POLYTECHINC OF TURIN

Master Degree course in Computer Engineering (Graphics and Multimedia)

Master Degree Thesis

From 2D Archives to 3D Avatars: Automation and Optimization of Photorealistic Synthetic Humans in the Broadcast Industry

Supervisors
Prof. Andrea Bottino
Prof. Francesco Strada
Co-Sup Ing. Roberto Iacoviello

Candidate
Miriana Martini

Academic Year 2022-2023

Summary

Throughout history, the idea of creating artificial humans has been a constant in the collective imagination, fascinating different cultures and eras. Not only does the interest in this topic continue, but thanks to the possibilities offered by today's technological landscape, it is possible to create 3D avatars with a remarkable degree of photorealism and likeness.

This study, conducted in close collaboration with RAI, aims to investigate and optimize an advanced workflow for the creation of photorealistic 3D avatars. The broadcast industry is constantly evolving, and viewers are increasingly demanding engaging and interactive contents with accessibility at the forefront. The challenge is to give a second life to Rai's huge archive of 2D footage of the subject by creating 3D avatars from 2D images, carefully analyzing the technical and creative challenges encountered during the creation process.

A critical aspect of this research is the identification and implementation of solutions to automate the workflow of creating synthetic humans, with the main goal of significantly reducing the need for human intervention and ensuring an efficient and flexible creation process that can operate at scale.

Subsequently, an evaluation of the generated synthetic humans was performed, both from a subjective and objective point of view, focusing on fidelity and similarity to the original subjects.

This study represents a first step towards a future where the creation of photorealistic avatars will be highly automated, enabling easy and widespread integration of synthetic humans into the broadcast and metaverse ecosystems. This will have a significant impact on visual storytelling, revolutionizing the way audiences interact with media content and opening up new possibilities for visual storytelling.

Contents

Chapter 1

Introduction

Throughout history, the concept of artificial human beings has been a constant in the human imagination, emerging as a recurring theme that has fascinated different cultures and eras. From ancient mythological stories, such as the legend of Pygmalion, through the groundbreaking pages of Frankenstein written by Mary Shelley, the depiction of artificial humans has spanned time, coming to permeate modern science fiction and products of the entertainment industry.

These narratives and representations have played a crucial role, continually shaping and reformulating the way we perceive and imagine artificial humanity. Nowadays, the interest in this topic not only persists, but with the advent of Synthetic Humans, or Digital Humans, has intensified and the ambition to create human beings artificial humans are intertwined with the opportunities offered by the contemporary technological landscape.

The goal being pursued is dual-faced: firstly, to accurately replicate what is perceived as a "human being", and secondly, to transfer and reinterpret this essence within digital domains.

1.1 Understanding Synthetic Humans

Synthetic Humans, also known as Digital Twins of Humans, Embodied Conversational Agents, Virtual Agents or simply Avatars, are essentially digital representations of real individuals. These are not limited to merely reproducing physical appearance, facial expressions and body movements of a person, but, depending on their purpose, they can also extend to the analysis and reproduction of human's peculiarities that make each individual unique, such as personality, sensitivity, thoughts and abilities. They can recreate in digital space social components such as human behavior and communication, allowing empathetic interactions similar to human ones.
Synthetic entities arise from the convergence of several disciplines, including computer graphics, computer vision, and artificial intelligence. Together with the use of Generative AI, it is possible to train them using specific information and data, generating a style or a "tone of voice" that reflects the user's needs, ensuring in this way a personalized and

relevant communication.
In addition, thanks to sophisticated emotion-recognition algorithms, digital humans are emotionally receptive, able to perceive and respond to users' emotions, creating a bidirectional communication and demonstrating a level of empathy comparable to the human one.
The term "Synthetic Humans"emphasizes the aspiration to develop simulations so refined and detailed that they blur the boundaries between virtual and real, making these entities digital virtually indistinguishable from human beings.

1.1.1 Contextualization of the concept of Human Digital Twins

In recent years, the digital representation of humans, is subject to progressive interest, thanks to a decade-long alternation of the focus of digital twin computing between "things" and "human begins" that can be observed in figure 1.1.

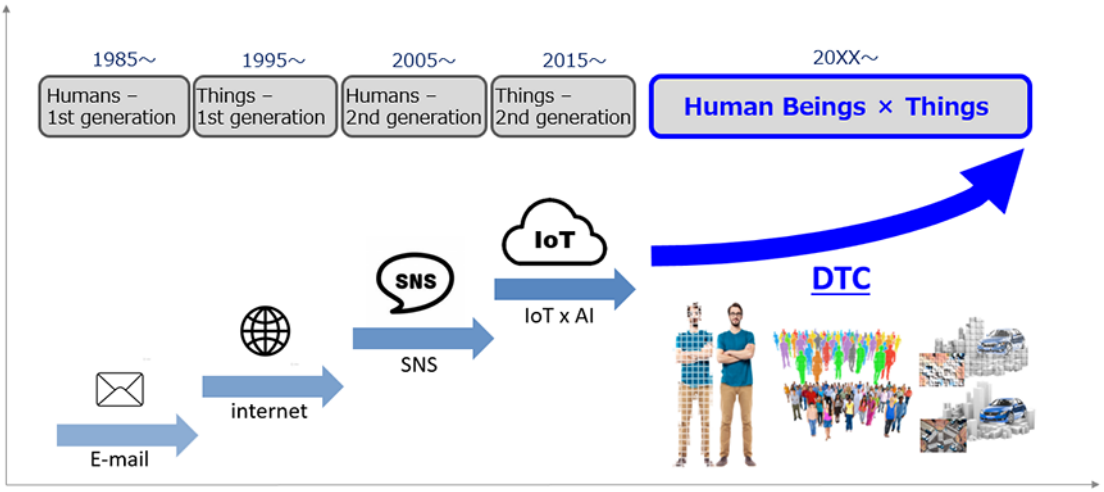


Figure 1.1: In 1985, human communication began to digitize with the introduction of e-mail. A decade later, in 1995, the focus shifted to the digitization of "things", such as time-schedules and maps, through the expansion of the Internet. In 2005, the focus shifted again to humanity, emphasizing connections and social networks through the emergence of social media. In 2015, digitization resumed its focus on objects, with the integration of the Internet of Things (IoT) and the development of artificial intelligence [1].

Analyzing current trends, it is possible to predict that the digitization of humankind will shape the future, considering the Covid-19 pandemic that has accelerated the process of virtualization of services, the wide interest in VR/AR/MR experiences, and the commitment of Big Tech to the Decentralized Web3, in which the Metaverse and the consequent definition of a virtual identity will play a key role.
It must be acknowledged that the impact of digital humans will not be limited to the virtual world, as it is foreseeable that synthetic entities will surpass human capabilities in a number of areas, and the data generated by them will play a prominent role, increasingly

replacing real data. Leading companies such as Microsoft, DataGen, Epic Games and Reallusion, are already adopting data from digital humans to enhance AI model training [2]. The goal is to overcome limitations of scarce, costly to collect, and privacy-sensitive real-world data, by enabling AI systems to learn from diverse, controlled and comprehensive synthetic datasets that mirror real human behaviors and interactions.

In the near future interaction with synthetic entities is set to become the norm in the emerging digital age, an evolution that will profoundly change the online experience and revolutionize technological, scientific and social fields.

1.1.2 Applications of Synthetic Humans in various sectors

Synthetic Humans find application in many domains, demonstrating their versatility and their cross-cutting impact:

- **Entertainment:**
  In the entertainment sector, video games stand out as a particularly rich domain for the application of these technologies, demonstrated by prominent titles like "The Last of Us" [3] and "Detroit Become Human" [4], in which digital characters express surprisingly detailed emotions and likeness, contributing to the creation of engaging and lifelike gaming worlds.
  Virtual production represents another area in which Synthetic Humans are valuable by simplifying and optimizing the creative process. The TV series "The Mandalorian" is an example of how the pre-visualization of scenes using digital actors can improve the efficiency of production [5].
  The music and live performance world has also been transformed by the Synthetic Humans, with groundbreaking initiatives such as ABBA's virtual concert, which allowed fans to experience a unique live performance, even though in the physical absence of the band members, see figure 1.2.
  In the social media field, digital personalities such as Lil Miquela [6] are virtual influencers and brand virtual ambassadors, have opened a new era in brand-consumer communication. Their ability to interact with audiences and participate in advertising campaigns marks a significant step toward new forms of engagement and digital relationships.

Figure 1.2: Avatars on stage during the Abba Voyage concert in London [7].

- **Training and Simulation:** Synthetic Humans are valuable tools for training in multiple fields such as aviation, medicine, education and many others. Their ability to reproduce realistic scenarios through digital human avatars results in more effective preparation of professionals and students, enabling them to deal with complex situations with greater confidence and safety.
  They take on the role of digital mentors, guiding new employees through the onboarding and on-the-job training process. This innovative approach accelerates the process of acquiring the required skills, while providing ongoing, personalized support.
  In the educational field, they can take on the role of virtual tutors and adapt learning to the specific needs of each student, this includes simulating conversations in a foreign language, creating virtual labs, or reproducing realistic scenarios to facilitate the student immersion and enhance the learning experience.
  Using Synthetic Humans to simulate the behavior and interactions of the workers within the Digital Twin environment, companies are able to assess risks, predict potential future scenarios, and identify opportunities for optimization.
  This approach not only allows companies to improve safety and operational efficiency, but also to refine the design of physical spaces in a precise way. Specifically, in architecture, their presence in Digital Twin models makes it possible to simulate the impact of design choices on the usability of spaces and the well-being of the people who occupy them [8]. This aspect turns out to be crucial for designing more functional, comfortable and safe buildings.

- **Scientific Research:** In the realm of scientific research, digital avatars facilitate the study of human behavior in controlled environments. Human perception, reactions and social interactions can be analyzed in detail, providing valuable information for a range of disciplines.
  In the fields of psychology and neuroscience, Synthetic Humans prove to be powerful tools for simulating human behaviors and studying emotional responses in specific situations. The creation of a "Virtual Identity" within virtual worlds opens up new perspectives for analyzing how people choose to represent themselves in digital

contexts, offering valuable insights into the dynamics between virtual identity and self-perception.
This aspect is particularly relevant in therapeutic settings, as it allows individuals to explore and confront aspects of their identity, fears or desires in a protected and controlled context, facilitating processes of introspection and self-knowledge

- **Health Care:** Synthetic Humans don't simply impersonate patients: they can also take on the role of doctors or medical staff, providing a mode of interaction that promotes empathy and the ability to relate to real patients.
  In terms of direct care, they can offer support during the post-operative recovery, facilitating both physical and cognitive rehabilitation processes.
  They are also able to perform an initial assessment of symptoms, that can direct patients to the most appropriate channels of care, the result is a more efficient patient management, that can avoid congestion in health facilities.
  Another relevant aspect is the creation of digital patient twins, which are faithful virtual replicas that can be used to monitor health status, predict the evolution of diseases and tailor treatments to the specific needs of each individual.

- **Communication and Interaction:** With the use of realistic avatars, digital communication gains depth and authenticity, elevating the quality of video conferencing, virtual chats and online interactions to a level never reached before.
  Through their ability to understand language and context, to faithfully represent a brand's identity and values, and to create authentic and empathetic interactions, Synthetic Humans offer brands new and effective ways of contacting and interacting with their audiences.
  This represents a significant advancement compared to traditional chatbots, such as those employed by Amazon, since unlike these conventional systems, Synthetic Humans are able to give context-aware responses that go far beyond simply answering a pre-set pool of questions.

Figure 1.3: Mark Zuckerberg interviewed in VR using Meta's new Avatar Codecs[9]

1.2 Contextualization of the concept of Synthetic Humans in the broadcast world

In the broadcast industry, which embraces the transmission of audiovisual content through a variety of channels such as television, radio and online platforms, the inclusion of highly realistic digital characters is redefining how content is created, distributed and perceived, offering new levels of audience engagement and participation.
As television broadcasters and journalistic platforms [10] experiment with the use of digital avatars to conduct specific programs and segments, **virtual interviews** can be conducted, in which a real person converses with a synthetic character. This mode proves particularly valuable for interviewing people located in geographically distant places and times. In parallel in the context of **livestreaming and real-time conversations** [11], Synthetic Humans open new frontiers of interaction.
Whether at sporting [12] events or live events, a "Real Time Conversation Digital Human" can take on the role of conductor, providing live commentary and interacting with the audience in a dynamic and engaging manner.
In the field of **documentary and film production**, the use of Synthetic Humans results in the possibility of exploring new visual narratives. The ability to recreate historical characters opens up new storytelling perspectives, allowing content creators to bring moments and figures from the past to life, while their presence in films and TV series as special effects allows them to overcome physical and creative limitations, replacing human actors in scenes of particular complexity.
They can be used to create virtual avatars for **advertising campaigns**, allowing greater flexibility and creative control. These avatars can be programmed to interact with consumers in personalized ways, enhancing the shopping experience and providing relevant product information [13].

1.2.1 Impact of Synthetic Humans on the Future

The future of broadcasting promises to be intrinsically connected to Synthetic Humans, with intriguing prospects for entertainment. Viewers will have the opportunity for more immersive television and film experiences opening up new dimensions of engagement and participation.
Entertainment will become more interactive and personalized, adapting to viewers' individual preferences. In parallel, accessibility in media will reach new heights, they will offer innovative solutions for people with disabilities, providing more comprehensive access to television programs, multimedia content and information. Avatars will be able to communicate in sign language, support visual accessibility, and offer new ways of enjoying content for people with developmental disabilities. This will represent a significant step toward a more inclusive society.

In addition, Synthetic Humans will serve as a bridge between physical reality and the Metaverse, becoming an essential part of the interaction between the physical environment and the virtual world. The Metaverse could become a platform where viewers interact directly with Synthetic Humans in virtual shows and experience digital worlds.

1.3 3D Reconstruction of Human Models: A Research Perspective

In recent years, significant studies have been conducted in the realm of three-dimensional reconstruction of human models. The three-dimensional reconstruction of digital models of humans poses a complex challenge within the fields of computer vision and graphics. The primary objective is to accurately retrieve the geometry and appearance of humans from visual sources such as images, videos, or depth data, precisely translating two-dimensional information into detailed and realistic three-dimensional representations of human bodies.

This section of the chapter explores the latest research [14] in this domain, highlighting innovative methodologies and techniques that contribute to the refinement of creating increasingly realistic and detailed avatars.

1.3.1 Detailed and Realistic Representation

The primary focus is on creating three-dimensional models capable of accurately capturing not only the physical form of an individual but also subtle details such as facial expressions and clothing folds. Challenges arise from anatomical complexity and the need to make such models accessible, efficient, and cost-effective.

1.3.2 Evolution from Traditional Pipelines to Machine Learning Innovation

Traditional reconstruction pipelines, relying on complex acquisition systems, serve as a fundamental starting point. However, limitations in terms of cost and portability of such approaches have led to the adoption of innovative machine learning-based approaches. These new approaches aim to overcome intrinsic challenges by providing efficient, accessible solutions capable of producing photorealistic three-dimensional models.
Different approaches to reconstructing the 3D model are illustrated in Figure 1.4:
a) The traditional reconstruction pipeline necessitates a dense camera array or depth cameras and involves numerous isolated steps.
b) Regression-based methods employ neural networks to directly regress human geometry or appearance from input images.
c) Optimization-based methods, utilizing differentiable rendering, reconstruct 3D human models by minimizing the rendering error between re-rendered images and the corresponding input images.

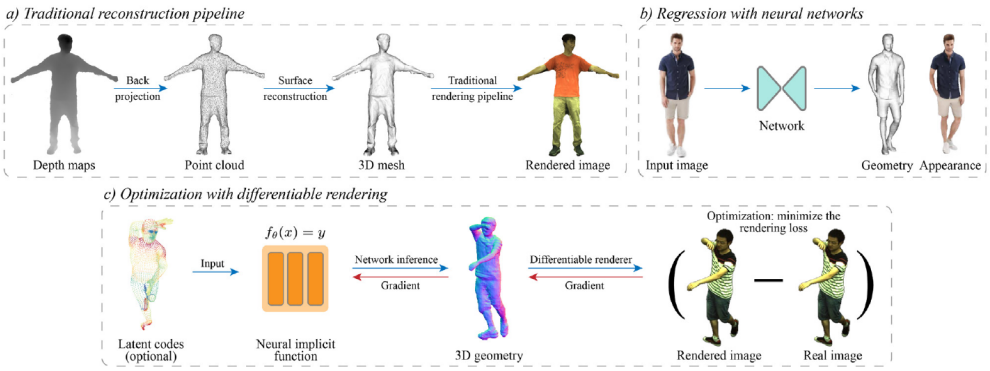


Figure 1.4: Comparison between three reconstruction approaches [14].

1.3.3 Current Trends and Developments

Recent advancements indicate a shift towards using deep neural networks to enhance the efficiency and robustness of reconstruction by learning human models from existing data. Implicit function-based approaches emerge as a preference over traditional forms such as meshes and voxels, emphasizing the importance of detailed visual rendering [14].

Mesh-based Approaches

Mesh-based approaches play a crucial role in three-dimensional human reconstruction, enabling the creation of detailed models that accurately reflect facial expressions and clothing shapes. However, the low dimensionality of meshes may limit their ability to capture high-frequency details [14].

Implicit Function-based Approaches

The use of implicit functions proves to be an effective alternative, with advantages such as ease of optimization, spatial resolution independence, and memory efficiency. Techniques like "Pixel-aligned Implicit Function (PIFu)," "PIFuHD," and "Geo-PIFu" enhance the rendering of details and local features [14].

Differentiable Rendering

Differentiable rendering is crucial for achieving highly photorealistic human models. Challenges and techniques, such as "mesh renderer," "differentiable sphere tracing," and "volume rendering," are explored to address efficiency issues and dynamic scene modeling [14].

1.3.4 Future Perspectives

Future research focuses on generalizable methods, efficient reconstruction for multiple subjects, advancements in photorealistic rendering, and improved techniques for scene content manipulation, such as object manipulation and clothing changes. The innovative use of neural networks promises to make three-dimensional reconstruction more accessible and cost-effective, thereby overcoming limitations of traditional hardware-intensive setups.

