project, an initiative that uses deepfake audio technology to restore a personalized voice to patients who have lost the ability to speak due to conditions like throat cancer or neurological disorders. By training on prior recordings, the technology can synthesize a voice that retains the patient's unique tone, preserving their identity and facilitating communication. This project exemplifies the positive, assistive potential of deepfake technology.[7]

Furthermore, chatbots are used in medical training to simulate patient interactions, allowing healthcare professionals to practice and refine their diagnostic and communication skills. These chatbots can be programmed with diverse scenarios and patient responses, helping trainees improve their ability to ask pertinent questions, interpret symptoms, and provide effective support in a controlled environment. This is particularly beneficial for training new medical professionals and for the continuing education of experienced practitioners.[8]

3. Metaverse

In the metaverse, deepfake technology enables the creation of highly realistic digital avatars, fostering deeper user immersion in virtual environments. Major technology companies like Meta, Microsoft, and Nvidia are exploring deepfakes to enhance their respective metaverse platforms (Horizon Worlds, Mesh, and Omniverse). While this promises more engaging virtual interactions, it also introduces significant risks, including impersonation and the spread of misinformation, which could undermine user trust and security.[9]

The expanding application of deepfake technology across industries highlights its sophisticated and growing influence. It is being used to enhance user experiences, optimize processes, and create novel forms of interaction. While its full potential is still being explored, the role of deepfake technology is set to expand, presenting new opportunities alongside challenges that demand a responsible and ethical approach.

5.3 Literature Review

5.3.1 Speech2Vid

The Speech2Vid model is designed to generate a talking-head video from a single static image of a person and a corresponding audio segment of speech. The model inputs a target face image and an audio clip and outputs a video where the facial movements are synchronized with the audio. A key feature noted by the authors is that the input image does not need to be from the training dataset, making Speech2Vid generalizable to novel faces and speech.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

Figure 2.1: The Speech2Vid model generates a video of a person speaking, using a facial photograph and an audio recording of speech.[10]

Rather than mapping phonemes to visemes, the model's approach is to learn a direct correlation between raw audio and video data, focusing on speech segments and precisely defined facial regions.

The Speech2Vid architecture comprises four main components: two separate encoders for audio and image data streams, a decoder that generates image frames conditioned on the audio signal, and a CNN-based deblurring module to enhance output frame quality. The video is generated frame-by-frame, processing the audio in 0.35-second sequences.

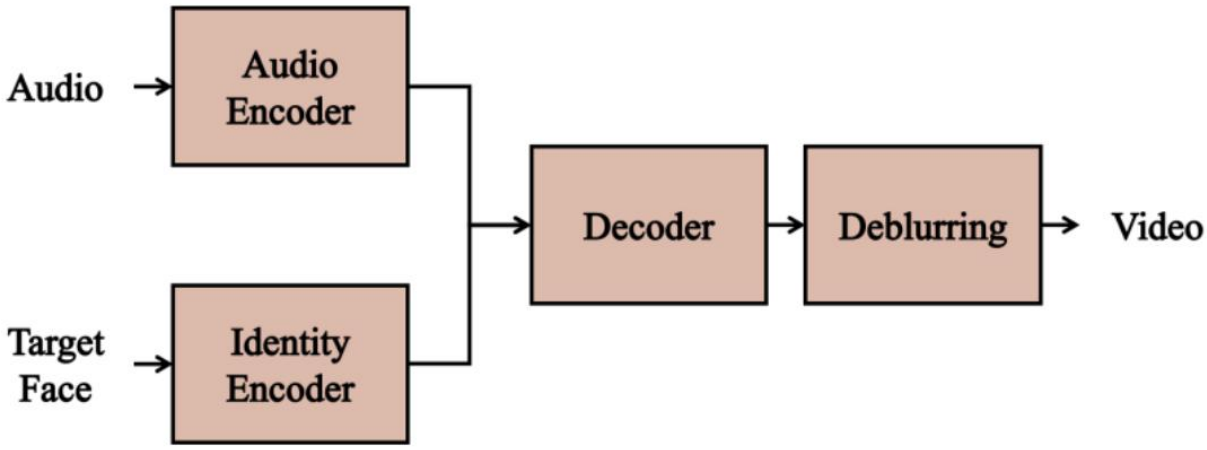


Figure 2.2: The model combines two encoders for audio and image, a decoder that generates images according to the sound, and a CNN module for sharpening the output frames.[10]

The audio encoder processes Mel-frequency cepstral coefficients (MFCCs) extracted from the raw audio. It uses 12 coefficients per sample, representing features on the Mel frequency scale. Each sample spans 0.35 seconds of audio at 100 Hz, yielding 35 time steps. This is represented as a 12×35 heatmap, where each column corresponds to the MFCC features at a specific time step. The encoder's architecture is a CNN based on AlexNet and VGG-M, with filter sizes adapted to the input dimensions. The identity encoder takes a static 112 × 112 × 3 image as input. To ensure the resulting identity vector captures features relevant to facial recognition, this encoder uses a VGG-M network pre-trained on the VGG Face dataset. The decoder receives the concatenated 256-dimensional feature vectors from the FC7 layers of both the audio and identity encoders. These vectors are progressively upscaled using transposed convolutions. The network also

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495.”

incorporates two skip connections, which merge encoder activations with decoder activations to preserve key identity features throughout the generation process.[10]

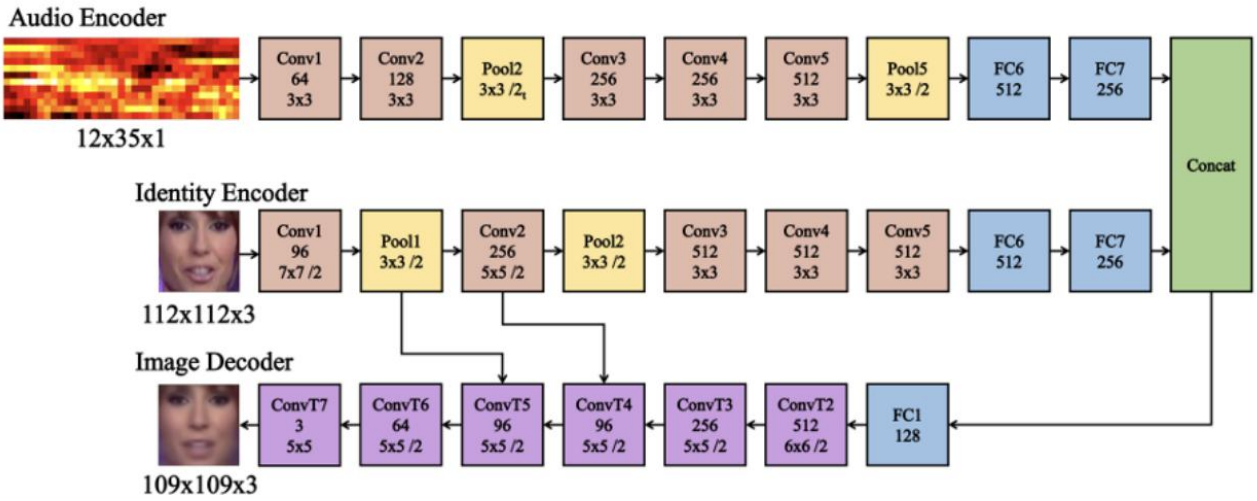


Figure 2.3: The three modules: an audio encoder, an identity encoder with a single static image as input, and an image decoder.[10]