1.4 Problems and Challenges

This innovation of Synthetic Humans brings significant challenges and technical problems that require strategic solutions.
Exploring key issues and challenges associated with the utilization of Synthetic Humans as digital representations in the broadcast world will be undertaken.

- **Data Integration and Quality** Creating an accurate photorealistic avatar requires the integration of data from various sources, including motion suits, mobile cameras, different software, and existing archives. Ensuring the quality, consistency, and reliability of this data can be a significant challenge. Data can be noisy, incomplete, or discordant, making it essential to develop rigorous data acquisition protocols and quality control systems.

- **Interoperability** Different workflow components may use different technologies and standards, making it complex to create an integrated pipeline for building Digital Humans.
  Ensuring that all components communicate effectively and seamlessly can be a daunting task. Standardization of interfaces and protocols can help mitigate this challenge.

- **Model Validation** Developing accurate 3D models that closely mimic human behavior is essential for the creation of convincing Digital Twins.

Validating these models to match actual performance in terms of similarity and movements can be difficult due to the lack of universal metrics or standards for assessing realism.

- **Cost and Resource Allocation** Developing a high-fidelity Synthetic Human can be resource intensive, requiring time, money, and people involved in the project. Adopting AI-based software and tools can be a strategic challenge, as it requires planning and allocating resources effectively to achieve the desired results.

- **Lack of Standardization** The lack of standardized protocols for the development of Digital Twins can lead to compatibility issues and difficulties in collaboration across different organizations or sectors.
  Establishing common standards and shared protocols could help overcome these barriers.

- **Cultural and Organizational Resistance** The introduction of new technologies for creating photorealistic avatars may require changes in the way production centers work and make decisions. There may be cultural or organizational resistance from employees who are unfamiliar with the technology or reluctant to adopt new processes.

- **Business Aspects** The lack of standardized protocols for rewarding the work of talent and ensuring the quality of work performed can create obstacles and risks. Establishing clear rules and commercial guidelines could be crucial to sustaining the Synthetic Humans industry.

- **Ethics and Privacy** Regarding the creation and manipulation of digital representations of real individuals, strict principles need to be established to ensure that the dignity and rights of digitally represented individuals are respected. Distributed content using digital avatars must be clearly distinguishable and identifiable from real ones, the goal is to maintain transparency so as to safeguard both the integrity of the individual and to build a more ethical and responsible digital environment.

1.5 Purpose and Objectives of the Thesis

This thesis work, conducted in collaboration with RAI, the well-known Italian radio and television broadcaster, aims to analyze and improve the workflow for the production of photorealistic avatars that portray relevant figures, such as historical figures, performers, and athletes, by utilizing the resources available in the RAI archive.

The initial phase of the research focused on outlining every stage of the workflow, with the aim of establishing a comprehensive and organized set of procedures for developing Synthetic Humans. By conducting a thorough examination of each phase, the main technical and creative challenges that emerge during the avatar generation process were identified.

Afterwards, an evaluation of the generated Synthetic Humans was conducted, with an emphasis on fidelity and likeness to the original subjects. The goal was to obtain the most accurate outcomes while minimizing the resources expended in terms of time, people and artistic skills required.

In conclusion, the thesis is dedicated to identifying workflow automation solutions to make the entire process more efficient and reduce the need for human involvement. This work is proposed as a first step forward in the field of automatic generation of photorealistic avatars, with the aspiration of making the integration of Synthetic Humans in television and multimedia productions more accessible and flexible.

Chapter 2

Technologies involved

To embark on the path towards the automated creation of photorealistic avatars from historical images in the RAI archive, it is essential to gain a comprehensive understanding of the state of the art in three critical areas:

- Image restoration
- 3D facial reconstruction
- Facial animation

This chapter is dedicated to examining advanced software solutions and prevailing methodologies in these three domains, with the aim of providing a comprehensive overview of current trends and best practices.

Furthermore, a special emphasis will be placed on MetaHuman, a cutting-edge technology in the realm of digital character creation, given its increasing adoption in the entertainment and media industry.

2.1 Image Restoration

The image restoration process aims to obtain high-quality images from damaged or compromised input images. Image corruption can occur due to the capture process (e.g., noise, lens blur), post-processing (e.g., JPEG compression), or photography under non-ideal conditions (e.g., fog, motion blur).

Historical images from the RAI archive often suffer from issues such as low resolution and noise, making the use of Image Restoration tools essential.

2.1.1 Adobe Photoshop Super-Resolution feature

Adobe Photoshop Super-Resolution [15] is a well-known software for upscaling and Super Resolution of images using machine learning. Through machine learning, it analyzes image patterns and generates realistic and consistent details during upscaling.

It is the most widely used and versatile software because it offers a high degree of creative control, advanced retouching tools, and the option to restore textures manually to ensure optimal results.
Its processing speed allows for a seamless workflow.

2.1.2 Topaz Gigapixel

Topaz Gigapixel [16] is specialized software for upscaling images and offers detailed control over resolution and quality. It is designed to increase the resolution up to 6 times the original image size. Unlike Adobe Photoshop, it allows you to preview the image before upscaling.
Because it is specialized for this task, it provides more controls for addressing issues like blurring, noise, color bleed, and face refinement.

2.1.3 GFP-GAN

This project is built upon a specially trained Generative Adversarial Network (GAN) for facial image restoration, with an emphasis on prioritizing facial details. GFP-GAN [17] distinguishes itself for its ability to achieve an optimal balance between realism and fidelity in the image restoration process. This achievement is made possible through the utilization of a broad and diverse knowledge base embedded in a pre-trained generative network, specifically designed for facial image restoration.

As we can see in figure 2.1, it consists of a degradation removal module (U-Net) and a pre-trained face GAN serving as a facial prior [17]. These components are connected through latent code mapping and various levels of Channel-Split Spatial Feature Transform (CS-SFT). During training, the following processes are used:
1) Intermediate restoration losses are employed to remove complex degradation.
2) Facial component losses with discriminators enhance the face.
3) Losses for preserving identity are applied to maintain the identity of the face.

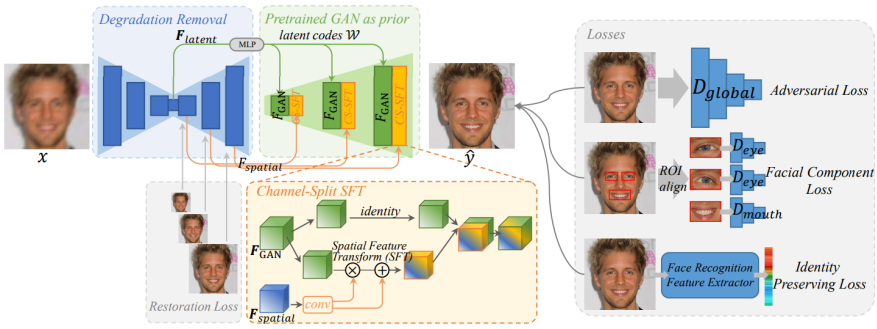


Figure 2.1: **The GFP-GAN framework overview**

2.1.4 CodeFormer

In the quest for solutions to restore damaged or dated faces, CodeFormer [18] emerges as a powerful facial restoration algorithm.
CodeFormer reduces uncertainty in transforming degraded faces into high-quality ones. This method is based on a Transformer-based prediction network to model low-quality faces and predict codes, producing impressive results even with heavily damaged images. Thanks to its advanced knowledge and global model, CodeFormer outperforms existing methods in terms of quality and fidelity, demonstrating significant robustness to degradation. Results on both synthetic and real-world datasets confirm the effectiveness of this approach.

As we can see in figure 2.2 initially, a discrete codebook and a decoder are learned to store high-quality visual parts of facial images through self-constructive learning. Then, with a fixed codebook and decoder, a Transformer module is introduced for predicting the code sequence, modeling the global composition of the low-quality input face. Additionally, a controllable feature transformation module is used to control the information flow from the low-quality encoder to the decoder.
Note that this connection is optional and can be disabled to avoid adverse effects when inputs are severely degraded, and a scalar weight can be adjusted to strike a balance between quality and fidelity. [18]

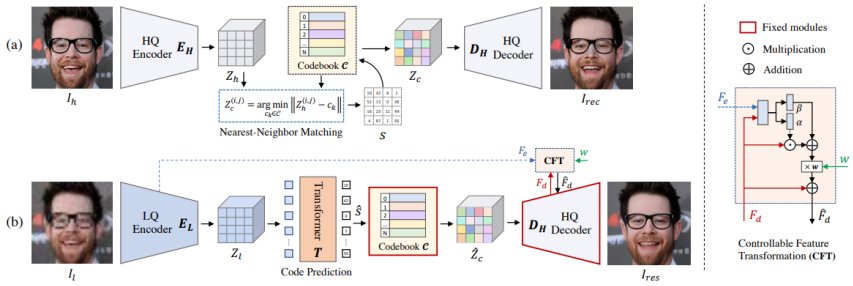


Figure 2.2: **CodeFormer Structure**

2.2 Face Reconstruction

3D Face Reconstruction, which also includes **Texture Reconstruction**, is a computer vision activity involving the creation of a 3D model of a human face from a 2D image or a set of images. The output of this process can be used for various applications such as virtual reality, animation, and biometric identification.

This section will explore both advanced methodologies and software/tools/plugins for creating 3D models of human faces from 2D images or sets of images.

2.2.1 3D Photogrammetry

Among the most commonly used methodologies for face reconstruction, we have **3D Photogrammetry**.
This approach involves capturing 2D images from various angles and using stereo vision algorithms to generate 3D models of faces. Examples of software based on this technology include ***Intel RealSense*** [19] and ***RealityCapture*** [20].

Intel RealSenseis an Intel 3D sensor platform that captures high-quality 3D data from human faces using RGB cameras, infrared sensors, and a structured light projector. The associated software processes this data in real-time to create 3D models of faces that can be used in facial recognition and animation applications.

RealityCapture is photogrammetry software that generates 3D models, including facial models, from 2D images captured from various angles. This software is known for producing high-resolution 3D models with realistic textures and is widely used in industries such as architecture, video games, and film.

2.2.2 3D Scanning

The 3D scanning technique for Face Reconstruction is an advanced method for acquiring detailed three-dimensional data of the human face. This approach involves using 3D scanning devices, such as Microsoft Kinect or similar scanners, that employ structured light technology, which utilizes the deformation of projected light on a face to calculate its 3D geometry.

This technology is widely employed in facial modeling, offering a precise solution for capturing facial geometry. A well-known software that works with the Microsoft Kinect is ***Skanect*** [21], which processes 3D scanning data to create detailed facial models usable in applications such as animation, virtual reality, and game development.

2.2.3 Blender Face Builder plugin

Face Builder is a plugin developed by Keentools for Blender, and it provides an intuitive tool for creating 3D models of human faces using a series of reference photos [22] [23].

FaceBuilder reads the information contained in the photos and uses it to automatically set up virtual cameras, allowing you to use photos of various sizes, like in figure 2.3.

For the highest quality and precision, it is recommended to use photos with neutral facial expressions, as we can see in figure 2.4. However, FaceBuilder is also capable of supporting non-neutral facial expressions, allowing for acceptable results even with photos that do not show neutral expressions.

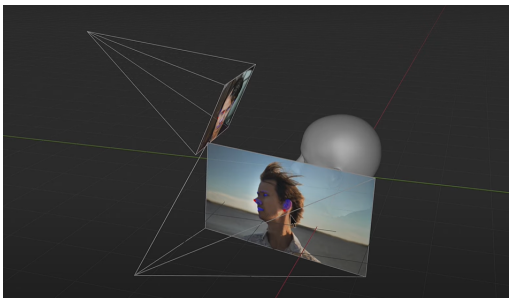


Figure 2.3: Automatic camera setup.

Figure 2.4: Support for facial expressions.

The use of artificial intelligence automates the process of aligning facial points, like in figure 2.5, eliminating the need for manual point placement initially. However, it's important to note that automatic alignment may not always be perfect, so slight manual adjustments may be required. Nonetheless, this approach significantly reduces the time needed for facial modeling.

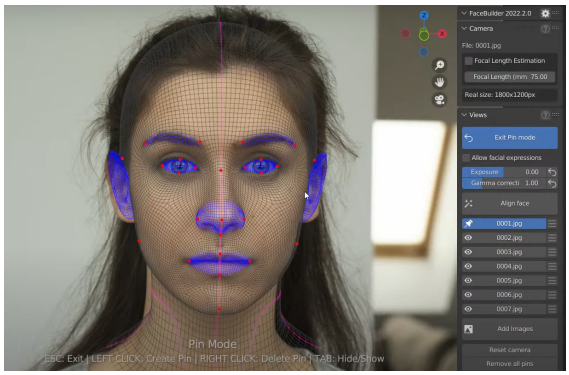


Figure 2.5: Automatic face alignment with AI.

It allows for extracting the texture for the 3D face using the views in which the model has been alligned. FaceBuilder offers four UV map options, that we can see in figure 2.6, and there are: Butterfly, Legacy, Maxface, and Spherical, each with specific features. In the latest version, a UV map (mh that stands for MetaHuman) compatible with Metahuman has been made available.

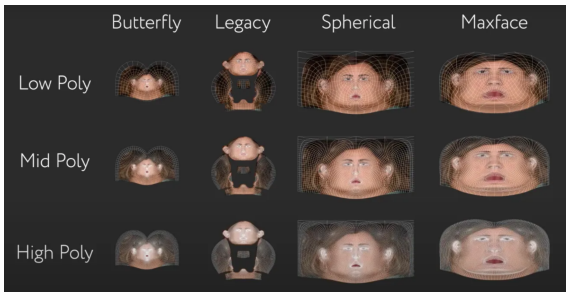


Figure 2.6: UV maps in Face Builder. [23]

Models generated through FaceBuilder can be easily exported in formats compatible with Unreal Engine and Unity, enabling their use in game development and animation contexts. This gives FaceBuilder a high degree of versatility in terms of functionality and export capabilities, making it an extremely valuable resource.

Another advantage of FaceBuilder is its ability to import facial animations recorded using the Live Link Face app by Epic Games. This app is compatible with creating automated animations on Metahuman through Unreal Engine.

2.2.4 Character Creator Headshot plugin

The Headshot plugin for Character Creator [24] serves as an alternative to Blender's Face Builder. It provides advanced features for generating realistic 3D models from photographs, with the ability to directly manipulate the face shape and a wide range of customization options through control areas directly on the 3D model.

Furthermore, the model generated by Headshot already includes a complete rig for voice lipsync animations, facial expressions, and body animations.

The term "complete rig" refers to a set of virtual bones that can be controlled, allowing realistic animation of the 3D model. The complete rig for voice lipsync animations is designed to synchronize the movements of the model's lips with the audio, while facial expressions can be modified and customized using the controls included in the rig.

Additionally, this plugin offers solid compatibility with software such as Unreal, Blender, and NVIDIA Omniverse.

2.2.5 Daz3D Face Transfer feature

This is a feature of Daz Studio, a 3D modeling and rendering software specialized in character creation [25].

This feature allows you to use a single photo for a face scan, and after a calibration process of key points, these will be transferred to a basic 3D model. After creating the 3D face model, you can further customize it by adjusting parameters such as face shape,

distinctive features, and expressions. However, it requires high-quality images for optimal results.

Additionally, it is compatible with the NVIDIA Omniverse pipeline.

2.2.6 Avatar SDK

Avatar SDK [26] utilizes artificial intelligence (AI) for the creation of 3D avatars from images. It provides advanced customization options, including face and body shape, and allows for adjustments of facial features to achieve realistic results.

2.2.7 FaceGen

FaceGen [27] is a software for creating realistic 3D models of faces and heads. It offers numerous controls for modifying facial features and detailed rendering.

2.2.8 AVATAR ME++

AVATAR ME++ [28] represents an advanced method for the reconstruction of photorealistic 3D faces from individual images. This project introduces a rendering methodology that allows for realistic output even in different lighting environments. It uses Albedo and Normal maps to make the reconstructed faces highly detailed, capturing the facial skin's appearance, shadows, and reflections accurately. This method promises to capture precise facial appearance details, making 3D avatars extremely realistic and suitable for use in virtual environments and animation applications.

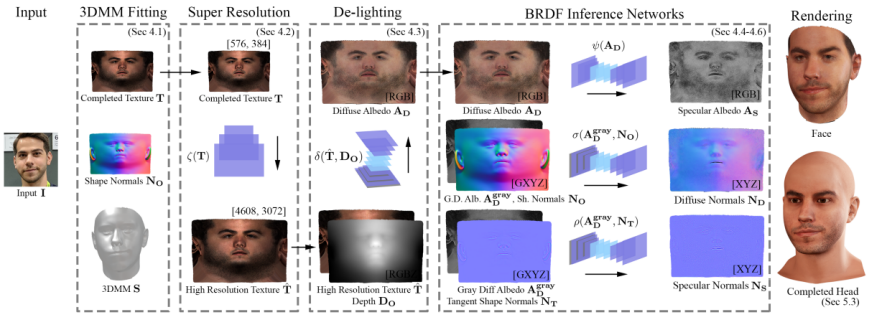


Figure 2.7: **AVATAR ME++ Architecture** is a 3D Morphable Model (3DMM) adapted to an "in-the-wild" image to create a detailed UV texture. Neural networks process normals, albedo, and face shape to realistically render the face and head in various situations. [28]

2.2.9 DAD-3D Heads

The DAD-3D Heads project [29] represents a valuable resource for 3D facial reconstruction. It is a dense, accurate, and diverse dataset designed for three-dimensional head

alignment from a single image. This dataset contains over 3,500 landmarks that accurately represent the three-dimensional shape of the head. The proposed model uses this dataset to learn head shape, expression, and pose parameters. It then employs a Fits Like A Glove (FLAME) mesh for three-dimensional reconstruction. This approach offers a high degree of accuracy and diversity, making it a promising choice for 3D facial reconstruction, although it does not address texture reconstruction.

As we can see from the figure 2.8, the Gaussian estimator predicts the coarse positions of head landmarks. The fusion block combines the coarse heatmap, the BiFPN feature map, and the output of the CNN encoder to adjust a series of parameters of the 3D head model and the finer positions of head landmarks. [29]

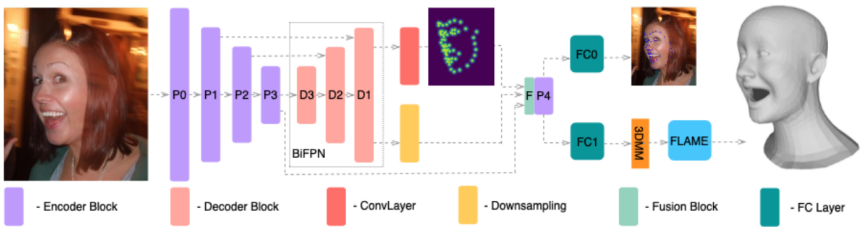


Figure 2.8: **DAD-3DNet Architecture Design**

2.2.10 OSTEC

OSTEC [30] is a project that focuses on facial texture reconstruction without the need for extensive facial texture datasets. Instead, it leverages the knowledge stored within the algorithm to achieve high-quality results. The process involves 3D rotation of the input image to obtain different views of the face. Subsequently, a 2D face generator is used to reconstruct the parts of the image that are visible in these different views. This 2D face generator relies on the knowledge stored in generative models to produce a realistic representation of the observable parts of the face. Experiments demonstrate that the completed UV textures and front-facing images are of high quality and maintain the original appearance of the facial identity. This approach offers an innovative solution to facial texture reconstruction. It is possible to observe its architecture in the figure 2.9.

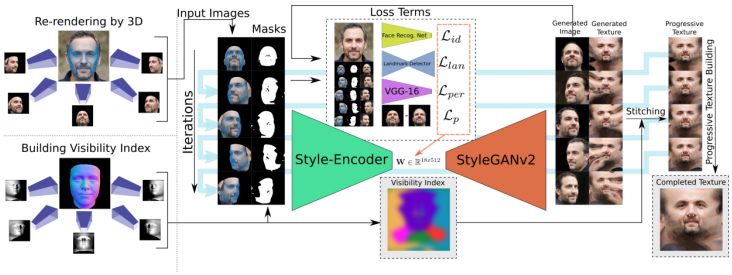


Figure 2.9: **OSTEC Architecture** The proposed approach iteratively optimizes the UV texture maps for various re-rendered images along with their masks. At the end of each optimization, the generated images are used to acquire partial UV images from dense landmarks. Finally, the completed UV images are sent to the next iteration for the progressive construction of textures. [30]

2.3 Facial Animation

Facial animation is crucial for bringing life and realism to photorealistic avatars. There are various facial animation techniques, including keyframe animation, motion capture-based facial animation, and animation based on artificial intelligence algorithms like neural networks.

These techniques involve the use of animation software. Below is a list of prominent software for facial animation.

2.3.1 iClone

iClone [31], developed by Reallusion, uses motion capture to create realistic facial animations. It can record the movement of an actor and apply it to a virtual character in real-time, ensuring accurate lip synchronization (lypsync animation) and emotional expressions.

2.3.2 Audio2Face

Developed by NVIDIA, it's an AI-based facial animation software that allows for automated facial animations generated from audio recordings [32].
The input audio is then fed into a pre-trained neural network, and the output will drive the 3D vertices of the character's mesh, creating real-time facial animation.

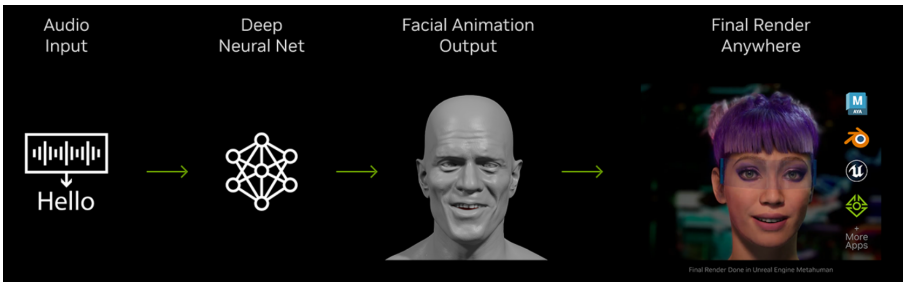


Figure 2.10: From audio to animation with ease, thanks to generative AI. [32]

2.3.3 Faceware

Faceware is a company that has developed markerless facial mocap software that works in real-time and allows for precise capture of an actor's facial expressions and transferring them to 3D models.
It is widely used in the film and video game industry to capture realistic facial performances. [33]

2.4 Creation Avatar Softwares

In the context of creating photorealistic avatars, the current landscape offers advanced platforms and software solutions, with **MetaHuman** and **Character Creator** being of the leading options.

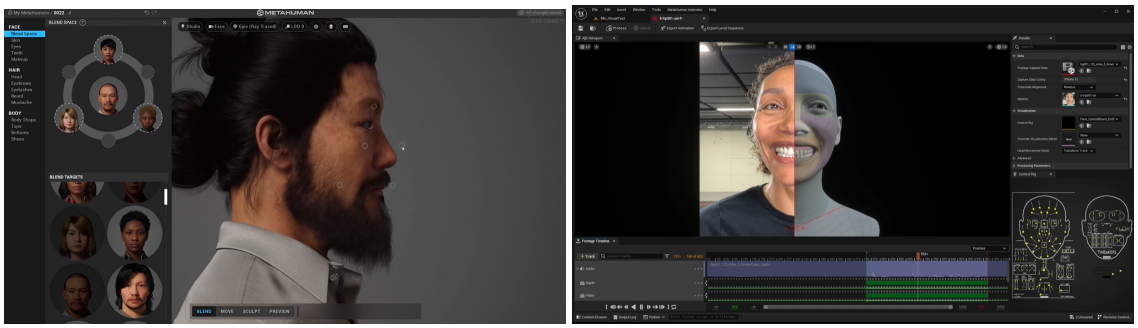
2.4.1 Metahuman

MetaHuman is a technology developed by Epic Games, representing a significant step in the generation of exceptionally high-quality and realistic digital characters.

MetaHuman Creator [34], the core component of MetaHuman, is at the heart of this innovation. This platform, accessible through a cloud-based application, allows for the generation of photorealistic avatars in a matter of minutes. This is a remarkable breakthrough from traditional methods, enabling content creators to quickly and intuitively create highly realistic digital human characters.

The efficiency of MetaHuman Creator extends to its flexibility. Users can start with predefined character presets or import custom meshes using the Unreal Engine plugin. In fact, the production pipeline for a MetaHuman revolves around MetaHuman Creator, which allows for creating various characters by combining different predefined features.

These MetaHumans created in MetaHuman Creator can be imported into Unreal Engine projects via Quixel Bridge, a plugin directly integrated into UE5.
This flexibility provides endless creative possibilities, giving developers and artists the freedom to bring their own visions to life.



(a) Metahuman Creator (b) Metahuman Animator

Figure 2.11

MetaHuman Animator [35] is another key component designed to capture high-fidelity facial performances on any MetaHuman character. This tool is capable of capturing every nuance of actor performances and convincingly transferring them to digital characters. This results in engaging and believable experiences for the audience.

Facial animation with MetaHuman Animator does not require complex equipment. Common devices like the iPhone can be used to achieve high-quality results, making the process accessible to a wide range of creators. This flexibility in performance capture allows for adapting the hardware to the specific needs of the project, significantly improving the process of creating MetaHumans and customized digital characters.

MetaHuman offers an efficient workflow that can be easily integrated into any existing content creation process. Creating a MetaHuman takes only a few minutes, significantly simplifying the digital character development process. Furthermore, MetaHumans can be exported to Digital Content Creation (DCC) software like Maya and Blender for further customization and made ready for real-time use in Unreal Engine, opening up new creative possibilities for content creators of all kinds.

MetaHuman goes beyond character creation, offering advanced capabilities for integrating realistic elements. Through the implementation of ray tracing and simulation models like subsurface scattering, it is possible to simulate details such as hair, fur, eyelashes, beard, and even achieve a high level of color fidelity, utilizing textures for wrinkles and blood flow beneath the skin.

2.4.2 Character Creator

Character Creator [36], developed by Reallusion, stands out as a versatile and comprehensive solution for shaping high-quality digital characters, aligning itself with the user-friendly approach of MetaHuman. This technology distinguishes itself through an intuitive interface that streamlines the creation of detailed and lifelike characters.

The flexibility of Character Creator is evident in its ability to adapt to user needs. Users

can either kickstart their designs with predefined presets or import custom meshes seamlessly using dedicated plugins.

Furthemore, The platform excels in crafting photorealistic details, managing elements like hair, fur, eyelashes, and beard with advanced tools. The platform provides an authentic reproduction of textures for wrinkles and blood flow beneath the skin, contributing to the authenticity of characters.

Character Creator seamlessly integrates into the pipelines of other leading tools, allowing users to effortlessly import characters into projects like Unreal Engine 5 and Unity. Additionally, Character Creator facilitates integration with digital content creation (DCC) software such as Maya and Blender, enabling further customization and preparing characters for real-time use.

Designed to deliver engaging experiences, Character Creator enables the creation of characters capable of conveying highly realistic facial expressions. A notable aspect is the ability to animate the face through ***iClone***, another software by Reallusion that, without the need for complex equipment, allows for high-quality results.

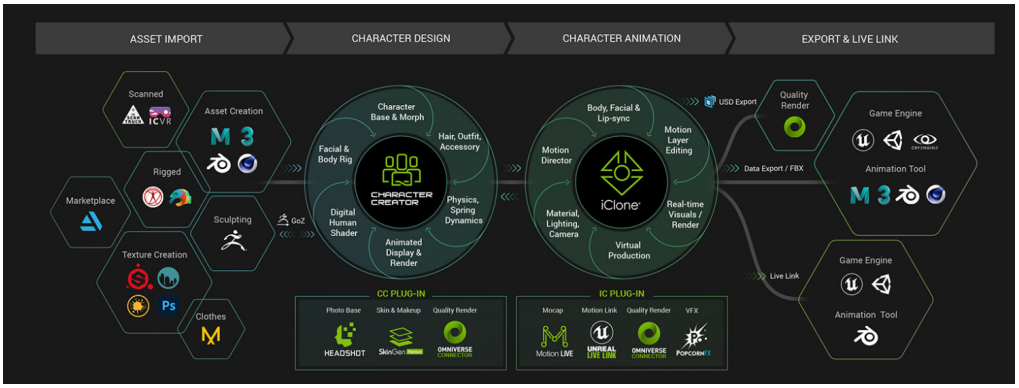


Figure 2.12: ***iClone*** and ***Character Creator*** seamlessly work together as one big design-to-animation character platform and greatly ease the in-and-out productivity with many other 3D applications.

In summary, Character Creator is a robust choice for crafting photorealistic avatars, seamlessly integrating into workflows with its rich features and flexibility. Its intuitive interface and exceptional detailing make it a standout player in digital character creation.

Chapter 3

Automation in Creating Synthetic Humans

My research work, in close collaboration with **Rai R&D**, as mentioned earlier, is oriented towards the optimization and automation of the workflow for the creation of Synthetic Humans.
The starting point of my research was a thorough analysis of the workflow used by RAI to create a representation of "Maria Callas".

This workflow was developed for the project presented at the International Broadcasting Convention (IBC), a renowned global event that serves as an exhibition platform for the latest trends and discoveries in the field of media and technological innovation.

RAI participated in the **"Synthetic Humans For Entertainment And Accessibility"** project as part of the IBC2023 Accelerator Media Innovation Programme [37] [38]. This project focuses on two distinct use cases based on Synthetic Humans. The first case relates to the field of entertainment, with the goal of bringing the Synthetic Human of Maria Callas onto the stage. The second case focuses on accessibility, proposing a photorealistic sign language interpreter to address accessibility challenges in broadcasting.

RAI has focused its efforts on the first use case, aiming to breathe new life into the RAI archive by transforming it into digital avatars. This strategic choice aligns with the prospect of working with economies of scale, emphasizing the goal of automation to facilitate the efficient and scalable production of Synthetic Humans. Below is presented the representation of the workflow for creating the Synthetic Human of Maria Callas, constructed from images extracted from the RAI archive.

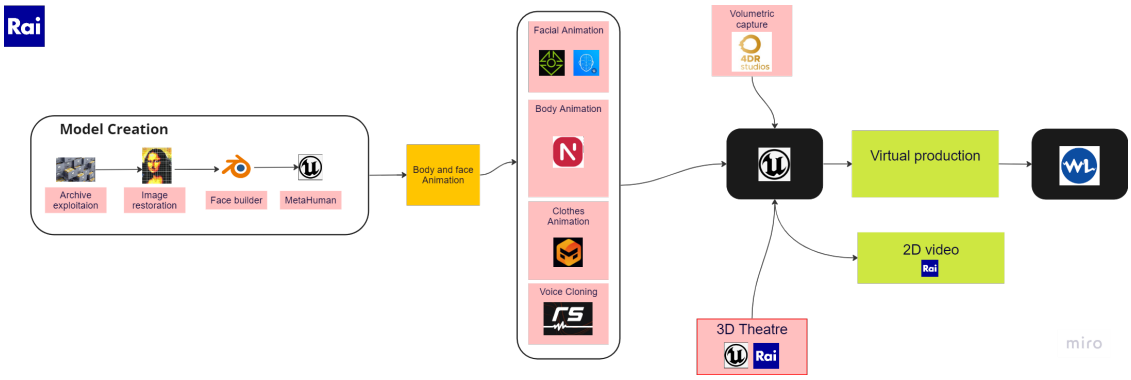


Figure 3.1: The workflow used by RAI for the realization of the project **"Synthetic Humans For Entertainment And Accessibility."**

3.1 Synthetic Humans Workflow

Let's begin this chapter by exploring the key element in the process of creating Synthetic Humans, which is the phase of 3D model creation. The goal of this phase is to create a Synthetic Human that closely resembles the reference character we are trying to depict while also aiming to achieve the highest level of automation possible.

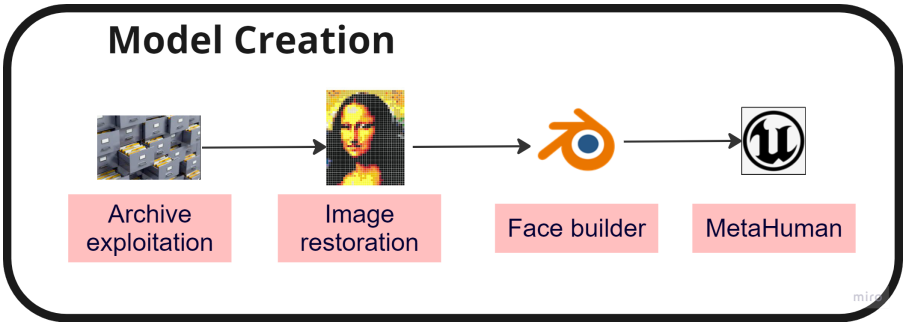


Figure 3.2: Workflow used by RAI for the Creation of the 3D Model of Maria Callas

The highest level of automation for this phase is represented by the ability to use a video or images as input and obtain the 3D model of the Synthetic Human of the reference character as output.

This is the goal proposed by Rai, and we will explore various aspects of how such automation can be achieved.

3.1.1 Archive Exploitation: Using the RAI Archive

The process of creating 3D models begins with a crucial phase: collecting materials, consisting of images and videos of the character intended to become our Synthetic Human.

In the context of this project, material is drawn from the valuable RAI archive.

RAI provides a rich heritage of visual material collected over decades of productions, including TV programmes, photos and other material. This project was born with the goal of maximizing the use of RAI's archive resources to bring historical figures back to life through virtual avatars, and this archive is a valuable treasure for achieving this objective.

To simplify and automate the search for the necessary images and videos in this process, RAI has developed an innovative solution called the ***"Smart Gallery"***. This advanced platform leverages face analysis techniques based on clustering and deep learning methods, allowing for the automated identification of similarities between faces.

Thanks to this cutting-edge technology, it's possible to conduct targeted searches and easily retrieve relevant videos and images extracted from the archived contents involving the character of our interest. This represents the first step towards automation and serves as the starting point for transforming 2D images into precise and realistic 3D models. The preview screen of the RAI Multimedia Catalog can be seen below.

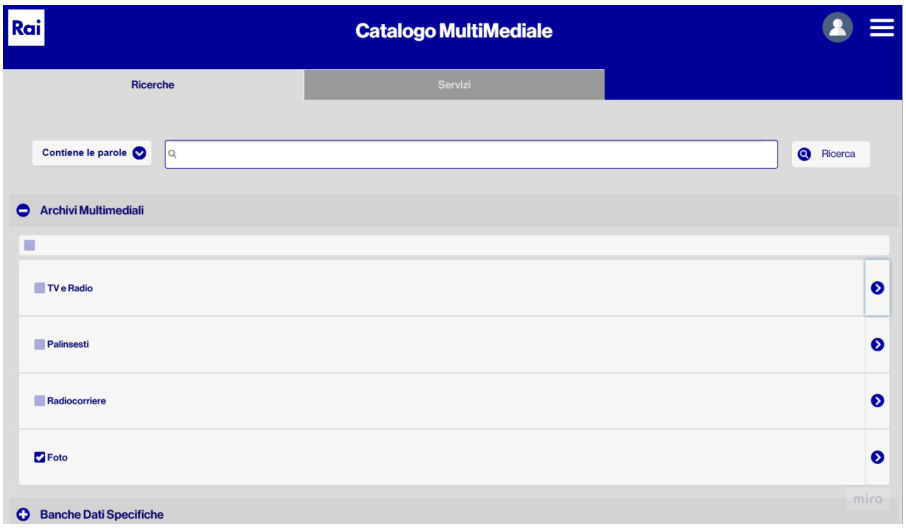


Figure 3.3: RAI Multimedia Catalog

However, the images and videos in the RAI archive, especially content from the past, often have rather poor quality, characterized by low resolution and the presence of noise. An additional challenge is presented that must be overcome to achieve high-quality results.

3.1.2 Orchestrator: Automation in 3D Face Reconstruction

After completing the material collection phase from the archive, my primary goal is to automate the "3D Face Reconstruction" process as much as possible, which involves creating 3D models of faces from images or videos.

The collaboration with **PluxBox** [39], a valuable partner of RAI, within the IBC project "Synthetic Humans For Entertainment And Accessibility," has led to the creation of a fundamental component: the Orchestrator.

The Orchestrator is an advanced technological solution that acts as a bridge between the various stages that follow the material collection and the creation of the 3D face model.
Thanks to the Orchestrator, we completely automate the "3D Face Reconstruction" process, turning a complex procedure into a smooth and efficient workflow.