5.3.2 Wav2Lip

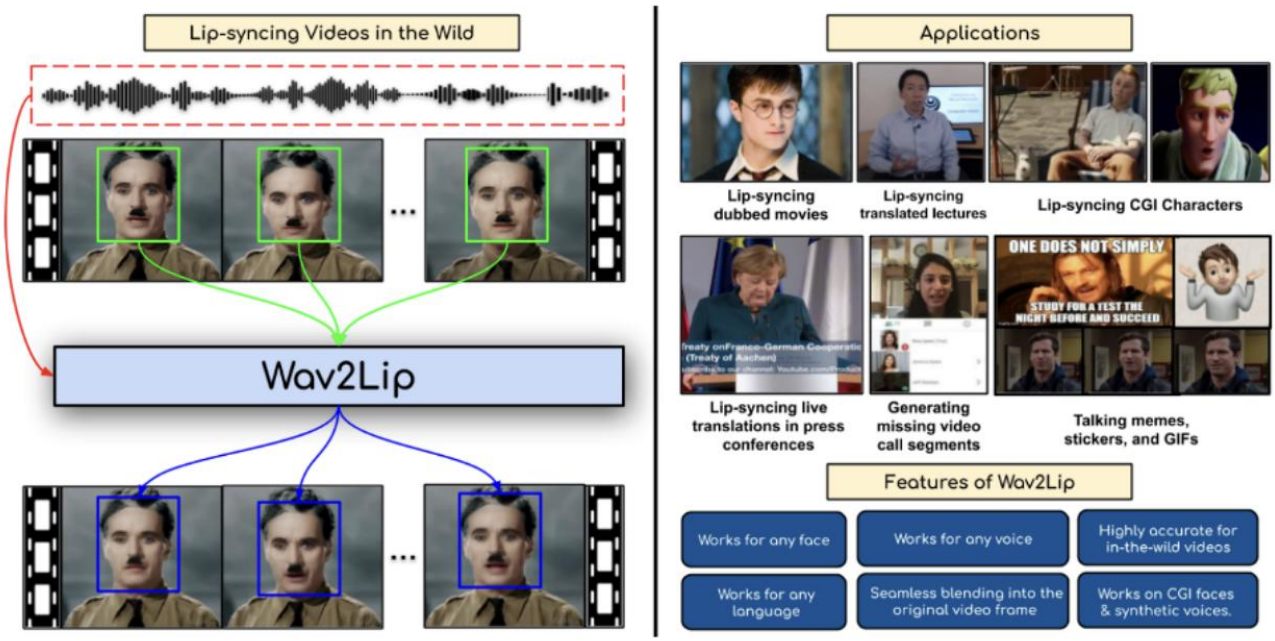


Figure 2.4: Wav2Lip enables precise lip synchronization, making it suitable for various practical applications.[11]

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

Wav2Lip is an advanced deep learning model for synchronizing lip movements with an arbitrary audio track. Unlike previous models that only penalized reconstruction loss, Wav2Lip employs a sophisticated discriminator that evaluates the quality of the lip-sync itself. Furthermore, the model is speaker-agnostic, making it universally applicable to any face in any video, regardless of identity or language.

The key contributions of this work are:

- The introduction of the Wav2Lip network, which achieves significantly higher accuracy and speed in lip synchronization compared to prior methods. It functions effectively on arbitrary speech and video without constraints.
- The development of a novel evaluation framework, including new metrics, to enable fair and robust assessment of lip-sync quality in unrestricted, "in-the-wild" videos.
- The creation of the ReSyncED dataset, a new benchmark designed to evaluate model performance on previously unseen subjects and scenarios.
- Wav2Lip is the first speaker-independent model to achieve lip-sync accuracy on par with real, synchronized video footage.

Human evaluations confirm the model's superiority, with subjects preferring videos generated by Wav2Lip over those from existing methods in over 90% of cases.[11]

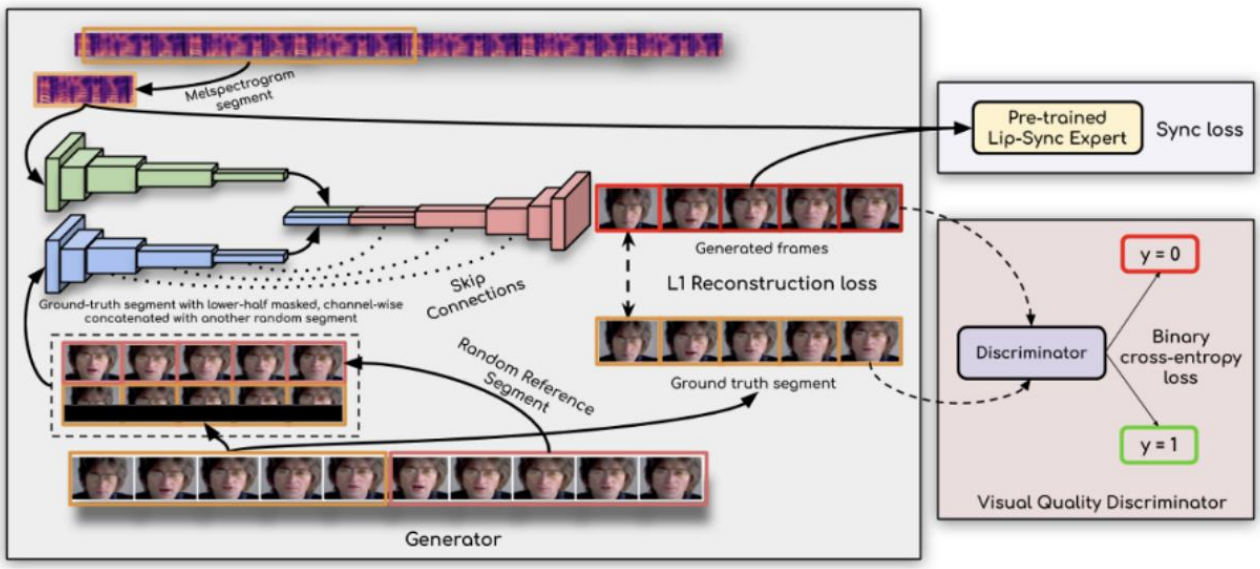


Figure 2.5: The model uses a pre-trained lip-sync discriminator and an additional visual quality discriminator to improve the precision of lip synchronization and the visual quality of the generated faces.[11]

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

6. Implementation

This chapter details the tools and procedures employed for the implementation of the deepfake chatbot. The focus is on key development phases to provide a clear insight into the technical aspects and challenges associated with creating a convincing and functional system.

6.1 Environment

The project was implemented within Google Colab to leverage its readily available computational resources. Project execution requires an Nvidia graphics card compatible with cuda 12.2; consequently, a Tesla T4 GPU was utilized. The Tesla T4, an Nvidia-developed GPU, is specialized for deep learning and artificial intelligence applications. It employs the Turing architecture, optimized for tasks such as image classification and tagging, video analysis, natural language processing, automatic speech recognition, and intelligent searching. The T4 GPU facilitates faster training and execution of deep neural networks, particularly when working with large models or resource-intensive data. Furthermore, the free, limited-use allocation of this resource within the Google Colab environment provided an excellent foundation for testing and developing the deep fake chatbot.[12]

6.2 Chatbot

6.2.1 Llama 3

The LLaMA 3 8b model was introduced in April 2024 by Meta. Although LLaMA 3 is neither multilingual nor multimodal, it represents a significant advancement over LLaMA 2. Improvements in post-training procedures have significantly reduced the rate of false refusals, increased the diversity of model responses, and enhanced reasoning capability, code generation, and adherence to user instructions. During development, in addition to standard benchmark tests, a new set of human evaluations was created, containing 1,800 different queries from 12 of the most common use cases, such as seeking advice, brainstorming, classification, coding, creative writing, extraction, and summarization.[13]

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495.”

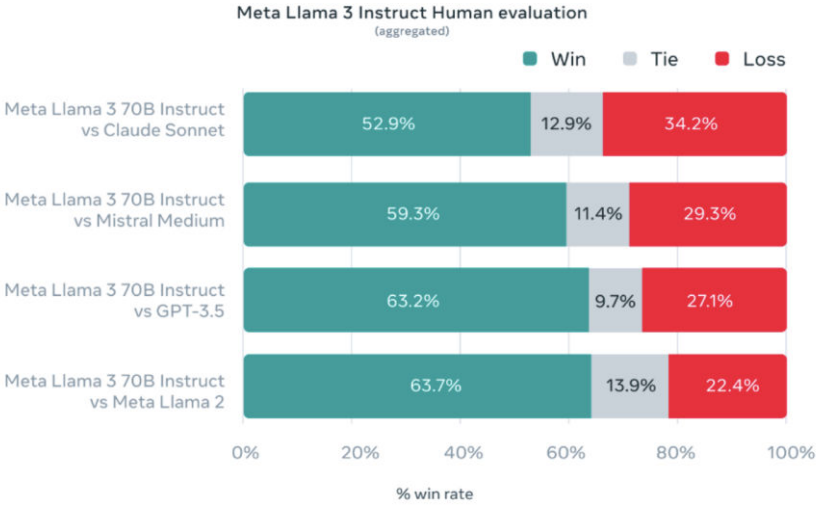


Figure 3.1: The graph shows the results of human evaluations relative to the Claude Sonnet, Mistral Medium, and GPT-3.5 models.[14]

6.2.2 Groq

The initial implementation step involved deploying a chatbot for generating textual responses to user questions. The Groq AI interface was selected due to its exceptional speed, support for various LLM models, and free API usage at the time of this writing. The API integration is straightforward, as demonstrated by the following code:

Program 3.1: Utilizing the Groq API to retrieve answers via the LLaMA 3 model

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

```
import os
from groq import Groq

# Set the API key
os.environ["GROQ_API_KEY"] = "API_KEY"

def groq_chat(question_input):
    client = Groq(
        api_key=os.environ.get("GROQ_API_KEY"),
    )

    chat_completion = client.chat.completions.create(
        messages=[
            {
                "role": "user",
                "content": "Give me answer in two sentences and no more than 20 words. " +
question_input,
            }
        ],
        model="llama3-8b-8192",
    )

    return chat_completion.choices[0].message.content
```

The code establishes the API key and client, then sends the query to the API. Parameters include the role, the content (prompt), and the model. The function returns the generated text response content. The prompt explicitly prepends the user's question with a constraint: "Give me answer in two sentences and no more than 20 words." This constraint was implemented to ensure the response is concise and clear, thereby accelerating audio track creation, reducing the final video duration, and ultimately yielding a faster result. The API and model exhibited excellent performance, with response generation time being nearly negligible. Although the LLaMA 3 model is not designated as multilingual, tests were conducted in the native language; however, the responses were grammatically and coherently unsatisfactory. Therefore, we proceeded exclusively with the English language.

6.3 Text-to-Speech Conversion

The Deepgram API was utilized for converting text to speech, as shown below:

Program 3.2: Generating an audio recording from text using the Deepgram text-to-speech API[15]

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495.”

```python
from deepgram import DeepgramClient, SpeakOptions

DEEPGRAM_API_KEY = "API_KEY"

def text_to_speech(answer_text):
    TEXT = {
        "text": answer_text
    }
    FILENAME = "audio.mp3"

    try:
        deepgram = DeepgramClient(DEEPGRAM_API_KEY)

        options = SpeakOptions(
            model="aura-arcas-en",
        )

        response = deepgram.speak.v("1").save(FILENAME, TEXT, options)

    except Exception as e:
        print(f"Exception: {e}")
```

The text_to_speech function receives the text (the answer from the previous step), creates a client, sets supplementary options such as the synthetic voice model, and calls the API. Upon successful generation, the response is saved as an MP3 file named **"audio.mp3"**. Since a male actor is involved, the **aura-arcas-en** model was selected to generate a male voice, chosen from a dozen available models. The **aura-arcas-en** model is specialized exclusively for the English language, making it the appropriate choice given the English text input. Furthermore, the speed of text-to-speech conversion is optimized to the level of real-time conversation, ensuring this step does not introduce significant latency into the final response.

6.4 Web Service Implementation

6.4.1 Flask

Flask is a micro web framework written in the Python programming language that enables fast and easy development of web applications. It is considered a microframework because it does not require special tools or libraries and does not include built-in modules for database operations and validation, instead relying on extensions to add these functionalities. Its modular architecture facilitates the simple addition of extensions for authentication, databases, Object-Relational Mappers (ORMs), and form validation. Developed as a lightweight alternative to larger frameworks like Django, Flask provides flexibility and simplicity for building APIs and dynamic web pages. It adheres to the principle of minimalism, making it ideal for small and medium-sized projects requiring flexibility without excessive complexity.[16]

Flask is based on several key components:

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

- **Werkzeug** – A library for WSGI applications providing objects for handling requests and responses.
- **Jinja** – A powerful template engine enabling the dynamic generation of HTML pages.
- **MarkupSafe** – A library that ensures safe handling of strings and prevents XSS attacks.
- **ItsDangerous** – A tool for secure data serialization, enabling secure storage of sessions within cookies.

In this project, Flask is used to create an API that provides user access to video files hosted in Google Colab. Two endpoints were created to retrieve video files. The first endpoint, /api/video, uses the Flask function send_file to return the complete video file (output_transparent.webm) in WebM format upon an HTTP GET request, allowing for direct download. The second endpoint, /api/stream_video, uses the Flask function Response and enables video streaming. Instead of returning the entire file, this endpoint sends the video in segments (chunks) of 1 MB, allowing users to begin playback as soon as the first segment arrives, optimizing performance for large video files. Flask served as the backend to manage access to video content from Colab using these two distinct methods.

Program 3.3: Example of a Flask API for downloading and streaming a video file in WebM format

```
from flask import Flask, send_file, Response
from flask_cors import CORS

# Create Flask app
app = Flask(__name__)
CORS(app)

# API that returns a video file
@app.route('/api/video', methods=['GET'])
def get_video():
    return send_file("/content/output_transparent.webm", mimetype="video/webm")

# New endpoint that streams the video
@app.route('/api/stream_video', methods=['GET'])
def stream_video():
    def generate():
        with open("/content/output_transparent.webm", "rb") as video_file:
            while chunk := video_file.read(1024 * 1024):  # Reads 1MB at a time
                yield chunk

    return Response(generate(), mimetype="video/webm")
```

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

6.4.2 Ngrok

Ngrok is a globally distributed reverse proxy that exposes a local server to the internet via secure tunneling, thereby eliminating the need for complicated port forwarding configuration. In addition to connecting private networks to the internet, ngrok combines the functionalities of a reverse proxy, load balancer, API gateway, DDoS attack protection, and data transfer acceleration. Ngrok facilitates testing webhooks, allows sharing local web pages, and enables testing mobile applications with a local backend. It is also used for securely connecting APIs in external networks, on IoT devices, and in development environments. With Ngrok, temporary, publicly accessible URLs for local servers can be quickly generated, simplifying project sharing and integration with services that require online access. Ngrok also provides options for authentication, encryption, and traffic inspection.[17]

Program 3.4: Starting the ngrok tunnel to expose the local Flask server via a public URL address

```
from pyngrok import ngrok
import os

# Set your ngrok auth token
NGROK_AUTH_TOKEN = ""
ngrok.set_auth_token(NGROK_AUTH_TOKEN)

# Start ngrok tunnel on port 5000
public_url = ngrok.connect(5000).public_url
print(f"Public URL: {public_url}")

# Start the Flask server
app.run(port=5000)
```

The code connects the local Flask server to the public internet using ngrok. This involves initializing ngrok authentication with ngrok.set_auth_token(NGROK_AUTH_TOKEN), followed by starting a tunnel to local port 5000 via ngrok.connect(5000).public_url. The Flask server is then started on the same port, and the access address is printed to the console. This method allows the application running locally in Google Colab to be accessible from the internet without manual network configuration.

```
Public URL: https://f559-34-82-38-110.ngrok-free.app
 * Serving Flask app '__main__'
 * Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug:127.0.0.1 - - [16/Feb/2025 13:52:36] "GET /api/stream_video HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [16/Feb/2025 13:53:05] "GET /api/stream_video HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [16/Feb/2025 13:53:18] "GET /api/video HTTP/1.1" 200 -
```

Figure 3.2: Display of the Flask application log using ngrok to enable public access to the local server.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

During testing, this approach proved efficient for sending video content in WebM format. Furthermore, the alpha component (transparency) was successfully retained during video transmission.

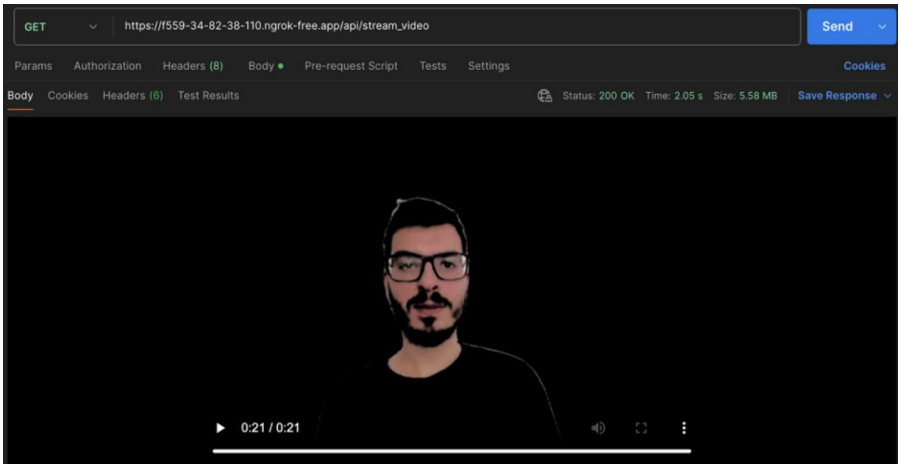


Figure 3.3: Display of the response obtained from the /api/stream_video endpoint.