A representation of the Orchestrator is provided in Figure 3.4 to facilitate a complete understanding of its functionality.

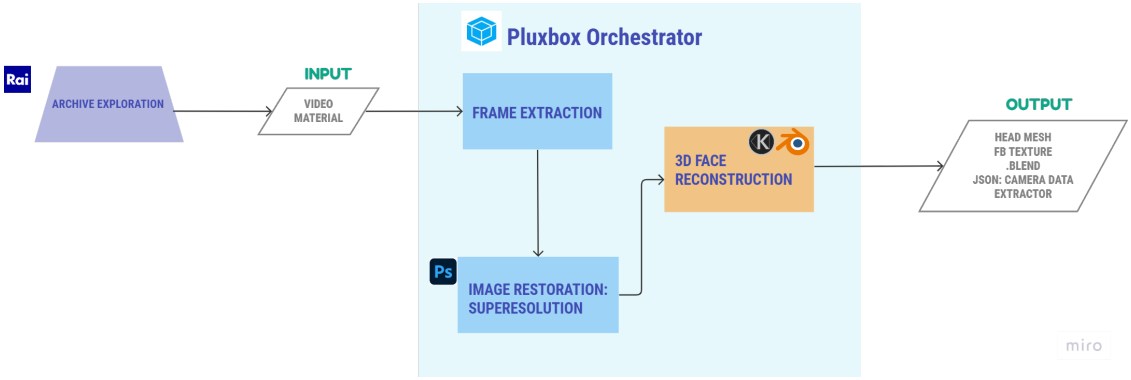


Figure 3.4: Orchestrator structure

By observing the chart, it is easy to understand how it works: by taking videos as input obtained through the Archive Exploitation process, and through a series of automated processes such as Frame extraction, Image Restoration, and **my work on the 3D face reconstruction** module it converges into the creation of the 3D mesh of the face of the target subject.

But let's take a closer look at these automated processes within the Orchestrator.

1. **Frame extraction: Capturing Key Images**

   The Frame Extraction phase enables the extraction of frames from the input videos. This delicate and crucial operation, representing the initial step in creating the 3D mesh of the face, is facilitated by the Orchestrator, although manually performed by the user.

   The main goal of this operation is to capture frames that exclusively depict the target subject, avoiding the presence of other individuals in the same frame.

   Equally essential is to ensure the diversity of the selected frames. This variety of images is of crucial importance as it allows capturing the character's face from various angles. This diversification is fundamental to ensure that the reconstruction of the *head mesh* closely reflects the reference subject.

   In the figure below 3.5 you can see the interface of the orchestrator in the Frame extraction phase.

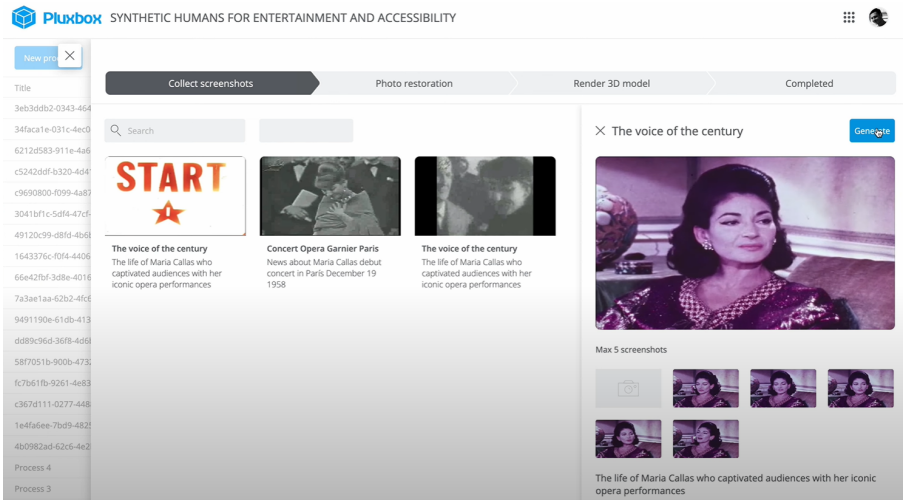


Figure 3.5: Synthetic Human Demo: Frame Extraction

2. **Image Restoration: Super Resolution with Photoshop API**

   In the image restoration stage, I focused on improving the quality of the frames extracted in the previous stage from videos obtained from the RAI archive. It is important to note that these frames are often of low quality, characterized by low resolution and noise, which can affect the accuracy of 3D reconstruction.

   To overcome these limitations, we employ an image restoration phase aimed at enhancing the visual quality of the frames. In particular, we use an upscaling technique, leveraging the APIs provided by **Adobe Photoshop** [15].

Upscaling is an advanced process that enables an increase in the resolution of images, enlarging them without compromising quality or clarity. This approach is crucial to ensure that the captured frames retain all essential details, contributing to an even more accurate and realistic 3D face reconstruction.

In the current phase, the Orchestrator utilizes the Photoshop APIs to send the frames that were previously extracted. In response, it receives images that have undergone an upscaling procedure, thus enabling the use of high-resolution frames for the subsequent 3D Face Reconstruction phase.

In the figure below 3.6 you can see the interface of the orchestrator for the Super Resolution phase.

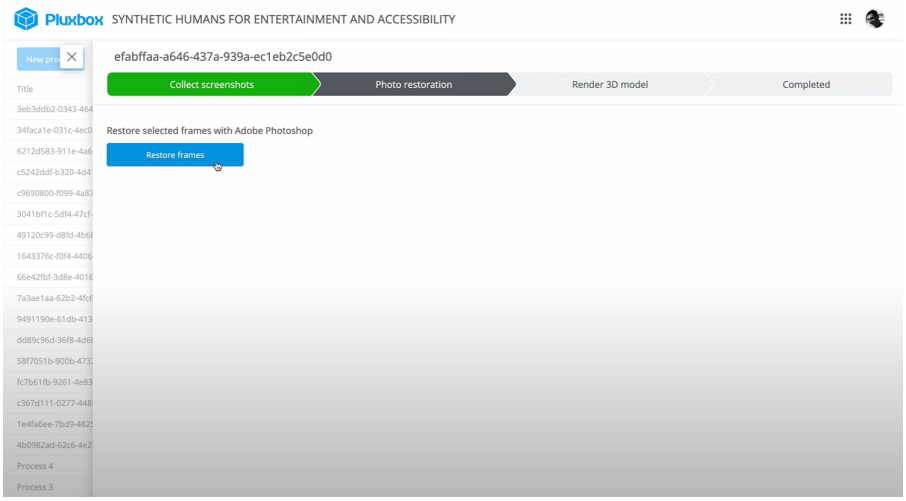


Figure 3.6: Synthetic Human Demo: Super Resolution

3. 3D Face Reconstruction: FaceBuilder

In our journey towards creating Synthetic Humans, we reach a crucial stage: 3D Face Reconstruction. In this phase, 2D images are transformed into 3D models of the faces of reference characters.

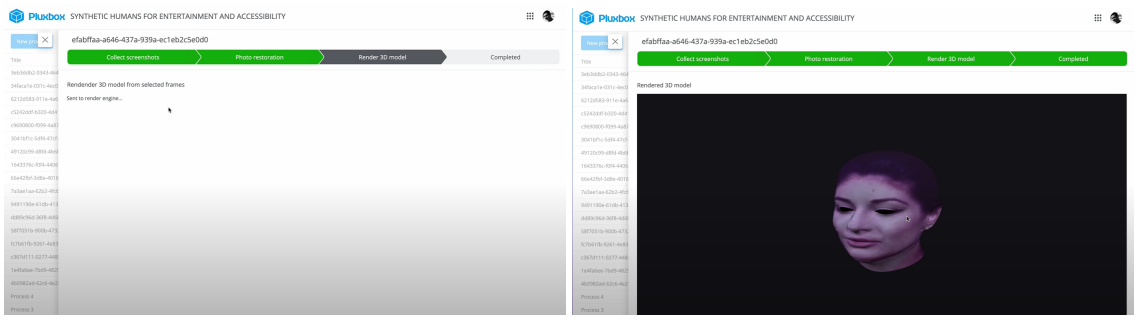
To simplify and automate this complex process, I developed a script called ***"FB_Head_Reconstruction_Automatized.py"*** in Python, which leverages some of the powerful features of FaceBuilder, a Blender plugin created by KeenTools. For more detailed information, you can refer to Chapter 2 in the "Blender Face Builder Plugin" section.

With the combined use of the script and FaceBuilder, transforming images depicting our subjects into 3D models of their faces becomes an effortless procedure. This allows for a highly automated 3D Face Reconstruction process, ensuring the creation of a high-quality base mesh of the face.

To integrate the "FB_Head_Reconstruction_Automatized.py" script, which is dedicated to automating the 3D Face Reconstruction, into the Orchestrator, I had to implement a service called ***"FaceG3n"***.
The FaceG3n service is hosted on a dedicated server and facilitates communication between the Orchestrator and the "FB_Head_Reconstruction_Automatized.py" script through REST APIs.

In the following section there is an in depth description of the script to fully understand how they work.

In the figure below 3.7 you can see the interface of the orchestrator for the Face Reconstruction phase.



(a) Creating the 3D Model (b) Final Result

Figure 3.7: Synthetic Human Demo: Face Reconstruction

3.2 Automation: 3D Face Reconstruction

3.2.1 Why rely on FaceBuilder?

Creating realistic 3D models of human faces is a challenge that requires specific skills and the use of appropriate tools to ensure high-quality results. In this context, KeenTools' FaceBuilder plugin emerges as an excellent choice for 3D face reconstruction.

As discussed in Chapter 2, FaceBuilder is a plugin designed for Blender that greatly simplifies the process of creating 3D models of human faces from a set of reference photos [22] [23].
This plugin stands out for its ability to automatically analyze the information present in the photos, leveraging artificial intelligence to configure camera virtual cameras. This makes it possible to work with photos of any size and to handle expressions non-neutral facial expressions with ease. In addition, FaceBuilder was specially designed to meet the needs of the creators of Metahuman. [40]
This plugin also allows 3D models to be exported to formats compatible with Unreal Engine, offering broad versatility and also supporting UV mapping of the mesh, for texture, suitable for Metahuman.

Thus, the choice of FaceBuilder for 3D face reconstruction is motivated by several key reasons, such as automatic alignment of facial points and compatibility with Metahuman. But to further optimize this mesh creation step, I developed a script called "FB_Head_Reconstruction_Automatized.py" that automates the manual FaceBuilder steps, eliminating the need to use Blender and greatly simplifying the process.
Script functionality includes creating the "head," inserting images, aligning virtual cameras for each image, and creating a texture in accordance with Metahuman-specific UV mapping.

The main goal of the script "FB_Head_Reconstruction_Automatized.py" is to offer a fully automated process for generating high-quality 3D models within the Orchestrator. This approach makes the entire Face Reconstruction process accessible to users of any experience level, thus helping to further simplify the overall workflow.

3.2.2 Implementation: *FB_Head_Reconstruction_Automatized.py*

In this section we are going to examine the practical implementation of the script "***FB_Head_Reconstruction_Automatized.py***".
Before going into implementation details, I will briefly discuss what it means to run a script in batch mode and the context in which this mode is particularly advantageous.

What is *Batch Mode*?

Execution in *batch mode* represents a mode of execution of a program or script in which user interaction is minimized or completely eliminated.
In other words, the program or script is executed automatically without requiring constant input or confirmation from the user.

This mode is extremely useful when you want to automate processes or execute commands efficiently, without the need for manual intervention.
In the context of my thesis, batch mode execution allows us to use the full functionality of Blender and the FaceBuilder plugin without having to interact with the interface.

Running the Script via Command Prompt

The script can be run through the command prompt using the following instruction:

```
blender -b -P "path\FB_Head_Reconstruction_Automatized.py" -- "path\imgs"
```

Let us examine each component of this command:

- `blender`: This part of the command represents the execution of the Blender program. **Blender** is a widely used 3D graphics software and offers a wide range of features for modeling, animation and rendering.
- `-b`: The -b option is a specific Blender command line option. When used, it tells Blender to run in batch mode. In other words, Blender is started without the graphical user interface (GUI) and can be controlled by scripts, making it ideal for process automation.
- `-P "path\FB_Head_Reconstruction_Automatized.py"`: This part of the command specifies the path of the Python script to be executed within the Blender environment.
- `--`: The double dash – acts as a separator in the command, indicating that what follows represent the arguments for Blender and not additional options.
- `"path\imgs"`: This part of the command represents the path to the directory containing the images needed to run the script. Blender will use this path to locate and load the images as part of the 3D model creation process.

In summary, the command starts Blender in batch mode and instructs the program to execute the Python script "FB_Head_Reconstruction_Automatized.py" located at the specified path.
Blender will use the directory containing the images as input for script execution, allowing for a fully automated process.

This batch mode ensures that all the features of Blender and the FaceBuilder plugin are used without requiring constant user interaction, making the process highly efficient and repeatable.

Stages of Script Execution

The process of executing the "FB_Head_Reconstruction_Automatized.py" script consists of several key steps.

The script begins by verifying that the FaceBuilder addon is installed and working properly. If successful, it proceeds by creating a new scene in Blender and removing the default elements to prepare the scene for the `head_reconstruction_automatized` stage.

This phase encapsulates all the basic steps for 3D Face Reconstruction.

1. **Adding a "New Head"**
   The `head_reconstruction_automatized` step begins with the addition of a "New Head," which is the starting point for the creation of the 3D model of the head. The script is then used to create the 3D model of the head. In this step, the script calls the function provided by FaceBuilder to begin the modeling process.

2. **Loading Images**
   Next, the script handles the loading of the images used as reference for creating the model.
   However, since FaceBuilder does not natively provide a method for loading images via script, I implemented the `add_images` function, you can see it in figure 3.8, to automate this step.
   This feature provides full script automation, without the need for manual interactions for uploading images.

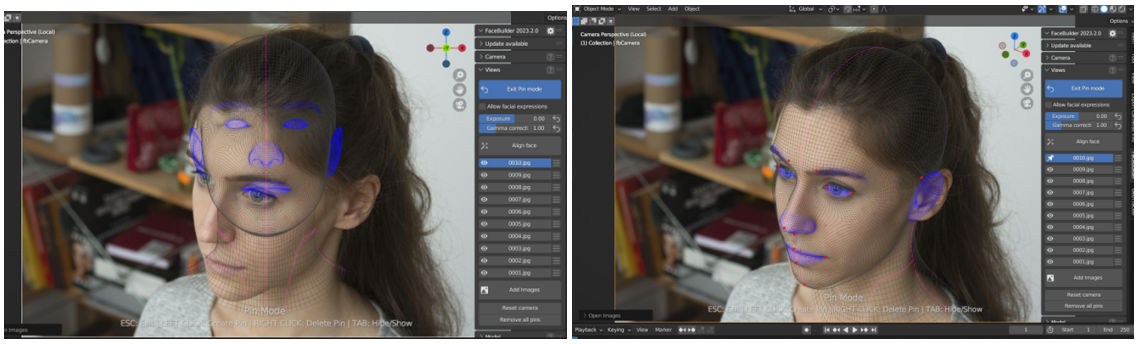
```python
def add_images(images_directory):
    global num_img_loaded  # Declare variable as global
    head_num=0

    # Get FaceBuilder settings
    settings = get_fb_settings()
    if not settings.is_proper_headnum(head_num):
        print(f'WRONG HEADNUM: {head_num}/'f'{settings.get_last_headnum()}')
        return {'CANCELLED'}

    if not settings.check_heads_and_cams():
        settings.fix_heads()
        return {'CANCELLED'}

    FBLoader.load_model(head_num)

    head = settings.get_head(head_num)
    last_camnum = head.get_last_camnum()

    # Iterate through files in the images directory
    for filename in os.listdir(images_directory):
        if filename.lower().endswith(('.jpg', '.jpeg', '.png', '.tif', '.tiff', '.bmp')):
            filepath = os.path.join(images_directory, filename)
            print(f'IMAGE FILE: {filepath}')

            # Add new camera with image
            camera = FBLoader.add_new_camera_with_image(head_num, filepath)
            read_exif_to_camera(head_num, head.get_last_camnum(), filepath)
            camera.orientation = camera.exif.orientation

            # Increment the number of loaded images
            num_img_loaded += 1

    for i, camera in enumerate(head.cameras):
            if i > last_camnum:
                auto_setup_camera_from_exif(camera)
                FBLoader.center_geo_camera_projection(head_num, i)

    FBLoader.save_fb_serial_and_image_pathes(head_num)
    print(f'\nAll images are saved\n\n')
```

Figure 3.8: function: *add_images*

3. **Aligning Virtual Cameras**
   Next comes the phase of automatic alignment of virtual cameras.
   In this step, the virtual cameras associated with each image are aligned so that they are oriented, positioned and set with the necessary focal length to achieve a perfect match between the images and the 3D model mesh.
   Alignment is a critical process to ensure the quality of the final result.

   So I developed a function, `align_face_to_images`, to perform the alignment for each uploaded image. First I change the mode of the images to *Pin Mode* and then I call the FaceBuilder function that aligns the camera to the corresponding image. To make this step clearer, in figure 3.9. I show what we would see graphically, for

   a single image, if we performed this step within Blender.



(a) Pin Mode Enabled (b) Alignment Completed

Figure 3.9

4. **Creation of Texture**
   Next, the script handles the creation of the texture of the 3D head model.
   Before starting the texture creation, some optimizations are made to ensure a more photorealistic and accurate appearance.
   These optimizations include excluding the areas related to the eyeballs, the inside of the mouth, and the lower part of the neck from the mesh and consequently from the texture.

   In addition, the UV mapping of the mesh, is modified to ensure a match with Metahumans.
   The texture creation process is implemented in the `bake_texture_create` function, which, after performing the previously exposed steps, extracts visual information from all the images used during the alignment phase. These images are "mixed" by a bake process on the mesh, generating a detailed texture that will be applied to the model.

5. **Saving Results**
   Finally, we have a function, `export_data`, that handles the saving of the results within a dynamically created folder in the same location as the folder with the images used to make the model.