6.5 WebM Format

The desired input and output from the model is a video in WebM format. While the model supports both MP4 and WebM as input, modifications were necessary within the Wav2Lip model to handle the output. The primary reason for this change was the requirement for a video that includes an alpha component to enable background transparency, facilitating integration into environments where transparency is essential. WebM with the VP9 codec and alpha channel support was identified as the optimal solution.

Key changes were required in the **inference.py** script to adjust the output video generation and encoding method. Additionally, minor adjustments were made to the **run.py** script to correctly define and process the resulting file. The first change redefined the output file path. In **inference.py**, the argument **-outfile** was adjusted to default to the WebM extension instead of the preset MP4 format.

Program 3.5: Changing the default output video extension from .mp4 to .webm

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495.”

```
# Original code
parser.add_argument(
    "--outfile",
    type=str,
    help="Video path to save result. See default for an e.g.",
    default="results/result_voice.mp4",
)

# Modified code
parser.add_argument(
    "--outfile",
    type=str,
    help="Video path to save result. See default for an e.g.",
    default="results/result_voice.webm",
)
```

The second required change is in the code segment that iterates through the frames and mel spectrograms in batches, using tqdm for the progress bar. This process prepares the data for the model by grouping images and corresponding mel spectrograms into batches. Mel spectrograms illustrate the visual representation of sound in the frequency domain, emphasizing the lower frequencies on the non-linear mel scale. In the Wav2Lip model, the mel spectrogram is used to analyze the rhythm and intonation of the input speech to synchronize the lip movements in the final video.[18]

Upon forming the first batch, quality parameters (including mask size and feathering) are set, and VideoWriter is initialized to create the output video using the mp4v codec and save it as an MP4 file in "temp/result.mp4".

Program 3.6: Processing input frames and mel spectrograms in batches, and creating the output video using VideoWriter[19]

```
for i, (img_batch, mel_batch, frames, coords) in enumerate(
    tqdm(
        gen,
        total=int(np.ceil(float(len(mel_chunks)) / batch_size)),
        desc="Processing Wav2Lip",
        ncols=100,
    )
):
    if i == 0:
        if args.quality != "Fast":
            print(
                f"mask size: {args.mask_dilation}, feathering: {args.mask_feathering}"
            )
        if args.quality != "Improved":
            print("Loading", args.sr_model)
            run_params = load_sr()
        print("Starting...")

        frame_h, frame_w = full_frames[0].shape[:-1]
        fourcc = cv2.VideoWriter_fourcc(*"mp4v")
        out = cv2.VideoWriter("temp/result.mp4", fourcc, fps, (frame_w, frame_h))
```

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495.”

An initial attempt was made to change the output video format from MP4 to WebM using the **VP9 codec**:

Program 3.7: Setting parameters for recording video in WebM format using the VP9 codec

```
frame_h, frame_w = full_frames[0].shape[:-1]

# VP9 codec for WebM format
fourcc = cv2.VideoWriter_fourcc(*"VP90")
out = cv2.VideoWriter("temp/result.webm", fourcc, fps, (frame_w, frame_h))
```

Testing with the **VP8 codec** (VP80) was also unsuccessful, resulting in the following error for both codecs:

Program 3.8: Lack of support for the VP9 codec

```
OpenCV: FFMPEG: tag 0x30395056/'VP90' is not supported with codec id 167 and format 'webm / WebM'
```

This error indicated that OpenCV with FFmpeg did not directly support generating WebM video with the **VP9 codec** via **cv2.VideoWriter**, a limitation that could not be resolved within the current FFmpeg version's codec support.

A condensed portion of the subsequent code within the same loop is shown below:

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495.”

Program 3.9: Processing and integrating generated frames into the original video

```
with torch.no_grad():
    # Model generates predictions based on input data (mel spectrogram and image)
    pred = model(mel_batch, img_batch).cpu().numpy().transpose(0, 2, 3, 1) * 255.0

    # Iterate through predictions, original frames, and ROI coordinates
    for p, f, c in zip(pred, frames, coords):
        y1, y2, x1, x2 = c  # Region of Interest (ROI)

        # If mask debugging is enabled, convert frame to grayscale and back to BGR
        if str(args.debug_mask) == "True":
            f = cv2.cvtColor(f, cv2.COLOR_BGR2GRAY)
            f = cv2.cvtColor(f, cv2.COLOR_GRAY2BGR)

        # Resize prediction to match ROI dimensions
        p = cv2.resize(p.astype(np.uint8), (x2 - x1, y2 - y1))
        cf = f[y1:y2, x1:x2]

        # Apply super-resolution if enhanced quality is selected
        if args.quality == "Enhanced":
            p = upscale(p, run_params)

        # Create mask for better integration into the original frame
        if args.quality in ["Enhanced", "Improved"]:
            if str(args.mouth_tracking) == "True":
                p, last_mask = create_tracked_mask(p, cf)
            else:
                p, last_mask = create_mask(p, cf)

        # Insert enhanced prediction into the original frame
        f[y1:y2, x1:x2] = p

        # If preview settings enabled, display and possibly save results
        if str(args.preview_settings) == "True":
            cv2.imwrite("temp/preview.jpg", f)
            if not g_colab:
                cv2.imshow("preview - press Q to close", f)
                if cv2.waitKey(-1) & 0xFF == ord('q'):
                    exit()
            else:
                out.write(f)  # Write frame to output video
# Cleanup: close windows and release video writer
cv2.destroyAllWindows()
out.release()
```

This section shows that each generated frame is adapted and inserted into the corresponding Region of Interest (ROI) within the original video, along with the application of supplementary image quality enhancement techniques.

To circumvent the unsupported codec for **WebM video** with **cv2.VideoWriter**, the code was modified to save each processed frame as a separate **PNG image**. This approach enables the

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495.”

subsequent recreation of the **WebM video** and addresses the requirement for preserving **background transparency**, as the **PNG format** supports the **alpha channel**.

Program 3.10: Saving frames as PNG images for subsequent processing and preservation of transparency

```
frame_filename = os.path.join(frames_dir, f"frame{frame_counter:04d}.png")
cv2.imwrite(frame_filename, f_bgra)
frame_counter += 1
```

Following frame processing, the final output video creation stage begins. The original code uses FFmpeg to merge the temporary video recording (temp/result.mp4) and the audio file (args.audio) into the final video, using the "libx264" codec suitable for MP4 format.

Program 3.11: Creating the final MP4 video by merging image and sound[19]

```
if str(args.preview_settings) == "False":
    print("Converting to final video")

    subprocess.check_call([
        "ffmpeg",
        "-y",
        "-loglevel", "error",
        "-i", "temp/result.mp4",
        "-i", args.audio,
        "-c:v", "libx264",
        args.outfile
    ])
```

The third and final change in the **inference.py** script modified this segment to generate WebM video. Since individual **PNG frames** from the **"temp/frames"** directory are now used instead of a temporary video file, **FFmpeg** is called with the **-framerate** option. The video codec is set to **"libvpx-vp9"** for the **WebM format**, and the **-pix_fmt** option is set to **"yuva420p"** to ensure the preservation of the **alpha channel**. The parameter **"1M"** sets the bitrate to 1 Mbps.

Program 3.12: Modified code for creating the final WebM video

```
ffmpeg_cmd = [
    "ffmpeg",
    "-y",
    "-framerate", str(fps),
    "-i", os.path.join("temp", "frames", "frame%04d.png"),
    "-i", args.audio,
    "-c:v", "libvpx-vp9",
    "-pix_fmt", "yuva420p",
    "-b:v", "1M",
    args.outfile
]

subprocess.check_call(ffmpeg_cmd)
```

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

With these modifications in **inference.py**, minor adjustments were still required within the **run.py** script and the **Easy_Wav2Lip_v8_3.ipynb** notebook to test the results.

The first location requiring an extension change is where the path to the temporary route is created within the **run.py** script.

Program 3.13: Defining the path for the temporary output file

```
temp_output = os.path.join(working_directory, "temp", "output.mp4")
```

The second location where .mp4 must be replaced with webm is the code segment that formats the output video file path, combining the folder, file name, suffix, and extension, followed by path normalization (normpath).

Program 3.14: Generating the absolute path to the output video (.mp4)

Python

```
output_video = os.path.join(folder, output_filename + output_suffix + ".mp4")
output_video = os.path.normpath(output_video)
```

No further changes were needed in this script. Within the **Easy_Wav2Lip_v8.3.ipynb** notebook, the output video extensions must also be replaced in the same manner as in the **run.py** script to correctly display the WebM format output.

After executing the transformations, testing the model with a WebM format input video confirmed that the generated output video was also in the same format. However, the alpha component was not preserved, despite accounting for this criterion in the modified code sections.

Figure 3.4: On the left is a frame of the input video in WebM format with an alpha component, while on the right is the output WebM video, which is also in WebM format, but without the alpha component.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

6.6 Background Transparency

6.6.1 Methodology

Following the code modifications that enabled WebM output, the video transparency was not retained, indicating that the alpha component was lost during processing. Further analysis concluded that the issue stemmed from how the OpenCV library processes and writes video files. Specifically, the functions used for reading, processing, and recording video either do not support the alpha channel or discard it during the conversion process.

When a video is loaded using cv2.VideoCapture in OpenCV, it is loaded in BGR (Blue-Green-Red) format. If the video contains transparency, the alpha channel is lost by default. OpenCV uses BGR instead of RGB, and when converting to a NumPy array, any channel not part of the BGR format (including the alpha channel used for transparency) is ignored.

Program 3.15: Loading and preparing frames from a video file

```
# Check option for changing video resolution
if args.fullres != 1:
    print("Resizing video...")

# Open video file using OpenCV VideoCapture
video_stream = cv2.VideoCapture(args.face)

# Get frames per second (FPS) from the video file
fps = video_stream.get(cv2.CAP_PROP_FPS)

# List to store all video frames
full_frames = []

# Loop to read frames from the video file
while True:
    still_reading, frame = video_stream.read()
    if not still_reading:
        video_stream.release()
        break

    # If full resolution option is off, resize frame accordingly
    if args.fullres != 1:
        aspect_ratio = frame.shape[1] / frame.shape[0]
        frame = cv2.resize(frame, (int(args.out_height * aspect_ratio), args.out_height))

    # Rotation and cropping code would go here (omitted as not relevant for alpha channel)

    # Append processed frame to full_frames list
    full_frames.append(frame)
```

To preserve the alpha channel in videos using the RGBA format, it is necessary to bypass the OpenCV library and utilize alternatives such as Pillow for images or PyAV for video, which offer

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

greater control over color formats and channels. The initial alternative was to use the AV library instead of OpenCV.

Program 3.16: Loading video frames using the PyAV library

```
import av
import numpy as np

# Open video container using PyAV
container = av.open(args.face)

# Get average frame rate from the video stream
fps = container.streams.video[0].average_rate

# Iterate over frames decoded from the video stream
for frame in container.decode(video=0):
    # Convert frame to a NumPy array in BGRA format
    frame = frame.to_ndarray(format="bgra")
```

The implemented code uses av.open() to open the video file and then frame.to_ndarray(format="bgra") to attempt to preserve the alpha channel in each frame. However, this resulted in the error: malloc_consolidate():invalid chunk size, indicating a memory issue often associated with improper memory management during the decoding of large video files or high frame counts. Testing confirmed that the issue was specific to the bgra channel, as conversions using other non-alpha formats proceeded without error.

6.6.2 Performance and Optimization

Having become familiar with the potential drawbacks of the background removal approach, we will now focus on processing speed. The speed of background removal is a crucial factor in our work because it requires processing large quantities of images within the shortest possible timeframe. A set of 954 frames will be used for testing, where the green background is clearly emphasized. Since the green color can be easily isolated and there is a clear distinction between the object and the background, testing the efficiency of the background removal process with high precision is greatly simplified.

For the previously mentioned code, the frame processing speed amounted to 203.44 seconds. This procedure, which encompasses loading each image, background removal, and saving the results in PNG format, is quite time-consuming despite its simplicity. The opportunity for enhancement lies precisely in changing the sequential method—which means each frame is processed one by one—to parallelization.

Program 3.18: Parallel Image Processing

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495.”

```
# Process all images in the input directory in parallel
with ThreadPoolExecutor() as executor:
    # Create a list of all PNG images in the directory
    image_files = [f for f in os.listdir(input_frames_dir) if f.endswith('.png')]

    # Parallel processing
    executor.map(remove_background_and_save, image_files)
```

We implemented parallelization using ThreadPoolExecutor, which significantly improved processing speed. This approach enables the parallel execution of tasks, instead of a sequential process, allowing for the simultaneous processing of multiple images using different threads. This method of parallelization allows for better utilization of available computational resources, thereby shortening the total processing duration. For testing, we used a T4 GPU in Google Colab, which further accelerated image processing operations. The T4 GPU is extremely efficient for tasks involving high parallelization, such as image and video content processing, because it utilizes an advanced architecture with CUDA cores specialized for parallel data processing. We placed the image processing code within the process_image function, which allows for its multiple calls through the parallel process. In this way, multiple images are processed simultaneously, independently of other images, which speeds up the overall processing process. As a result of this optimization, we achieved an improved time of **144.24 seconds** for processing the image set, which represents a significant improvement over sequential processing.

Although the first method, by adding parallelization, brought significant improvements in processing speed, we decided to try a second approach using the OpenCV library. In this approach, the image is first loaded and converted into the HSV (Hue, Saturation, Value) color space, which allows for easier extraction of specific colors, in our case, the green background. Next, a color range corresponding to the green background is defined, and a mask is created that selects all pixels within that range. By mask inversion, we obtain the non-green area, and that mask is then applied to the original image. After background removal, the result is saved in BGRA format, which includes the alpha channel, enabling background transparency.

Program 3.19: Frame Processing Using the OpenCV Library

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

```python
import os
import numpy as np
import cv2

def process_image(filename):
    input_path = os.path.join(input_folder, filename)
    im = cv2.imread(input_path)

    # Convert image to HSV color space
    hsv = cv2.cvtColor(im, cv2.COLOR_BGR2HSV)

    # Color range for green background
    lower_green = np.array([35, 50, 50])   # Lower bound
    upper_green = np.array([85, 255, 255]) # Upper bound

    # Create a mask for the green color
    mask = cv2.inRange(hsv, lower_green, upper_green)

    # Invert the mask to keep non-green areas
    mask_inv = cv2.bitwise_not(mask)

    # Apply the inverted mask to the original image
    result = cv2.bitwise_and(im, im, mask=mask_inv)

    # Convert to BGRA and set the alpha channel using the inverted mask
    result_with_alpha = cv2.cvtColor(result, cv2.COLOR_BGR2BGRA)
    result_with_alpha[:, :, 3] = mask_inv

    # Save the resulting image
    output_path = os.path.join(output_folder_no_bg, filename)
    cv2.imwrite(output_path, result_with_alpha)

# Process all PNG images in the input folder
image_files = [f for f in os.listdir(input_folder) if f.endswith('.png')]

for filename in image_files:
    process_image(filename)

print("Finished processing all images.")
```

By using the OpenCV method for background removal, the processing time was significantly reduced from **144.24 seconds** to just **44.5 seconds** compared to the previous approach. This method proved to be very efficient; however, to further accelerate the process, we decided to

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495.”

implement parallelization, similar to the previous example. After applying parallelization, the processing time was further reduced to **31.41 seconds**. Although the processing time was significantly reduced, the image quality remained equally good and satisfactory, thereby achieving an optimal balance between processing speed and quality.

Figure 3.5: Example of Processed Images

The final approach, which involves using the OpenCV library for background removal with the implementation of parallelization, proved to be the most efficient for our needs, and therefore this method will be used in further work and all future applications.

6.7 Analysis of Time-Consuming Components

In this section of the paper, we will focus on the slowest parts of the process to identify the potential for optimization and shorten the total processing time. Although we extended the initial execution time in the previous section by performing background removal, we have nevertheless found an optimal process for preserving transparency, which has clear application and justification in real-world examples.