   The ***output files*** are:

   - the texture in .png format
   - the head mesh in .fbx and .glb formats
   - the .blend file to be able to allow editing of details on the mesh
   - a JSON file containing information about the virtual cameras

   To extrapolate camera information, I implemented the function `save_cameras_info` that extracts information such as position, orientation, focal length, and resolution, and saves it in a JSON file.

```python
def save_cameras_info(output_json_path):
    # Extract camera information for Unreal
    camera_info = []

    for obj in bpy.data.objects:
        if obj.type == 'CAMERA':
            # Convert location from meters to centimeters and handle Y coordinate sign
            location_cm = {"X": obj.location.x * 100, "Y": abs(obj.location.y) * 100, "Z": obj.location.z * 100}

            # Calculate field of view from focal length
            sensor_width = obj.data.sensor_width
            focal_length_mm = obj.data.lens
            fov = 2 * math.degrees(math.atan(sensor_width / (2 * focal_length_mm)))

            # Get camera resolution
            resolution_x = bpy.context.scene.render.resolution_x
            resolution_y = bpy.context.scene.render.resolution_y

            rotation_degrees = {"Pitch": math.degrees(obj.rotation_euler.y),
                                "Yaw": math.degrees(obj.rotation_euler.z),
                                "Roll": math.degrees(obj.rotation_euler.x)}

            camera_data = {
                "Name": obj.name,
                "Location": location_cm,
                "Rotation": rotation_degrees,
                "FocalLength": focal_length_mm,
                "Filmback": [resolution_x, resolution_y]
            }

            camera_info.append(camera_data)

    # Save camera information to a JSON file
    with open(output_json_path, "w") as file:
        json.dump(camera_info, file, indent=4)

    print(f"Camera information extracted and saved to '{output_json_path}'.")
```

Figure 3.10: function: *save_cameras_info*

The utility of the JSON file will be explored in more detail in a second step, as it will be used to make a comparison between the Synthetic Human obtained at the end of the new workflow, and the source images with which it was generated.
Through the JSON file, it will be possible to extract the information about the cameras, to reproduce the same view and angle as the photos to portray the Synthetic Human.

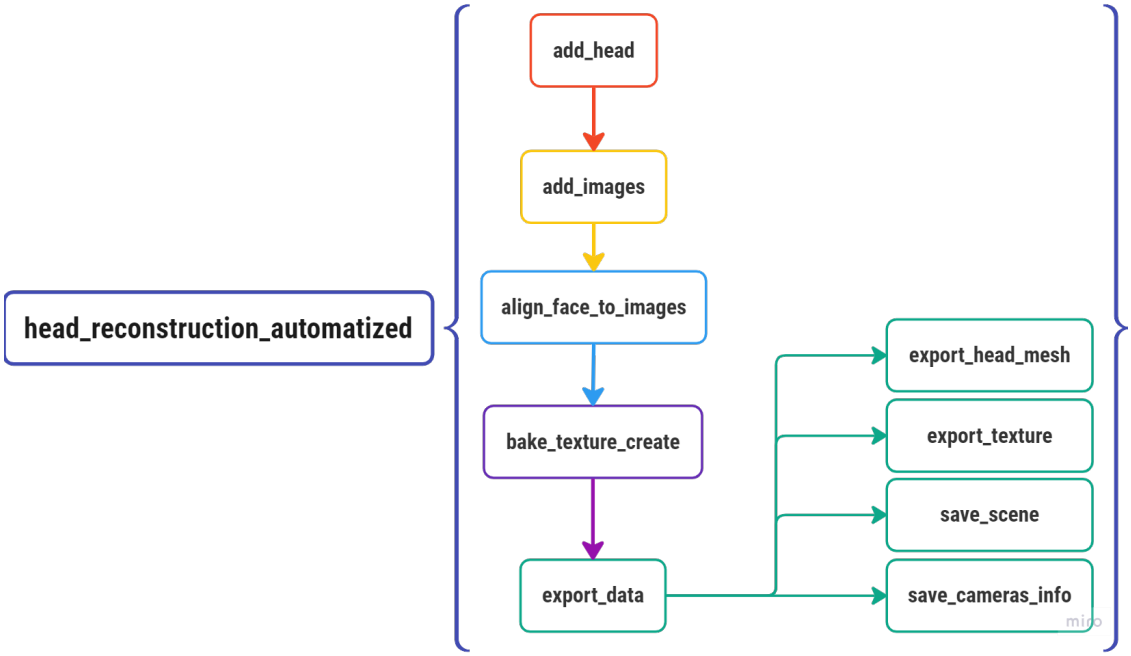


Figure 3.11: General scheme of the script *FB_Head_Reconstruction_Automatized.py"*

3.3 Implementation of REST APIs: FaceG3n

REST APIs, which stands for Representational State Transfer Application Programming Interfaces, are a fundamental tool in software development that enables communication and data exchange between different applications or systems. These APIs allow interaction with the services provided by an application or a remote server.

In the context of our project, REST APIs play an essential role in coordinating the execution of the script *"FB_Head_Reconstruction_Automatized.py"* and all the related activities, aiming to achieve the desired results in an automated manner.

3.3.1 FaceG3n

FaceG3n is the service I have developed to implement the REST APIs in our project. This service plays a crucial role in managing the interaction between the Orchestrator and the 3D head creation script.

The primary objective of FaceG3n is to enable asynchronous communication between the Orchestrator and the 3D head creation script. To achieve this goal, the service receives data sent by the Orchestrator and performs all the necessary tasks to prepare the execution of the script. Once these preliminary tasks are completed, FaceG3n initiates the script asynchronously and manages the process of 3D head generation.

Taking into account the service's requirements, such as efficient management of CPU-intensive tasks, optimized resource usage, and robust stability and concurrency management, the **SpringBoot framework** was chosen for the development of the service:

1. **Handling Long-Duration Tasks:** Spring Boot was chosen for its efficient handling of long-duration tasks. The service can benefit from robust concurrent operation management, making it ideal for executing CPU-intensive operations that take several minutes to complete.

2. **Optimized and Stable Resource Usage:** The choice of Spring Boot was influenced by the need to use resources optimally and balance them. In tasks that also require several minutes of processing, precise control of memory and threads becomes crucial to prevent resource overutilization or stability issues.

3. **Robust Concurrency Management:** Another key factor in choosing Spring Boot was its robust concurrency management. It can efficiently and reliably handle multiple concurrent requests. This feature is essential when dealing with CPU-intensive operations, ensuring that multiple requests do not interfere with each other and can be executed in parallel without issues.

4. **Exception Handling and Stability:** Spring Boot offers an advanced exception handling system, contributing to service stability. Structured exception and error handling reduces the risk of unexpected behavior during the execution of long-duration tasks. This is particularly important to ensure that the service operates reliably and predictably, even during complex and CPU-intensive operations.

3.3.2 FaceG3n Structure

Controller, Service and Properties

The structure of FaceG3n is divided into two main components:

- **Controller**: This part of the service defines the REST interfaces exposed by the application. The controller handles HTTP requests, defining methods to handle incoming requests, including the main endpoint that accepts data sent by the Orchestrator.

- **Service**: The service is the core of FaceG3n. Here, all the operations necessary to manage the execution of the 3D head creation script are implemented. This includes downloading incoming images, processing them, and placing them in a specific directory.

In addition to the two components mentioned, there is a third component crucial to the program's operation: **Properties**.
Within the Spring Boot configuration files, properties necessary for the service's proper functioning are defined. These properties include the main paths and settings for communication with the Orchestrator.

In the next figure 3.12 you can observe the Scheme of the organization of FaceG3n's components and functions.

Main Tasks

1. **Preliminary Tasks:** Before executing the main script, FaceG3n performs a series of preliminary tasks. These tasks include downloading images sent by the Orchestrator, processing them, and placing them in a specific directory.

2. **Script Invocation:** Once the preliminary tasks are completed, FaceG3n asynchronously invokes our script "FB_Head_Reconstruction_Automatized.py." This script is responsible for creating 3D heads from the received images. Upon completion, the script generates a .glb file containing the 3D head model.

3. **Results and Sending:** After running the script, FaceG3n handles the obtained result. The service sends the requester a URL to download the resulting .glb file, along with the associated process ID. This allows the Orchestrator to retrieve the process result and proceed with the subsequent stages of the workflow.

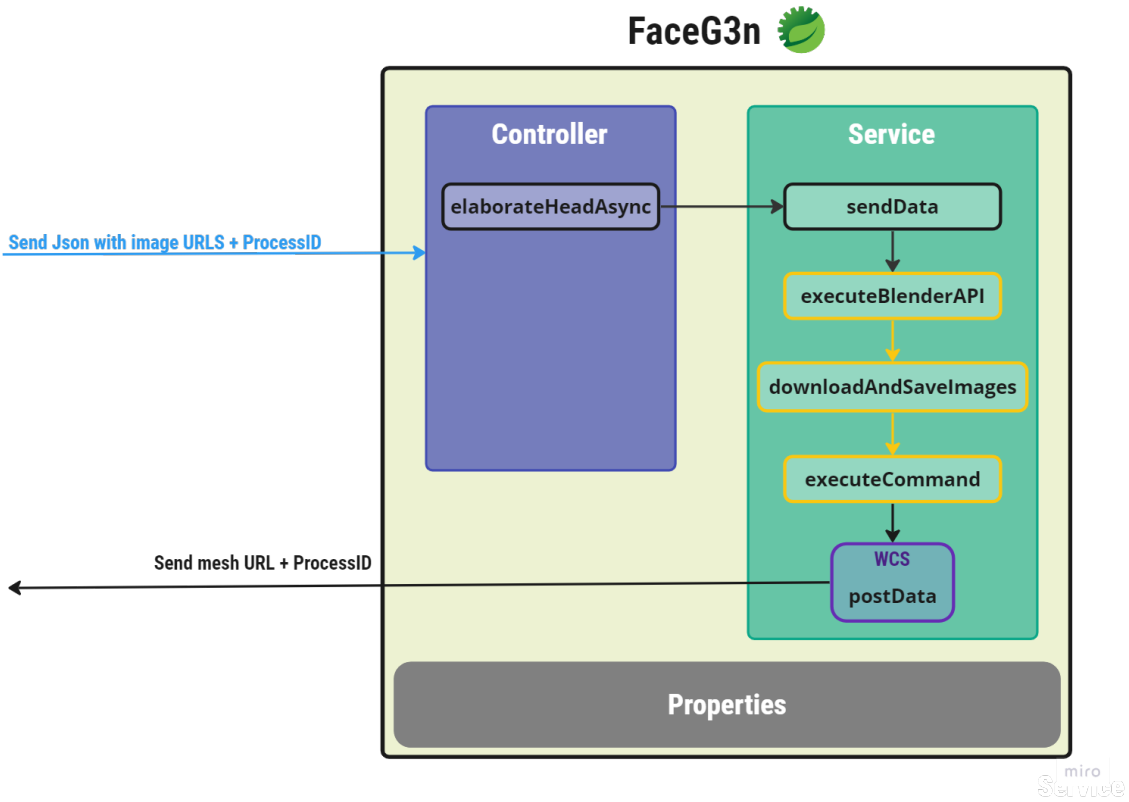


Figure 3.12: Outline of the organization of components and functions of **FaceG3n**

3.3.3 Implementation of FaceG3n

Hosting and Access Points

FaceG3n exposes the endpoint `elaborateHeadAsync` on port 8080. This endpoint is the interface through which the Orchestrator communicates with the service.

```
@PostMapping("/elaborateHeadAsync")
public ResponseEntity<?> elaborateHeadAsync(@RequestBody ImageDataRequest request) throws IOException {
    if(request==null || request.getFrames().isEmpty()) {
        return ResponseEntity.status(HttpStatus.BAD_REQUEST)
                .contentType(MediaType.APPLICATION_JSON)
                .body(new ResourcesResponse("The JSON request is empty or malformed"));
    }else {
        CompletableFuture.runAsync(() -> {
            // this method is non-blocking, async
            faceG3nService.sendData(request);
        });
        ResponseEntity<Object> responseEntity = ResponseEntity.status(HttpStatus.OK)
                .contentType(MediaType.APPLICATION_JSON).body(null);
        return responseEntity;
    }
}
```

Figure 3.13: Endpoint: *elaborateHeadAsync*

The PluxBox Orchestrator will send a POST request to the following endpoint: `http://108.142.221.68:8080/elaborateHeadAsync`. It will send as input a JSON containing a `process_id` and a list of image URLs. Through the Controller and its related Service, FaceG3n will asynchronously return a JSON containing the `process_id` and the URL reference to the obtained mesh. Example of the JSON received as input:

```
{
  "process_id": "64dcbb97872ad4ba253b4725",
  "frames": [
    { "url": "https://i.imgur.com/1qxUJmL.jpg" },
    { "url": "https://i.imgur.com/R6VianN.jpg" },
    { "url": "https://i.imgur.com/wbCYGL4.jpg" }
  ]
}
```

Start of the Process

Upon receiving a request, FaceG3n verifies that the request is not null and that the list of URLs is not empty.
If affirmative, the method returns a response with a status code of 200 to indicate that the request is valid and has been taken into consideration. The data will then be sent to the Orchestrator via an asynchronous call.

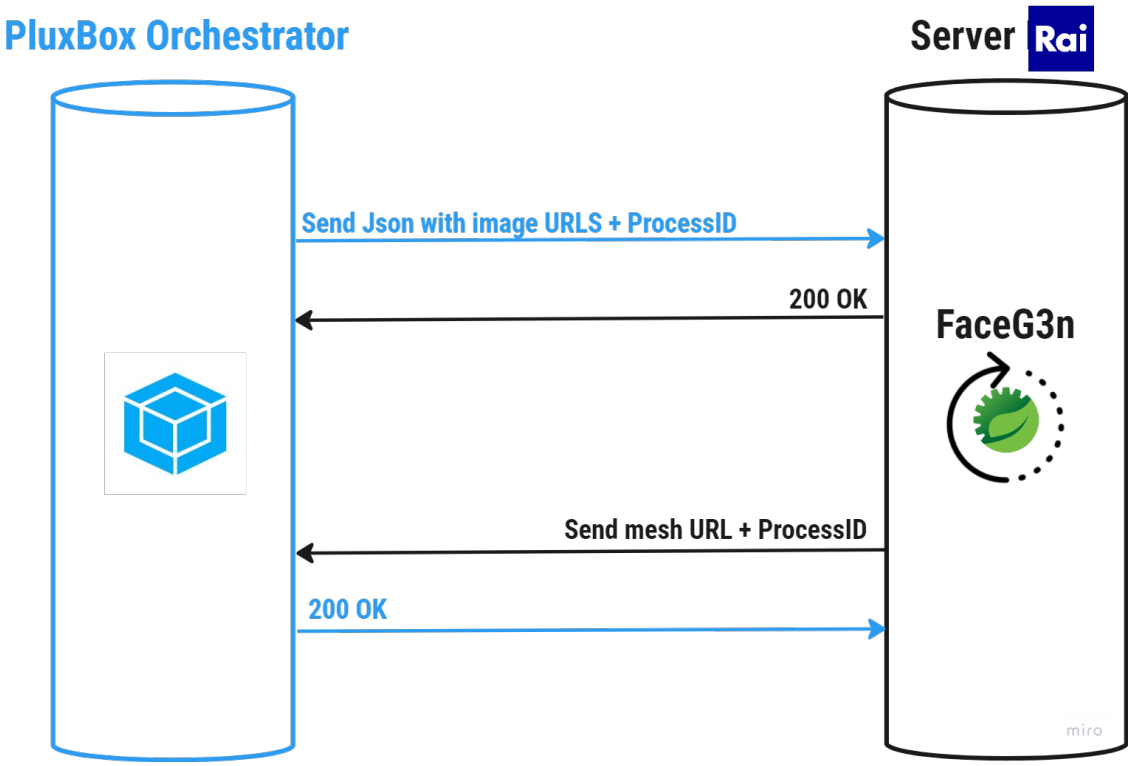


Figure 3.14: FaceG3n: REST API Workflow Diagram

Preparation and Image Download

Within the `executeBlenderAPI` function, the call to the `downloadAndSaveImages` function is located. In this function, the input images are downloaded and saved in a specific folder identified by the process_id provided in the JSON. Before saving them, the service checks and deletes any previously existing images in the same directory. The downloaded images are in the ***jpg*** format and are renamed using Universally Unique Identifiers (UUIDs) generated to ensure unique names.

```
public String executeBlenderAPI(ImageDataRequest request) {
    String imagesFolderProcessId = scriptsPathimgs +File.separator+ request.getProcessId();
    downloadAndSaveImages(request, imagesFolderProcessId);

    String command = "\""+blenderPath+"\" -b -P \""+scriptPathpy+"\" -- "+ imagesFolderProcessId;
    executeCommand(command);

    String resultPath = scriptsPathimgs+File.separator+"_Results_"+File.separator+request.getProcessId()+"_HEAD_RECONSTRUCTION"+File.separator;
    String resultPathGLB = request.getProcessId()+"_head.glb"; //path of GLB file
    Resource resourceMesh = getFileAsResourcePath(resultPath, resultPathGLB); //resourceMesh contain the glb file
    if(resourceMesh == null) {
        logger.info(" > GLB not found!");
        return "";
    }
    logger.info(" > GLB found");

    String url = "https://108.142.221.68/imgs/_Results_/"+request.getProcessId()+"_HEAD_RECONSTRUCTION/"+request.getProcessId()+"_head.glb";
    return url;
}
```

Figure 3.15: FaceG3n: *executeBlenderAPI*

```java
public void downloadAndSaveImages(ImageDataRequest imageData, String localDirectory) {
    //looking for the folder if it exists already
    if(directoryExists(localDirectory)) {
        //delete old elements
        Path directoryPath = Path.of(localDirectory);
        try {
            eraseDirectory(directoryPath);
            logger.info("deleted old elements");
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    //saving images
    for (ImageFrame frame : imageData.getFrames()) {
        String imageUrl = frame.getUrl();
        try {
            URL url = new URL(imageUrl);
            try (InputStream in = url.openStream()) {
                // Create the directory if it doesn't exist
                Path directoryPath = Path.of(localDirectory);
                Files.createDirectories(directoryPath);

                String fileName = UUID.randomUUID().toString() + ".jpg";
                Path localPath = directoryPath.resolve(fileName);
                Files.copy(in, localPath, StandardCopyOption.REPLACE_EXISTING);
                logger.info("Downloaded and saved: {}", localPath);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Figure 3.16: FaceG3n: *downloadAndSaveImages*

Script Execution

Following the preparation of the images, within `executeBlenderAPI`, the `executeCommand` function is called, where the script "FB_Head_Reconstruction_Automatized.py" is invoked using ProcessBuilder.

The service is configured to interrupt the execution thread after a customizable time, which can be set through the "application.properties" file. This configuration is in place because, given that the service is hosted on a server without a GPU, if more than five images are sent, Blender is unable to generate the texture, causing the execution to hang.

During the script execution, all the informations about the operation is logged on the server.