A second location where a certain slowdown was observed is within the inference.py script. Namely, in the initial part of the script, various libraries are imported, some of which are more resource-intensive, consequently leading to additional time consumption with every execution.

Program 3.20: Library Imports within inference.py[19]

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

```
print("\rloading torch", end=""); import torch

print("\rloading RetinaFace", end=""); import batch_face as RetinaFace

print("\rloading Image", end=""); from PIL import Image

# ...

print("\rloading load_model", end=""); from easy_functions import load_model, g_colab
```

Although the intention is for these imports to execute only once, the inference.py script is unfortunately called directly from run.py, which means that every new execution functions as a completely separate process with its own context and memory. Consequently, all modules are reloaded upon every execution, with no caching between sessions.

In the conducted experiments, we observed that a significant portion of the total execution time is also consumed by the face detection process on each individual frame, which is particularly pronounced with longer videos. This detection is performed by the face_rect function, which processes images in small groups (batches), uses the predefined detector model to find faces, and returns the face coordinates if the detection is successful. The system attempts to find a face for every frame, and if none is detected in a given frame, the result from the previous frame is used (the so-called fallback mechanism).

To optimize this time-consuming process, a newer version of the model offers the use_previous_tracking_data option, which allows previously saved face detection data (bounding boxes and face positions per frame) to be reused. This avoids re-executing detection during subsequent runs with the same input video. Using this option significantly shortens the model's total execution time, as it skips the most intensive part of the processing-multiple face detection.

6.8 Analysis of Execution Time

For the purpose of **performance measurement**, the execution time of the entire Wav2Lip module will be analyzed, excluding the phases of response generation and its conversion from text to speech. Testing will be conducted using identical input data and the same query, thereby ensuring the consistency of the results. The measurement will be repeated five times, and the average of all obtained results will be used as the final value, providing a more reliable estimate of the module's efficiency.

The first performance measurement will be conducted using an MP4 video lasting **13 seconds** and an audio file of speech lasting **6 seconds**. In this case, the basic version of the Wav2Lip model will be used, without the application of an additional mask, which represents the most basic and also the fastest model configuration. This measurement will serve as a reference point for comparison with other model variations and settings that include additional processing. The result obtained with the specified settings was **34.09 seconds** for complete processing.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

To examine the influence of input data length on execution time in more detail, we will conduct additional measurements below with shorter and longer videos, as well as shorter and longer audio files. First, we will focus on the impact of the audio file duration on execution time. We will perform measurements using two audio files of different lengths, one lasting **3 seconds** and the other **9 seconds**, while the video will remain unchanged. The measurement results show that for the shorter audio recording, the execution time was **22.55 seconds**, while for the longer audio recording, the execution time increased to **46.89 seconds**. These results indicate an express increase in execution time with the increase in audio input duration, which is expected, as this component determines the length of the final video.

Next, we conducted an additional measurement with the aim of examining the influence of video duration. Videos lasting **10** and **16 seconds** were used, while the audio file remained constant at **6 seconds**. The obtained results showed that the execution time for the longer video was **34.6 seconds**, while the result for the shorter video was **33.71 seconds**. Given that an execution time of **34.09 seconds** was recorded in the initial measurement, we can conclude that the video duration does not play a significant role in the total execution time. This is also the expected result, since only those frames corresponding to the audio duration are processed.

The next item we will investigate is the duration of the process when previously saved data for face detection is used. It is important to note that in this approach, the data for the longest audio recording must be saved first, otherwise not all necessary frames will be available, which can result in an execution error. In the testing, the same audio files of **3, 6**, and **9 seconds** were used again, as well as a video recording lasting **10 seconds**. The obtained execution results were, respectively: **18.52, 27.17**, and **35.12 seconds**. If we represent these results graphically, we can observe that the execution time increases almost linearly with the audio recording length. This indicates a proportional dependence between the length of the input data and the total processing duration.

During testing in the Google Colab environment, outliers appeared in individual measurements, most likely caused by the variable availability of system resources at a given moment. Such cases were not taken into account during the analysis to avoid skewing the results. Therefore, it is desirable to perform a larger number of measurements under similar conditions to obtain a more reliable estimate of the average execution time.

Now we will examine whether the size of the face in the frame, or the fact that the face is not in the video foreground, affects the processing speed, as well as the very possibility of model execution. In other words, we are testing how the model behaves in situations when the face is distant from the camera or occupies a smaller part of the frame, which may make face detection difficult or impossible. In this testing, we will use two videos, both lasting **10 seconds**, one with a clearly highlighted face in the foreground, and the other with a significantly smaller face in the background. In both cases, we will use the same audio recording lasting **6 seconds** so that the other input parameters are identical and we can clearly compare the influence of face position

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

and size on model performance. The execution time for the video where the face is in the foreground has already been recorded during previous testing, and it amounts to **27.17 seconds**. Since we used the same video recording for this experiment, we can reuse that result. For the second video, in which the face is not in close-up and occupies a smaller part of the frame, the recorded execution time also amounts to **27.36 seconds**. It is important to note that the option of using previously saved face detection data (use_previous_tracking_data) was enabled in both cases. The result of the second video is very satisfactory, as there was almost no extension of processing time compared to the video in which the face is in close-up, which tells us that this component is not significant for time consumption during video processing and also indicates the robustness and efficiency of detection under different visual conditions.

The next item we are testing relates to the model's speed when using videos in WebM format, as well as the additional processing that includes background removal, i.e., the green key from the video, or the addition of the alpha component to achieve transparency. As we stated earlier, this functionality is important to us because of the possibility of using this module in real-world applications that require a transparent background.

When measuring time in this segment, the total duration of the process was broken down into three key parts. The first segment relates to frame generation via the Wav2Lip model. Let us recall that to generate the video in WebM format, it was necessary to make certain modifications in the inference.py file, which caused a certain slowdown compared to the original code. The second segment encompasses background removal (of the green background) from each individual frame, while the third and final part of the process involves merging the processed frames into the final video with an alpha channel. The measurement was performed using a video recording lasting **10 seconds** and audio recordings lasting **9, 6**, and **3 seconds**, as in previous measurements. The measurement results for the mentioned three components, as well as the total duration of the process, are shown below.

For an audio duration of **9 seconds**, the background removal time was **22.01 seconds**, the time for creating the final WebM video was **120.26 seconds**, the model duration time was **38.62 seconds**, while the total process duration was **180.89 seconds**.

For an audio duration of **6 seconds**, the background removal time was **13.78 seconds**, the time for creating the WebM video was **79.32 seconds**, the model duration time was **28.26 seconds**, and the total process duration was **121.36 seconds**.

For the shortest example, with an audio duration of **3 seconds**, background removal lasted **6.83 seconds**, WebM video creation lasted **38.83 seconds**, the model duration time was **20.25 seconds**, while the total process duration was **65.91 seconds**.

Analysis of the measured results suggests that the phase of creating the final WebM video with an alpha channel consumes the most time in the entire process. This is expected because video encoding with support for transparency (alpha channel) represents a technically demanding task,

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

especially when using the VP9 codec which is optimized for quality and support for additional channels. On the other hand, we can observe that the duration times for the model itself are also greater, as the measured results for model processing and MP4 video formation are shorter than the processing in this specific case.

In the next test, we attempted to speed up the time for creating the WebM video from the processed frames. In the previous case, the videos were generated at a resolution of **1280x720**, while in this test, we reduced the quality to **640x360** to examine the impact on performance. We repeated the testing using the same video and audio recordings lasting **9, 6**, and **3 seconds**.

Implementation The results showed significant improvements compared to the original quality. For a 9-second audio, the video creation time was reduced from 120.26 to 64.02 seconds ; for 6 seconds, from 79.32 to 41.64 seconds , and for 3 seconds, from 38.83 to 21.27 seconds. This change can be especially useful in systems where the priority is not maximum video quality, but processing speed.

3.9 Practical Implementation in the Unity Environment

A Unity project was created as a demonstration example, considering the capabilities it offers for the development of interactive 3D experiences. The Map Chamber from the game Hogwarts Legacy served as inspiration, where a character communicates with "pictures" on the walls. The interaction occurs by the player selecting one of the offered options/interactions, and based on this, receiving an adequate response that helps them continue the game. In this way, the player learns details they consider relevant for the continuation, and the visual segment is creatively connected with interactivity, creating a dynamic relationship between the player and the game.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495.”

Figure 3.6: Excerpt from the game Hogwarts Legacy [20]

In this project, the implementation of a similar mechanism is planned, where users will have the opportunity to interact with the displays on the walls, which will allow for space exploration and the discovery of new information should this module be integrated into a real game. This concept opens up the possibility of implementing a chatbot that could provide players with various information about the next steps or missions, depending on their in-game interactions. In this way, the chatbot would not be just a passive source of information, but could react based on the player's exploration approach. If a player explores a certain area, the chatbot could offer additional guidance on what they might discover or what steps might be important for further game progression. For the chatbot's correct response, it is necessary to forward information about the progress achieved so far or the player's current status. Based on this data, the next logical step in the game could be determined and relevant advice could be offered. This enables personalized interaction, as the chatbot can adapt its responses according to the current progress and the player's exploration approach.

To connect all components in Unity, it is necessary to create a C# script. For this purpose, a script named "LoadVideo" was created. This script contains the StartVideo() method, which is called via the On Click action of the Send button. The public fields of the LoadVideo class include the inputField of type TextMeshPro input field, from which the user's question is retrieved, and subsequently encoded and passed as part of the HTTP request. There are also GameObjects videoPlane and endScreen, on which the generated video and the animation after playback completion are displayed, respectively. The last public field is videoUrl, which defines the base

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495.”

server URL. Since Ngrok, whose operation we explained earlier, generates a new public URL with every startup, it is necessary to regularly update this URL in the script for correct access to the desired API endpoint. In this example, we send a request to the endpoint for video generation, to which we pass the user's question. Based on this input, the server returns video content, which is then downloaded and locally saved on the device. After the video is downloaded, it is loaded and played via the VideoPlayer component, connected to the videoPlane GameObject. The script automatically links audio playback using the AudioSource component so that the video plays with sound. In short, the entire process is as follows: when the user enters text and presses the button, the StartVideo method is launched, which processes the text, prepares the full URL address, and calls the coroutine function DownloadAndPlayVideo. In this function, the video is downloaded via the UnityWebRequest method, and after the download is complete, the video is saved locally on the device using Application.persistentDataPath, which ensures reliable loading and playback of the file. After saving the video, the path is set in videoPlayer.url, the audio system is configured, video preparation is started, and then the PlayVideo() method is called, which deactivates the waiting screen. When playback is finished, the OnVideoEnd() method is activated, which enables the endScreen GameObject and thus displays the final animation. The script is designed to simply connect user input with the server that returns a personalized video, making it suitable for various applications where media content needs to be retrieved based on certain inputs.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495.”

Program 3.21: LoadVideo script

```
using System.Collections;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Networking;
using UnityEngine.Video;
using TMPro;

public class LoadVideo : MonoBehaviour
{
    public TMP_InputField inputField; // Input field for user text input

    public GameObject videoPlane;     // GameObject where the video will be shown

    public GameObject endScreen;      // GameObject displayed when the video ends (e.g., vortex)

    public string videoUrl = "";      // Base URL to fetch the video

    public string fullUrl;

    private VideoPlayer videoPlayer;
    private AudioSource audioSource;

    private void Awake()
    {
        // Get the VideoPlayer component from videoPlane
        videoPlayer = videoPlane.GetComponent<VideoPlayer>();

        // Get the AudioSource component from videoPlane
        audioSource = videoPlane.GetComponent<AudioSource>();
    }
```

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495.”

```
    // Called by a Unity UI Button's OnClick event
    public void StartVideo()
    {
        string question = inputField.text; //Get user input from the field

        // Encode the question string for URL compatibility
        string encodedVideoString = UnityWebRequest.EscapeURL(question);

        // Create full video request URL
        fullUrl = $"{videoUrl}?video_string={encodedVideoString}";

        // Start downloading and playing video
        StartCoroutine(DownloadAndPlayVideo());
    }

    IEnumerator DownloadAndPlayVideo()
    {
        using (UnityWebRequest request = UnityWebRequest.Get(fullUrl))
        {
            // Send the request
            request.SendWebRequest();

            // Wait until the request is done
            while (!request.isDone)
            {
                yield return null;
            }

            if (request.result == UnityWebRequest.Result.Success)
            {
                // Save video to persistent data path
                string filePath = System.IO.Path.Combine(Application.persistentDataPath,
"downloadedVideo.mp4");

                // Write downloaded video bytes to file
                System.IO.File.WriteAllBytes(filePath, request.downloadHandler.data);

                // Set the video file path for the player
                videoPlayer.url = "file://" + filePath;

                // Configure audio playback
                videoPlayer.audioOutputMode = VideoAudioOutputMode.AudioSource;
                videoPlayer.controlledAudioTrackCount = 1;
```

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495.”

```
                // Enable and assign audio track
                videoPlayer.EnableAudioTrack(0, true);
                videoPlayer.SetTargetAudioSource(0, audioSource);

                // Prepare and play
                videoPlayer.Prepare();
                videoPlayer.prepareCompleted += PlayVideo;
            }
            else
            {
                Debug.LogError("Error downloading video: " + request.error);
            }
        }
    }
```

```
    void PlayVideo(VideoPlayer vp)
    {
        endScreen.SetActive(false);
        videoPlayer.Play();

        // Detect when the video ends
        videoPlayer.loopPointReached += OnVideoEnd;
    }

    void OnVideoEnd(VideoPlayer vp)
    {
        if (endScreen != null)
        {
            endScreen.SetActive(true); // Show end screen when video ends
        }
    }
}
```

Based on this simple Unity project, we demonstrated the operation of our deepfake chatbot, showing how user input can be converted into a personalized video response. This module is flexible and can be easily integrated into any other project, thereby allowing the addition of interactive video communication to various applications and systems. The appearance of the final project can be seen in the images below.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495.”

Figure 3.7: Final Project Appearance

7. Conclusion

7.1 Moral and Ethical Aspect

One of the most significant ethical challenges inherent in the development and application of deepfake technology is its capacity to generate visually and audibly convincing content capable of audience deception. Manipulated videos, which falsely depict individuals saying or doing things they never actually performed, possess the potential to be exploited in deliberate disinformation campaigns. Consequently, such content can precipitate a range of adverse consequences, including blackmail, public opinion manipulation, sabotage, and the destabilization of social and political structures. In the digital age, characterized by the rapid dissemination of news and the frequent bypassing of verification filters, deepfake technology

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

further obfuscates the boundary between truth and falsehood, thereby complicating users' ability to discern reliable sources from manipulated ones. Even when the audience does not fully accept the authenticity of deepfake content , the technology's mere existence fosters an atmosphere of suspicion and skepticism. Research indicates that the prevalence of deepfake technology in the media landscape results in a general erosion of trust, impacting not only specific content but also institutional sources of information. This phenomenon reduces the public's willingness to accept the authenticity of any public statements or recordings. In such an environment, the sense of security and orientation within the digital space is severely compromised. Ultimately, the effect of doubt produced by deepfake content can prove destructive to democratic processes, informed decision-making, and overall social cohesion.[21]

Deepfake technology presents a significant risk not only to society but also to the individual. Fabricated videos can severely compromise a person's credibility, potentially leading to the loss of personal relationships, denied professional advancement, canceled business collaborations, and other detriments. Furthermore, deepfake content can have devastating effects on individuals seeking employment. A Microsoft study revealed that over 90% of employers use online searches to evaluate job candidates, with these results negatively influencing their decision in over 77% of cases, often due to "inappropriate photos". Deepfake content further intensifies this risk because once-published material is exceptionally difficult to remove from the internet, leaving a persistent digital trace that compromises the victims' private and professional lives. Beyond its effects on the individual, false content can broadly impact society. Deepfake technology poses a serious threat to political stability by enabling the dissemination of false, yet compelling, video and audio content that can directly influence election outcomes. In addition to injecting visual and auditory falsehoods into political discourse, deepfake content serves as a potential means of sabotage, such as through the distribution of compromising, yet fabricated, recordings of political candidates. Crucially, such content can have a decisive effect if strategically released at a critical moment. A notable precedent is the 2017 French presidential election, where a manipulation campaign involved the theft, alteration, and publication of documents. Although those efforts ultimately failed, the scenario would have been far more severe had the attackers utilized skillfully prepared deepfake content.[22]

To date, only a handful of governments have adopted regulations specifically concerning artificial intelligence, while the majority have yet to do so, primarily due to concerns regarding the potential restriction of freedom of speech. Although most internet platforms, such as YouTube, have implemented certain legal mechanisms for content control, this process remains costly and time-consuming. The central challenge is that deepfake content often remains undetected by the naked eye. This difficulty has prompted both state institutions and private platforms to developdeepfake detection technologies and establish regulations governing their use. Proposed methods for combating misuse include the mandatory labeling of generated media, the development of forensic video analysis software, and the strengthening of the criminal-legal framework for cases of digital fraud.[23] Given the increasing ease of content creation and

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495.”

technological advancement, the detection of malicious content is an escalating challenge that demands continuous future work and investment.