```java
public Boolean executeCommand(String command) {
    Boolean result = false;

    ProcessBuilder processBuilder = new ProcessBuilder("/bin/bash", "-c", command);
    processBuilder.redirectErrorStream(true); // Redirect error stream to output stream
    try {
        Process process = processBuilder.start();
        processBuilder.redirectErrorStream(true);
        Thread timerThread = new Thread(() -> {
            try {
                TimeUnit.MINUTES.sleep(Long.parseLong(timeoutCommand));
                process.destroy();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        });
        timerThread.start();

        // Read the command output
        InputStream inputStream = process.getInputStream();
        InputStreamReader inputStreamReader = new InputStreamReader(inputStream);
        BufferedReader bufferedReader = new BufferedReader(inputStreamReader);

        // Close input stream and buffered reader
        inputStream.close();
        bufferedReader.close();
```

Figure 3.17: FaceG3n: *executeCommand*

Saving and Returning Results

Once the execution of the script is complete, FaceG3n saves the 3D head model in a specific directory. If the execution has occurred without errors, the service returns a URL from which the output of the 3D head creation can be downloaded.
To make the hosted file downloadable, Apache, a popular web server, has been configured on the server. Apache handles serving files to clients who request content.

The URL is associated with the process ID and is returned to the requester.

```java
public void sendData(ImageDataRequest request) {
    try {
        String urlAux = executeBlenderAPI(request);
        HttpHeaders headers = new HttpHeaders();
        headers.add("token", token);
        // Adding CORS headers
        headers.add("Access-Control-Allow-Origin", "*");
        headers.add("Access-Control-Allow-Methods", "GET, POST, PUT, DELETE, OPTIONS");
        headers.add("Access-Control-Allow-Headers", "Content-Type, Authorization, Accept, X-Requested-With, X-CSRF-Token, token");

        SendDataResponse data = new SendDataResponse();
        data.setProcess_id(request.getProcessId());
        data.setUrl(urlAux);

        Mono<String> responseMono = webClientService.postData(pluxboxendpoint, headers, data, String.class);
        responseMono
        .timeout(Duration.ofSeconds(Long.parseLong(timeout)))
        .onErrorResume(
            TimeoutException.class,
            error -> {
                return Mono.just("Timeout error occurred");
            }
        )
        .doOnSubscribe(subscription -> logger.info("Subscribed to resMono"))
        .doOnNext(result -> {
            logger.info("Response received: {}", result);
        })
        .subscribe(
            result -> {
                if (result != null) {
                    logger.info("result: {}", result);
                } else {
                    logger.info("result was empty");
                }
            },
            error -> {
                logger.error("Error during subscription", error);
            }
        );
    }catch(Exception e) {
        logger.error("error on sending post: {}",e);
    }
```

Figure 3.18: FaceG3n: *sendData*

Communication with the Orchestrator

With the URL ready, FaceG3n configures the necessary headers, including the authorization token and CORS settings. The data required for the asynchronous call, which is the process_id and the URL of the 3D mesh, are prepared within a JSON file and sent to the Orchestrator through the WebClient service, ensuring the asynchronous delivery of information to the Orchestrator.

```java
public <T, R> Mono<R> postData(String url,HttpHeaders headers, T requestBody, Class<R> responseType) {
    return webClient.post()
            .uri(url)
            .headers(httpHeaders -> httpHeaders.addAll(headers))
            .bodyValue(requestBody)
            .retrieve()
            .bodyToMono(responseType);
}
```

Figure 3.19: FaceG3n: *postData*

The JSON file will be of this type:

```
{
   "process_id": "64fb458334054fa8fe96c181",
   "url": "https://pluxbox.b-cdn.net/callas.glb"
}
```

It will be sent in the body of the POST request to the following endpoint:
`https://demo.pluxbox.studio/integration/synthetic_humans_synthetic_humans/model`.

3.3.4 Deploy and Hosting

FaceG3n has been made available on a RAI virtual machine based on Linux (Ubuntu 22.04.2 LTS) hosted on Azure.

Despite not having a dedicated GPU, this machine has been configured with Blender and the FaceBuilder plugin to enable script execution. Access to this machine is also possible via Remote Desktop (RDP), as the installation of the FaceBuilder plugin requires the use of Blender's graphical interface.

3.4 Transformation of the 3D Model into Metahuman

In our journey of creating Synthetic Humans, we have delved into the 3D Face Reconstruction process in detail, a crucial step in our workflow, with a focus on maximizing automation in this phase.

We utilized RAI's archive and its "Smart Gallery" to gather the necessary visual materials, such as images and videos, which are essential for the reconstruction of 3D face models.

Subsequently, the Orchestrator was introduced as a key element in the workflow, elucidating its role in automating the processes of Frame Extraction, Image Restoration, and 3D Face Reconstruction.
We have taken a close look at the 3D Face Reconstruction phase, with particular attention to tools like FaceBuilder and the "FB_Head_Reconstruction_Automatized.py" script, which enabled complete automation in the creation of the *head mesh*.
Furthermore, we have illustrated how the FaceG3n service enables asynchronous interaction between the Orchestrator and the 3D head creation script.

With this foundation, we are now ready to explore the next step in our workflow, focusing on the transformation of the 3D models obtained from the 3D Face Reconstruction phase into Metahuman using Unreal Engine.

Metahuman [41] is the most advanced software for creating Synthetic Humans.
It allows for the creation of avatars of exceptional quality and realism significantly faster and more efficiently than traditional methods.

MetaHuman Creator [34], the main component of Metahuman developed by Epic Games, is a revolutionary platform in the world of photorealistic avatar creation. Using a cloud-based application, you can generate MetaHumans in just minutes, starting from

one of the available character presets or converting a custom mesh with the Metahuman plugin for Unreal Engine.

In this chapter, we will delve into the details of each stage involved in the process of creating a Metahuman, starting from a head mesh. This analysis will allow us to fully understand how Unreal Engine and the Metahuman plugin work synergistically to achieve exceptionally realistic and high-quality results.

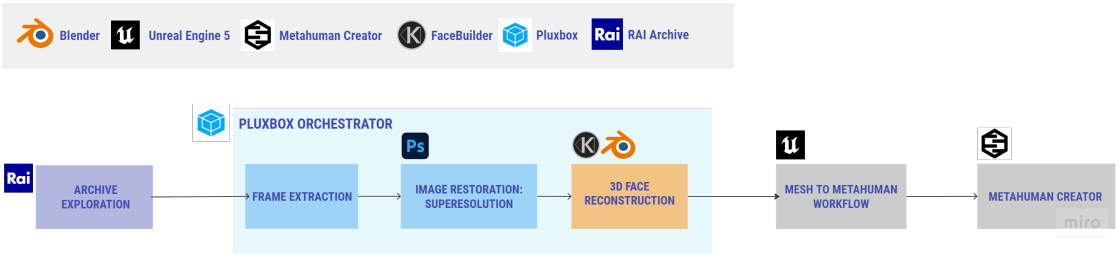


Figure 3.20

3.4.1 Plugin for Unreal: MetaHuman

The **MetaHuman Plugin** [42], is an incredibly versatile resource, consisting of two powerful sets of features, each designed to significantly enrich the content creation process through the use of Unreal Engine.

- **"Mesh to MetaHuman"**: This represents one of the most fascinating aspects of this tool. With it, you can achieve the remarkable ability to transform a custom mesh, created through scanning, traditional modeling, or sculpting, into a complete MetaHuman with a skeleton. Additionally, once this phase is completed, it opens up further opportunities for further character customization using MetaHuman Creator.

- **MetaHuman Animator**: This is one of the new features of this plugin, giving you the ability to capture an actor's performances using devices like iPhone or stereo HMC systems and translate them into highly faithful facial animations for MetaHumans. Every subtle expression, every glance, and every emotion are captured with incredible precision and delivered in a highly realistic manner, making astonishing results accessible to everyone. This tool uses an advanced 4D solver that combines video data with depth information and the representation of the actor's MetaHuman. The produced animation is processed locally using GPU hardware, making the final animation available in just a few minutes.
  However, it's important to note that to fully harness the capabilities of MetaHuman Animator, Unreal Engine 5.2 or later versions are essential.

In summary, the MetaHuman Plugin enables the production of high-quality results in an efficient and accessible manner.

3.4.2 Mesh To MetaHuman

Let's analyze the process of creating a Metahuman, starting from an existing mesh, whether it's static or skeletal, using the *"Mesh to Metahuman"* workflow.

Preliminary Requirements

To fully utilize the *"Mesh To MetaHuman"* workflow, it's essential to become familiar with key concepts related to Unreal Engine and Quixel Bridge, a tool that facilitates the seamless transfer of assets between Megascans and Unreal Engine by providing an interface for browsing, managing, and downloading Quixel Megascans assets directly into the engine.

In particular, it's important to understand the difference between Static Meshes and Skeletal Meshes in Unreal Engine. Static Meshes represent static objects with no intrinsic animations, while Skeletal Meshes are used for animated characters and objects, featuring a skeletal structure.
Additionally, it's crucial to register and log in to Quixel Bridge, even if access via the Epic Games Launcher has been previously done.

Mandatory Configuration

Before starting the workflow, there are some mandatory configurations to be done:

1. **Create a New Unreal Engine Project:** Create a new project in Unreal Engine to load the mesh from which you want to create the Metahuman.

2. **Install and Activate the MetaHumans Plugin:** To leverage the "Mesh to Metahuman" functionality, it's crucial to download, install, and activate the MetaHumans plugin in your project. After installing the plugin, you may need to restart the Unreal Engine project to ensure it's correctly installed.

3.4.3 Workflow *Mesh To MetaHuman*

The "Mesh to MetaHuman" workflow is divided into a series of key steps [43], which will be analyzed in detail.

Importing and Preparing the Character Mesh

The first phase of the process involves importing the character mesh, whether it's static or skeletal, in .fbx format into the Unreal Engine project. This will result in a textured mesh ready for the subsequent steps in the "Mesh to MetaHuman" pipeline.

Creating and Populating a MetaHuman Identity Asset

The next step is the creation of a MetaHuman Identity Asset. This asset plays a fundamental role as it collects crucial character details, including data related to the face

mesh, poses, and body information.
Initially, this asset will be empty and will require populating it with relevant data.

In the Main Toolbar of the Identity Asset Editor, as you can see in figure 3.21, you can locate and select the imported head mesh from the first step. This process will generate a Components panel on the left side of the MetaHuman Identity Asset Editor, comprising the following sections: Face, Poses, and Body.

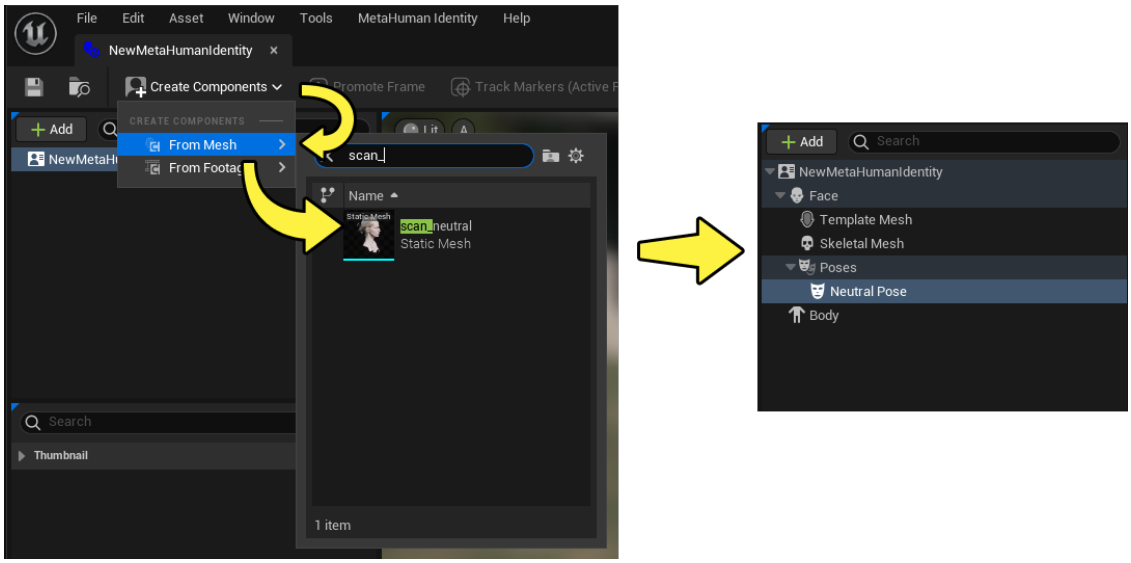


Figure 3.21

Selecting the Body Component and accessing the panel, in figure 3.21, allows you to specify the height and body type of your MetaHuman. These values can be later adjusted if necessary in MetaHuman Creator.

Creating and Capturing a Neutral Pose

Before the MetaHuman backend can create a Template Mesh compatible with the imported character, it's necessary to create and capture a Neutral Pose. In this context, a Neutral Pose represents the face mesh without any facial expression, ensuring compatibility with the MetaHuman.

To facilitate working on the mesh, you can adjust the lighting in the viewport.
You can switch between three different types of lighting: Lit, Unlit, and Lighting Only, but for textured meshes, as in our case, the most suitable mode is Unlit.

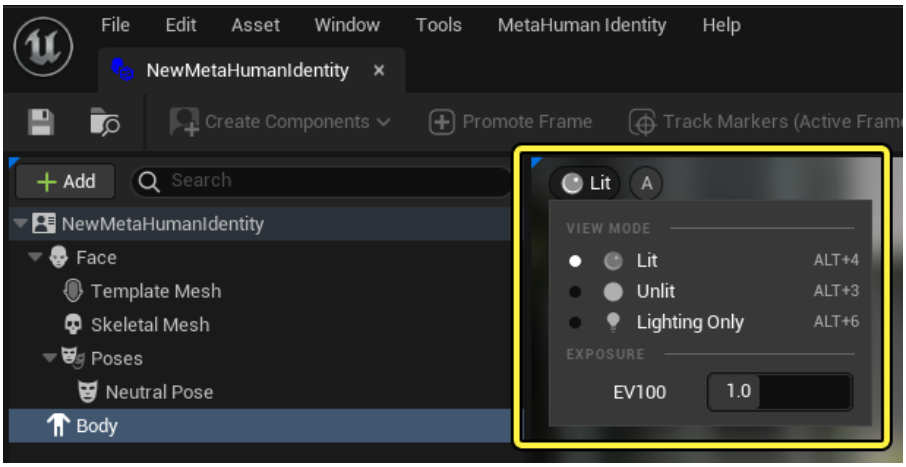


Figure 3.22

To create a Neutral Pose, select the "Neutral Pose" component in the Poses section of the components panel. Next, position the camera so that the mesh is facing directly forward and completely visible. Then, reduce the Field of View to less than 20 degrees. With the Neutral Pose selected, click on **"Promote Frame"** in the Main Toolbar.

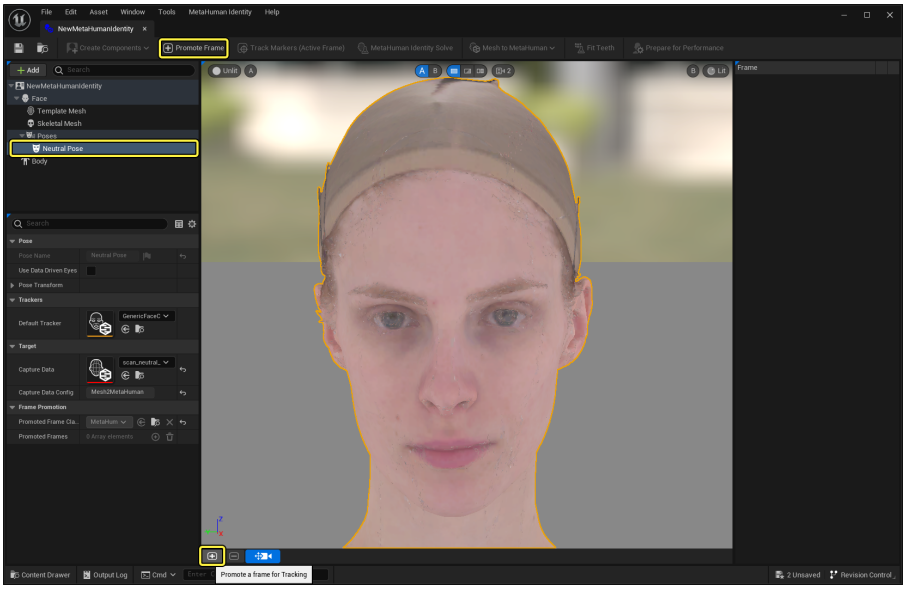


Figure 3.23

Selecting the frame created in the Neutral Pose Editor and activating the **"Track Markers (Active Frame)"** button in the Main Toolbar will automatically generate a series of markers along the facial features of the mesh.

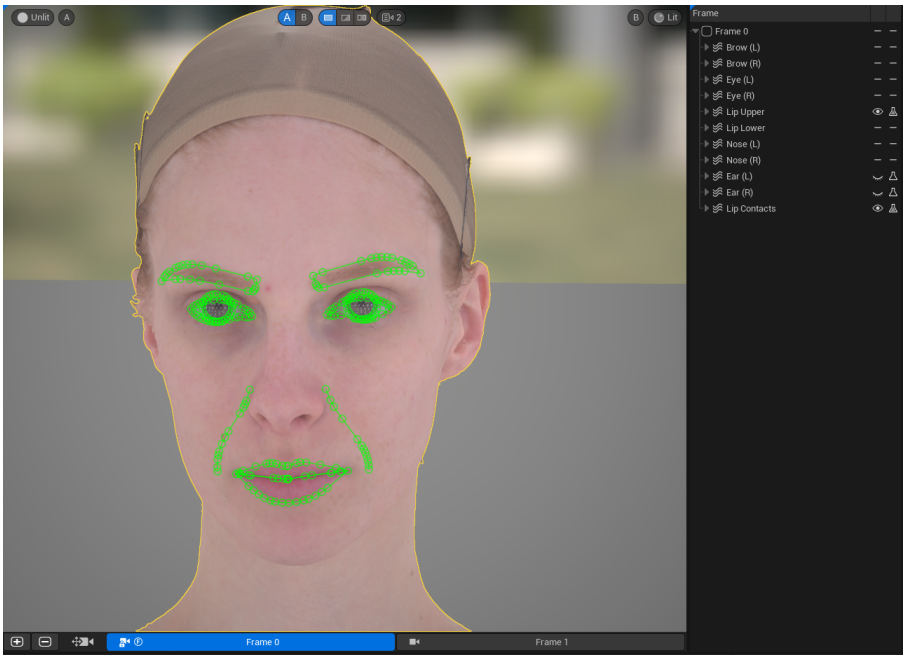


Figure 3.24

Running the Identity Solve

In the MetaHuman Identity Asset Editor, in the Main Toolbar, click on the **"MetaHuman Identity Solve"** button. This option is only enabled if there is at least one Neutral Pose in the timeline and produces optimal results when the markers are active and well-tracked. The Identity Solve adapts the vertices of the Template Mesh to the volume of the Neutral Pose mesh captured in the previous steps. This processing takes place in the cloud.

After the completion of the Identity Solve, you can switch between the original mesh and the Template Mesh in the viewport using the A/B tabs in the viewport Toolbar. Additionally, you can configure viewport settings, such as lighting, and enable or disable the display of both the original mesh and the Template Mesh to check if the two meshes "overlap" on the screen.

Figure 3.25

Sending the Template Mesh for Auto-Rigging

Now, you can send the Template Mesh to the MetaHuman backend for auto-rigging. In the Main Toolbar, click on the "Mesh to MetaHuman" button and choose one of the following options:

- **Auto-Rig Identity (Skeletal Mesh Only):** This option creates a Skeletal Mesh with MetaHuman DNA included. The mesh is automatically downloaded and added to the Unreal Engine project in the same location as the MetaHuman Identity Asset, with the same name as the Asset Identity, prefixed with "SK_".

- **Auto-Rig Identity (Skeletal Mesh + Full MetaHuman):** This option creates an auto-rigged Skeletal Mesh with MetaHuman DNA included and generates a customized MetaHuman that can be further edited in MetaHuman Creator and downloaded via Bridge.

In our case, we should select the **"Auto-Rig Identity (Skeletal Mesh + Full MetaHuman)"** option, and once the process is complete, a confirmation message will be displayed.

Customizing the MetaHuman in MetaHuman Creator

After completing the previous step, you can access the MetaHuman in MetaHuman Creator. In the case of MetaHumans created through the "Mesh to MetaHuman" workflow, you will see an additional panel called "Custom Mesh." This panel allows you to adjust the contribution of the difference between the Template Mesh and a standard MetaHuman parametrization, enabling greater customization.

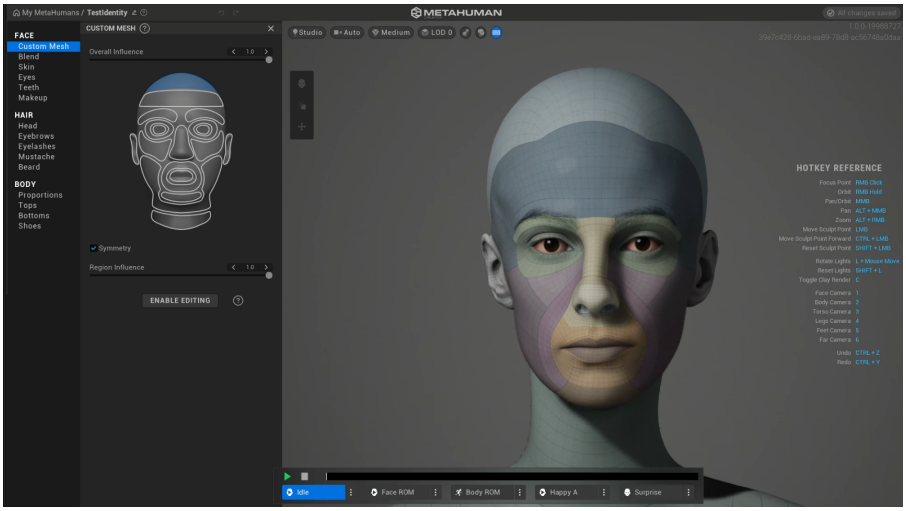


Figure 3.26