7.2 Conclusion

The paper, entitled "Real-Time Deepfake Video Creation", successfully presented a methodology for creating a functional deepfake chatbot, emphasizing the delivery of a convincing, real-time user experience. The model's current response time is within acceptable limits for a satisfactory level of interaction, and further convergence toward a true "real-time" response is anticipated with future technological developments, especially concerning the availability and capability of graphics cards for end-users.

During implementation, particular attention was devoted to optimizing resource consumption to ensure the fastest and most stable performance in response generation. In this context, the Wav2Lip model was utilized, which enables the synchronization of lip movements with the audio signal while imposing a minimal system load. By focusing exclusively on the lip region, this model reduces the complexity of image processing, which facilitates faster real-time response generation. This approach proved highly effective for chatbot interaction, where response speed is a critical factor in user experience. Furthermore, the functionality for removing the video background was successfully implemented, thereby providing an additional degree of flexibility and expanding the potential application of this solution across a wider spectrum of uses.

This work establishes a foundation for future research and optimization directed toward creating faster, more efficient, and more accessible real-time deepfake systems. Currently, the majority of the processing time remains dedicated to preserving visual transparency, which significantly impacts overall system performance. Consequently, it would be advantageous to consider implementing a predefined and optimized background. This strategy would reduce the number of operations the system must execute in real-time , thereby minimizing the need for additional algorithmic processing and shortening the response latency.

A crucial element for future implementation concerns the model's execution method. Given the considerable processing demands and the limited graphical resources on typical user devices , the optimal solution is to rely on a robust server-side infrastructure. Utilizing more powerful servers for model execution would ensure more stable and faster operation, reduce the computational load on the end-user device, and enable the broader adoption and usability of such solutions across diverse platforms and devices.

Finally, it is essential to stress that such systems are not yet fully prepared for the automation of complex or sensitive communication scenarios. To prevent the spread of irrelevant, inaccurate, or potentially harmful information, input queries must be carefully restricted, ensuring they are thematically limited and controlled. To mitigate misuse, these aspects must be addressed during the development phase through the implementation of robust security mechanisms, the

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495.”

limitation of output content, and the transparency of the algorithm's operation. This caution is especially critical when the technology is used in areas involving socially sensitive content. Ultimately, this technology necessitates continuous human oversight to guarantee its responsible, ethical, and secure application in real-world environments.

References

[1] Preeti, Kumar, M., Sharma, H. K., "A gan-based model of deepfake detection in social media", Procedia Computer Science, Vol. 218, 2023, str. 2153-2162, international Conference on Machine Learning and Data Engineering, available: https://www.sciencedirect.com/science/article/pii/S1877050923001916s

[2] Paul, O. A., "Deepfakes generated by generative adversarial networks", available: https://digitalcommons.georgiasouthern.edu/honors-theses/671 Honors College Theses. 671. 2021.

[3] George, A. S., George, A., "Deepfakes: The evolution of hyper realistic media manipulation", Vol. 01, 12 2023, str. 58-74.

[4] Kleine, F., "Perception of deepfake technology - the influence of the recipients' affinity for technology on the perception of deepfakes", Doktorski rad, 08 2022.

[5] Kietzmann, J. H., Lee, L. W., McCarthy, I. P., Kietzmann, T. C., "Deepfakes: Trick or treat?", Business Horizons, Vol. 63, No. 2, 2019, str. 135–146, available: https://doi.org/10.1016/j.bushor.2019.11.006

[6] Agostini, D. D. (2020) Why deepfakes will make you play vi- deo games instead of movies, available: https://medium.com/predict/why-deepfakes-will-make-you-play-video-games-instead-of-movies-99ee5c2d7c9e

[7] Whittaker, L., Kietzmann, T., Kietzmann, J., Dabirian, A., ""all around me are synthetic faces": The mad world of ai-generated media", IT Professional, Vol. 22, 09 2020, str. 90-99.

[8] Battegazzorre, E., Bottino, A., Lamberti, F., "Training medical communication skills with virtual patients: Literature review and directions for future research", available: https://arxiv.org/abs/2104.00316 2021.

[9] Tariq, S., Abuadbba, A., Moore, K., "Deepfake in the metaverse: Security implications for virtual gaming, meetings, and offices", in The 2nd Workshop on the security implications of Deepfakes and Cheapfakes, ser. ASIA CCS '23. ACM, Jul. 2023, str. 16–19, available: http://dx.doi.org/10.1145/3595353.3595880

[10] Chung, J. S., Jamaludin, A., Zisserman, A., "You said that?", available: https://arxiv.org/abs/1705.02966 2017.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."

[11] Prajwal, K. R., Mukhopadhyay, R., Namboodiri, V. P., Jawahar, C., "A lip sync expert is all you need for speech to lip generation in the wild", in Proceedings of the 28th ACM International Conference on Multimedia, ser. MM '20. ACM, Oct. 2020, str. 484–492, available: http://dx.doi.org/10.1145/3394171.3413532

[12] NVIDIA Corporation, "Nvidiaturing gpu architecture", https://images.nvidia. com/aem-dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/ NVIDIA-Turing-Architecture-Whitepaper.pdf, whitepaper. 2018.

[13] Grattafiori, A., Dubey, A., Jauhri, A. et al., "The llama 3 herd of models", available: https://arxiv.org/abs/2407.21783 2024.

[14] Meta AI, "Introducing Meta Llama 3", https://ai.meta.com/blog/meta-llama-3/, available: https://ai.meta.com/blog/meta-llama-3/. Apr. 2024.

[15] Deepgram Inc., "Deepgram api overview", available: https://developers.deepgram. com/reference/deepgram-api-overview 2025.

[16] Pallets Projects. (n.d.) Flask documentation, available: https://flask.palletsprojects. com/

[17] Ngrok. (n.d.) What is ngrok?, available: https://ngrok.com/docs/what-is-ngrok/

[18] Chauhan, N. S. (2020) Wav2lip: A lip sync expert is all you need for speech to lip generation in the wild, available: https://medium.com/towards-data-science/wav2lip-a-lip-sync-expert-is-all-you-need-for-speech-to-lip-generation-in-the-wild-b1cb48787190

[19] anothermartz, "Easy-Wav2Lip: A Robust Lip-Syncing Model", https://github.com/ anothermartz/Easy-Wav2Lip, 2024.

[20] Apelt, M., "Hogwarts legacy: Mit diesen gestrichenen inhalten, hätte es noch besser werden können", https://www.playcentral.de/hogwarts-legacy-gestrichene-elemente-besser/, May 2023.

[21] Vaccari, C., Chadwick, A., "Deepfakes and disinformation: Exploring the impact of synthetic political video on deception, uncertainty, and trust in news", Social Media + Society, Vol. 6, No. 1, 2020, str. 2056305120903408, available: https://doi.org/10.1177/2056305120903408

[22] Citron, D. K., Chesney, R., "Deep fakes: A looming challenge for privacy, democracy, and national security", California Law Review, Vol. 107, 2019, str. 1753–1819, available: https://scholarship.law.bu.edu/faculty_scholarship/640

[23] Ramluckan, T., "Deepfakes: The legal implications", in Proceedings of The 19th International Conference on Cyber Warfare and Security (ICCWS 2024), Vol. 19, No. 1. Kidmore End, United Kingdom: Academic Conferences International Limited, 2024, str. –, available: https://papers.academic-conferences.org/index.php/iccws/article/ view/2099

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD PODPREDSEDU VLÁDY SLOVENSKEJ REPUBLIKY PRE PLÁN OBNOVY A ZNALOSTNÚ EKONOMIKU
VAIA VÝSKUMNÁ A INOVAČNÁ AUTORITA
VÝSKUMNÁ AGENTÚRA

„Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00495."