It's important to note that the "Mesh to MetaHuman" process doesn't automatically adjust the skin tone of the MetaHuman to match the skin color of the mesh. To set a skin color for the MetaHuman, which initially has the same texture as the Template Mesh in Unreal Engine, you can use skin controls.

Download and Import of the MetaHuman into Unreal Engine 5

The new MetaHuman is now available for download through Quixel Bridge. In Unreal Engine 5, you can open Bridge from the Main Toolbar.
Make sure to log in to Bridge using your Epic Games account if you haven't already done so. Once logged in, you can access the new MetaHuman in the "My MetaHumans" section of Bridge.

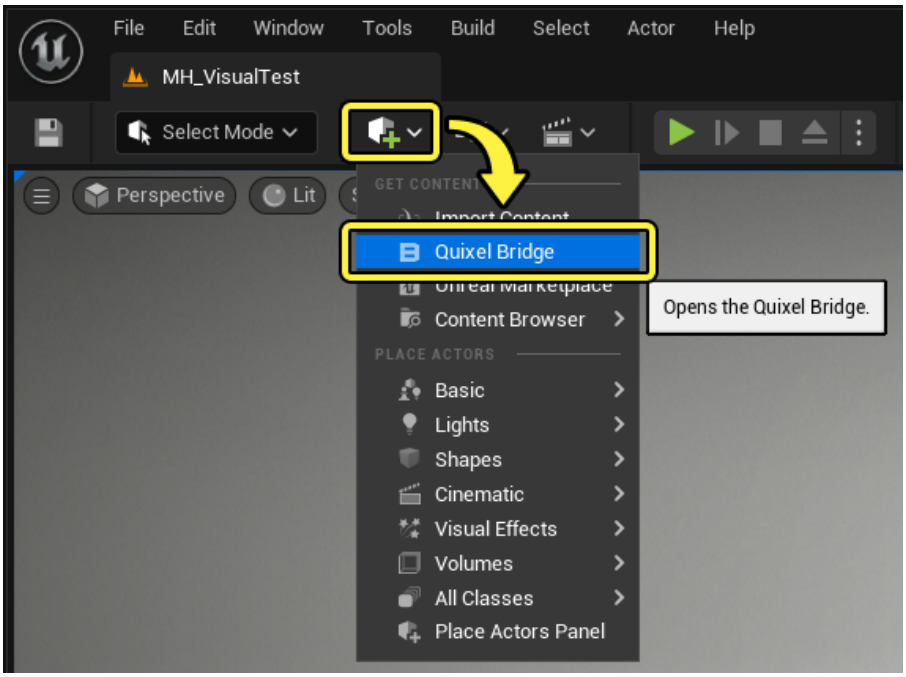


Figure 3.27

3.5 Facial Animation for MetaHumans

In this chapter we will delve into another step in our workflow for creating synthetic humans. So far we have looked in detail at the 3D face reconstruction process and highlighted the key role played by the Orchestrator in coordinating the various stages, as well as how to transform 3D face models into Metahumans using Unreal Engine. Now the focus shifts to a new semi-automatic step, facial animation.

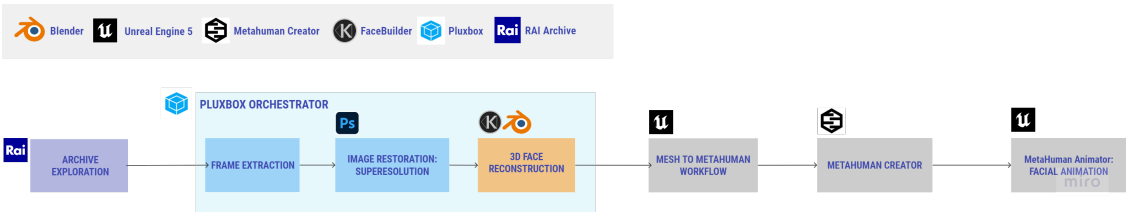


Figure 3.28

Facial animation plays a crucial role in bringing digital characters to life and infusing them with realism. The challenge of capturing realistic and high-quality facial expressions has always been a demanding task, requiring specialized skills and significant resources. However, the evolution of digital technologies has made it possible to simplify and make facial animation accessible. A cutting-edge tool in this field is the **MetaHuman Animator** plugin by Epic Games [44].

MetaHuman Animator represents the highly anticipated new set of tools dedicated to facial animation and performance capture for the MetaHuman framework developed by Epic Games.
This system was designed with the goal of simplifying the process of transferring an actor's facial performances from videos captured using devices like the iPhone or helmet-mounted cameras directly within **Unreal Engine**. This simplification of the high-quality facial animation creation process represents a significant step in the animation industry.

Initially, MetaHuman characters already had facial rigs, meaning they were set up for facial motion capture.
However, the process of transferring performances from actors with different facial proportions required substantial manual processing. MetaHuman Animator was specifically developed to streamline this operation, introducing an automated workflow known as **"Footage to MetaHuman"**.

3.5.1 Hardware Requirements and Compatibility

MetaHuman Animator is designed to work with a wide range of facial capture systems, from iPhones with Epic Games' free Live Link Face app to professional helmet-mounted cameras, including those from ILM Technoprops [44].

MetaHuman Animator is included in Epic Games' free MetaHuman plugin. The plugin is compatible with Unreal Engine 5.0+ and requires Unreal Engine 5.2+ for using MetaHuman Animator.
The Live Link Face app is available for free on iOS devices and requires plugin version 1.3+. To use it with MetaHuman Animator, you'll need an iPhone 12 or later.

3.5.2 Key Features of MetaHuman Animator

One of the standout features of **MetaHuman Animator** is its ability to efficiently generate high-quality facial animations. In the past, faithfully recreating every nuance of an actor's performance on a digital character required months of work by a team of experts. Now, MetaHuman Animator accomplishes this feat in a fraction of the time and with significantly less effort.

At the core of this remarkable capability is a sophisticated 4D solver that combines video and depth data with a representation of the actor within MetaHuman.
The animation is processed locally using GPU hardware, and the final result is available within minutes.

The process primarily occurs in the background, greatly simplifying the user experience. The procedure is reduced to a few steps: simply point the camera at the actor and start recording. MetaHuman Animator accurately reproduces the individuality and nuances of the actor's performance on any MetaHuman character.
An essential aspect is that the generated animation data is "semantically correct". This means it follows the rig controls' structure similar to what a human animator would do.

Facial Animations for Any MetaHuman

A remarkable aspect of MetaHuman Animator is its flexibility and universal compatibility. Facial animations captured with this tool can be applied to any MetaHuman character with just a few steps.

Technically, this is made possible through ***"Mesh to MetaHuman"***, a process that creates a MetaHuman Identity that can be use for the animation from only three frames of video, along with depth data captured using an iPhone or reconstructed using stereo helmet-mounted camera data.
This process customizes the solution for the actor, allowing MetaHuman Animator to produce animations that work on any MetaHuman character. Additionally, the system can use audio recorded during facial capture to generate realistic lip and tongue animations, further enhancing the believability of digital character performances.

3.5.3 Metahuman Animator Workflow

In order to take advantage of the capabilities of MetaHuman Animator in Unreal Engine, it is essential to follow a well-defined workflow. Listed below are the key steps to follow to create facial animations:

1. **Preparation:** Before you embark on the journey of creating facial animations with MetaHuman Animator, it's essential to ensure that you have all the prerequisites in place, that are:
   Use a powerful Windows PC that meets Epic's hardware requirements. Note that MacOS and Linux are not supported. Register for an Epic Account.
   Download the MetaHuman Plugin from the Marketplace and install it in Unreal Engine 5.2.
   Install the Live Link Face App on your mobile device, preferably an iPhone 12 or later.

   This are the first steps for the creation of facial animations.

2. **MetaHuman Plugin Setup:** With all preparation down, open a blank project. First thing, is to enable the MetaHuman Plugin inside your UE project.

   Click ***Tools -> Plugins ->*** *Search and Enable* ***MetaHuman Plugins*** *-> Restart the project.*

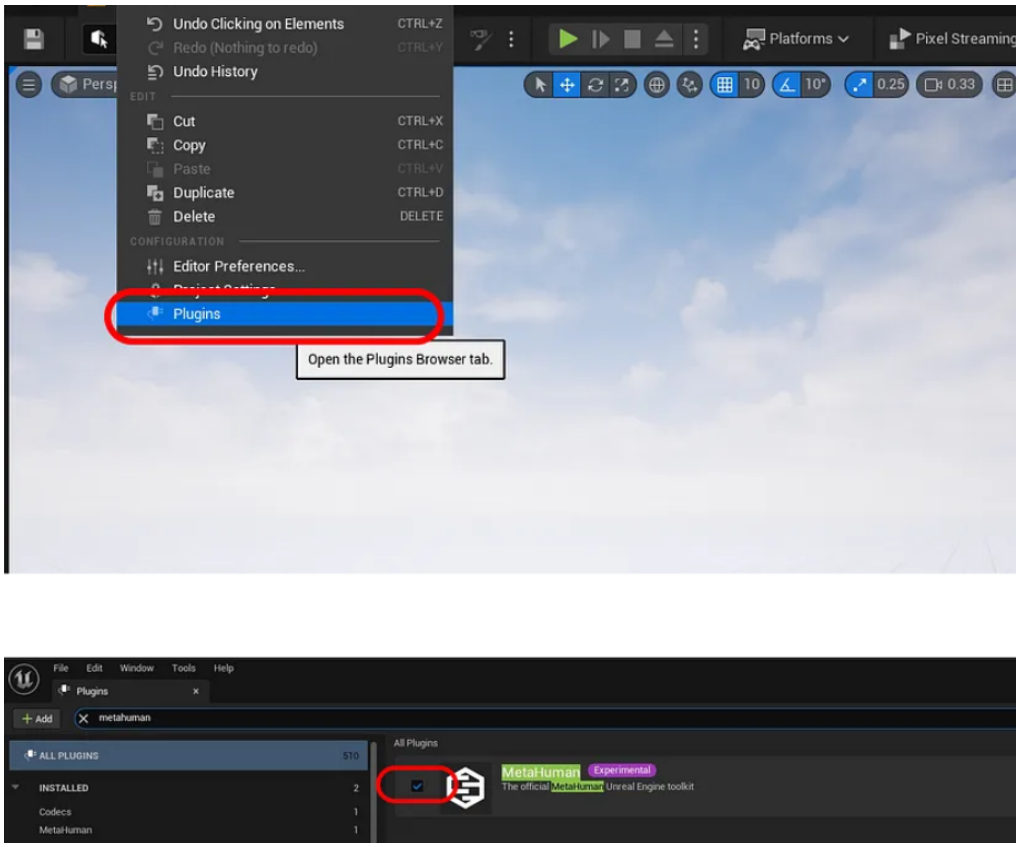


Figure 3.29

3. **Footage Capture with Live Link Face App:** Proceed to use the Live Link Face App for capturing the required footage:

   Launch the Live Link Face App and select *"MetaHuman Animator" mode.*
   Record two essential footages:
   a) Footage One: Capture the primary facial data for creating the MetaHuman. Ensure this video includes frontal, left, and right views of the face.
   b) Footage Two: Record facial expressions that the generated MetaHuman will mimic.

   In the ***Take Browser***, export the recorded video and download the file to your computer for later use.

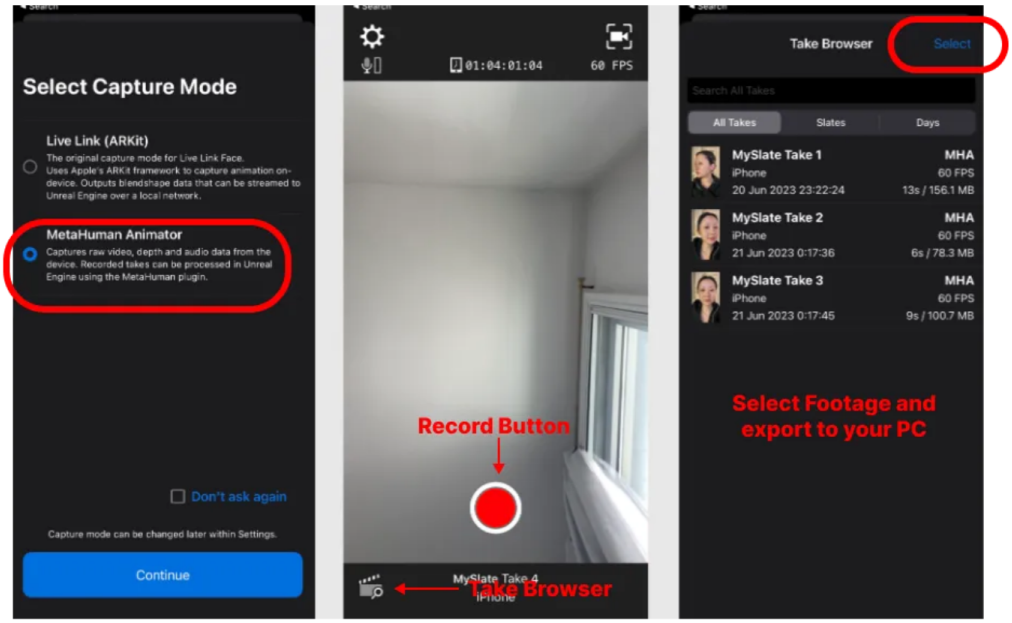
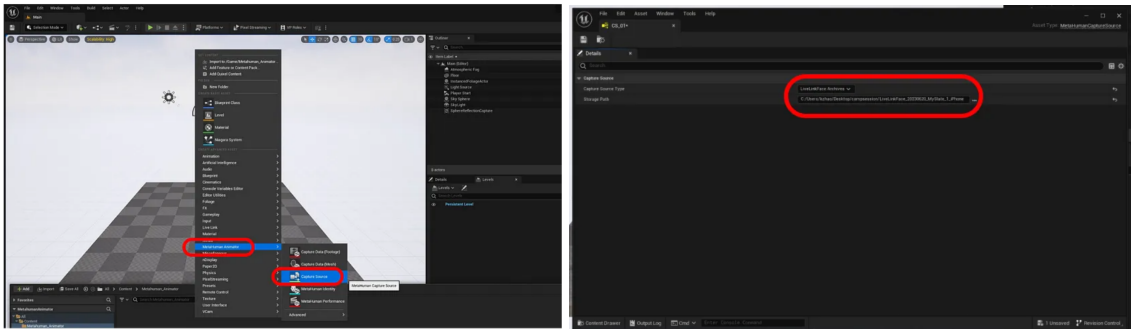


Figure 3.30: The exported files are zipped, be sure to unzip it on your computer

4. **MetaHuman Animator**
   a) Import Capture Footage to UE Project

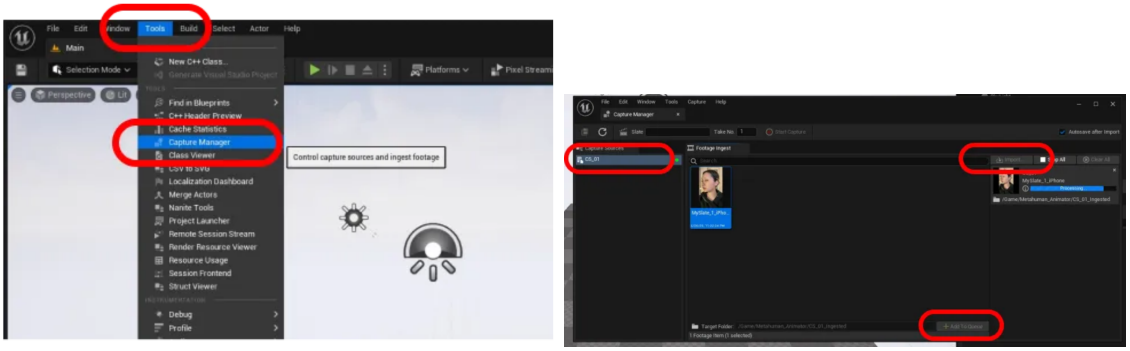
   In the Content Browser, right-click and navigate to ***MetaHuman Animator -> Capture Source -> LiveLink Archives***. Select your video file path.



(a) (b)

Figure 3.31: Import Capture Footage to UE Project

In the top left corner, go to *Tools -> **Capture Manager**. Choose the source you just added and click **Add to Queue**.* Then, click ***Import***.

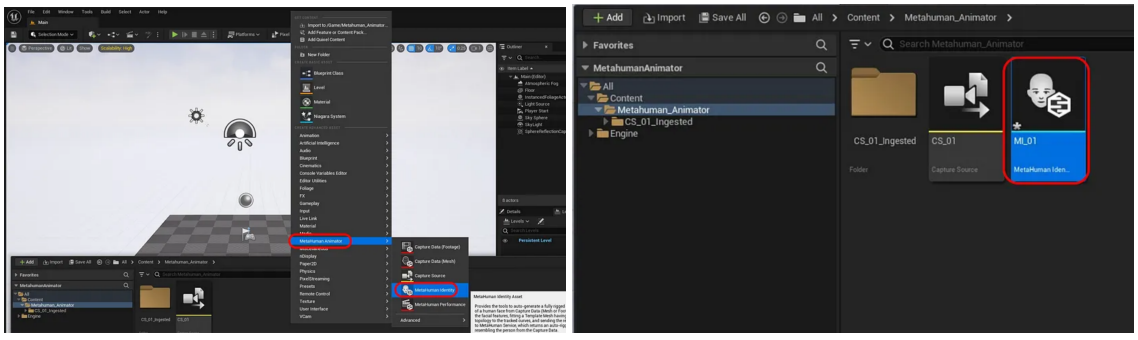


(a) (b)

Figure 3.32: a) go to Tools -> Capture Manager b) Select footage on the left panel -> Add to Queue-> Import

By now, you should have a new folder appear in your content browser.

b) Generate a MetaHuman from Video Source
In the Content Browser, right-click and select ***MetaHuman Animator -> MetaHuman Identity.***



(a) (b)

Figure 3.33: Generate a MetaHuman from Video Source

A login page will appear. After *logging in*, click ***Create Components -> From Footage***. Choose your previously recorded footage.

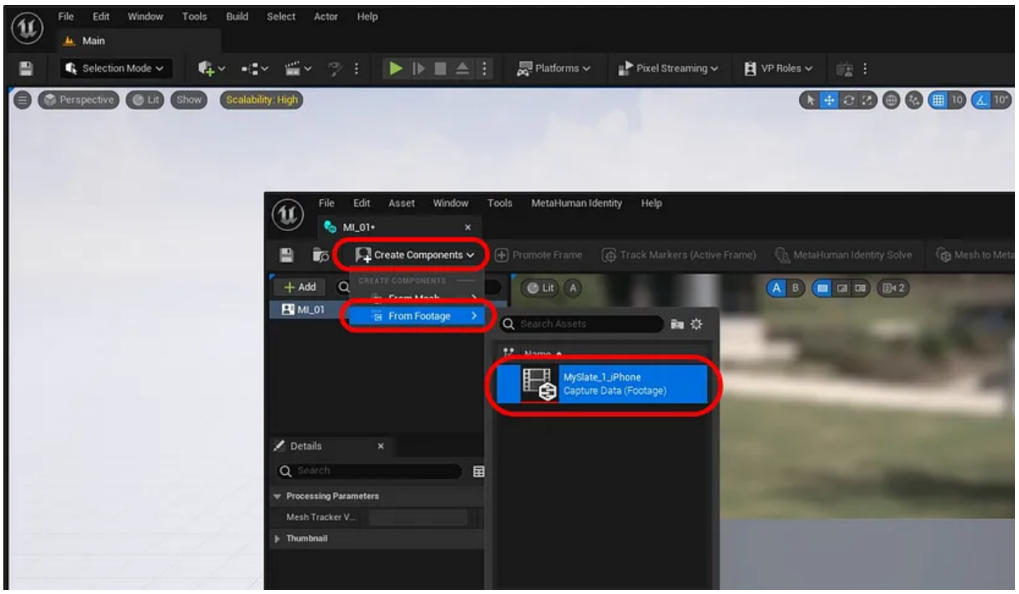


Figure 3.34: Create Components -> From Footage

Scroll the timeline to a frame with a clear frontal face, click the plus sign to *set that frame as a keyframe*. Repeat this for two more frames: one for the left side and one for the right side of the face.

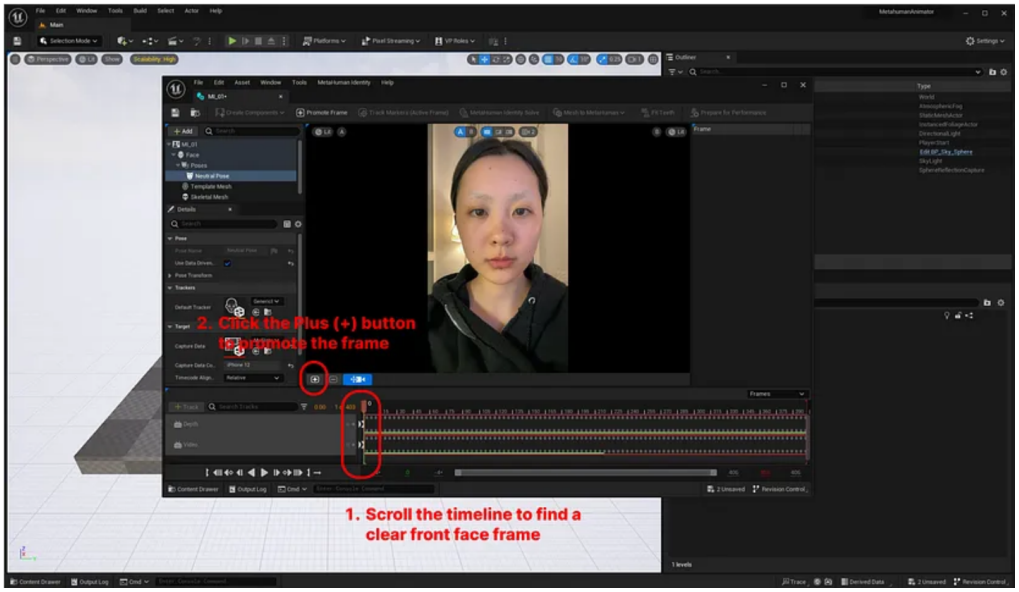
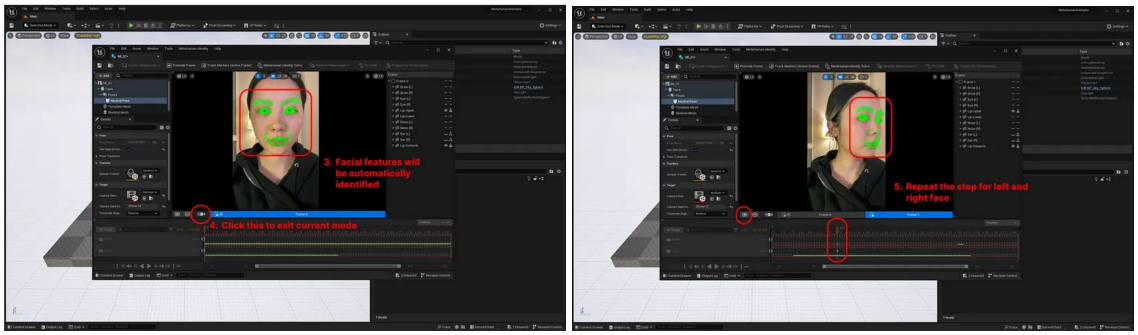


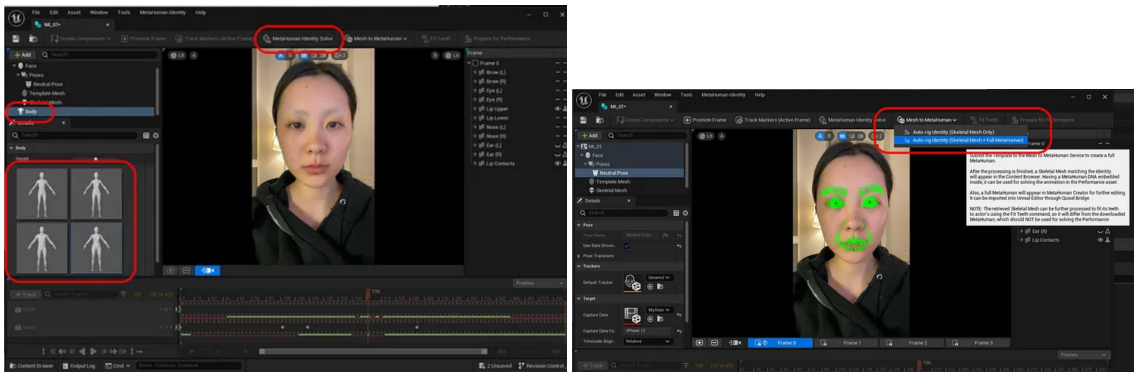
Figure 3.35: Scroll the timeline to a frame with a clear frontal face and set that frame as a keyframe.



(a) (b)

Figure 3.36: Set other frames as a keyframe: one (a) for the left side and one (b) for the right side of the face.

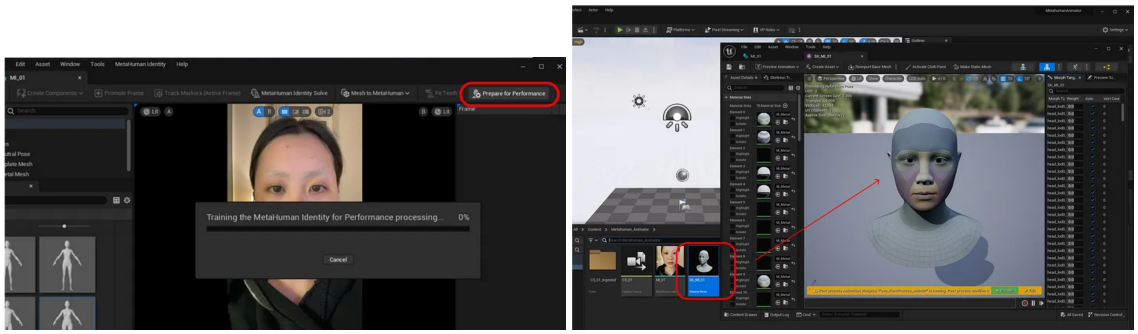
In total, you should have ***three keyframes***. Click on ***Body*** in the left panel and select a body type. Then, click ***MetaHuman Identity Solve*** and after that proceed to ***Mesh to MetaHuman***. After completing this step, your MetaHuman should be available in *MetaHuman Creator*.



(a) (b)

Figure 3.37: Creating of the MetaHuman Identity

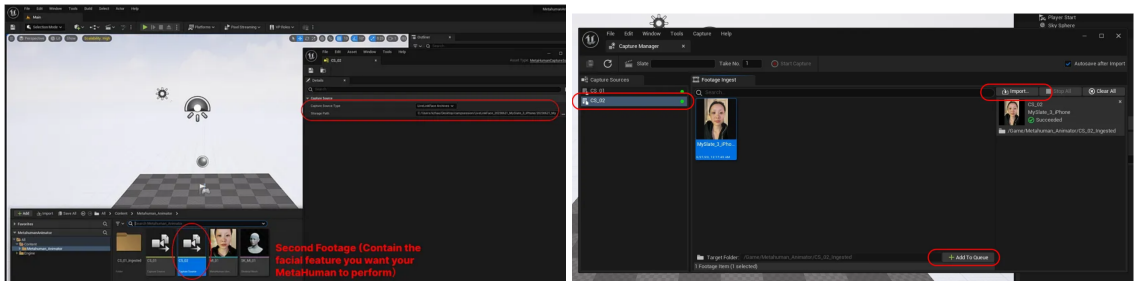
Finally, run Prepare For Performance. This step may take some time to complete. It is recommended to use this time for customizing your MetaHuman in MetaHuman Creator.
Afterward, a new Skeletal Mesh will appear in folder.



(a) (b)

Figure 3.38

c) Import Face Animation Footage to UE
Now, the second footage captured for the generated MetaHuman is imported to perform. The same steps as before should be followed:
Right-click on ***MetaHuman Animator -> Capture Source.***
Go to ***Capture Manager***, select the source, and click ***Add to Queue***, then click ***Import***.



(a) (b)

Figure 3.39: Import the second Face Animation Footage to UE

In Content Browser, *right-click -> **MetaHuman Animator -> MetaHuman Performance***.
Choose the second imported footage and the MetaHuman Identity as in figure 3.40. Go to Export Animation and select the target skeleton to be your MetaHuman as in fugure 3.41 and 3.42.

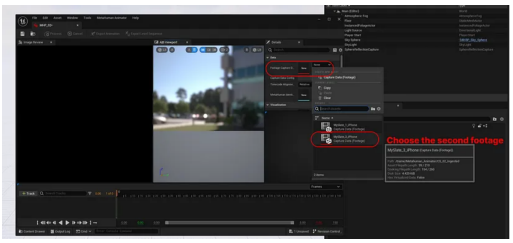


Figure 3.40: choose footage

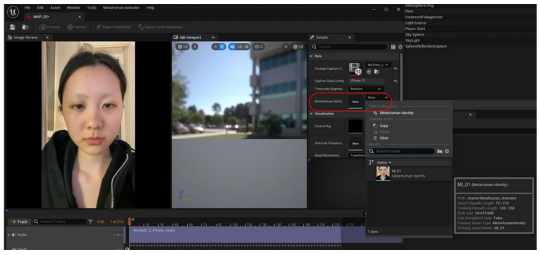


Figure 3.41: select the target skeleton

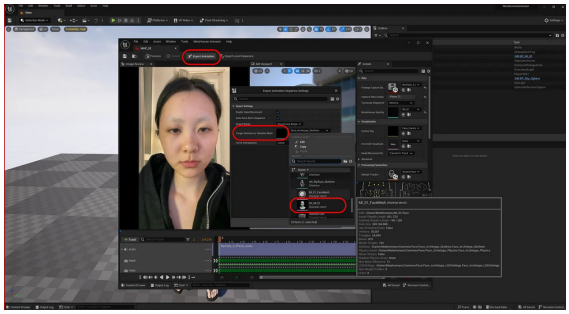


Figure 3.42: Export Animation

d) Import MetaHuman to Project

With your facial animation ready, the final step is to import the MetaHuman from the cloud to your project and perform the animation:
Click ***Add -> Add Quixel Content***, then *log into* your account and in *"In My MetaHuman"* section, select the MetaHuman you just created or anybody else.
Choose a quality level, download, and add it to your project.
After a successful import, you should find a ***MetaHuman folder*** in your Content Browser and drag your MetaHuman blueprint to the level.

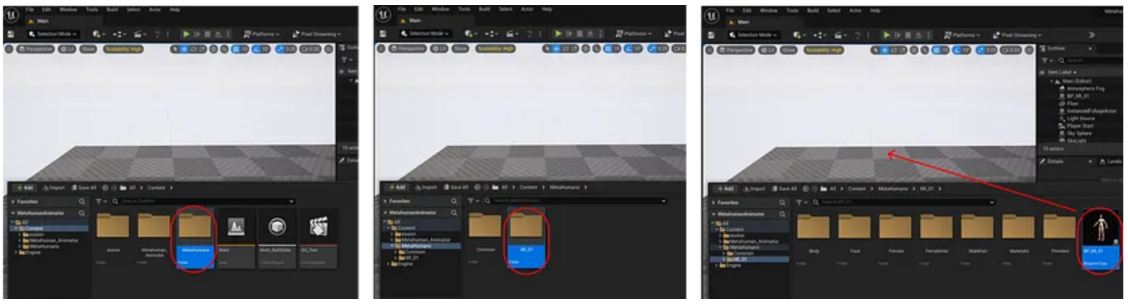


Figure 3.43: Import MetaHuman to Project

After that, create a ***new Level Sequencer***, and in the sequencer, drag the MetaHuman BP. Add the previously animation exported and delete the face control rig.

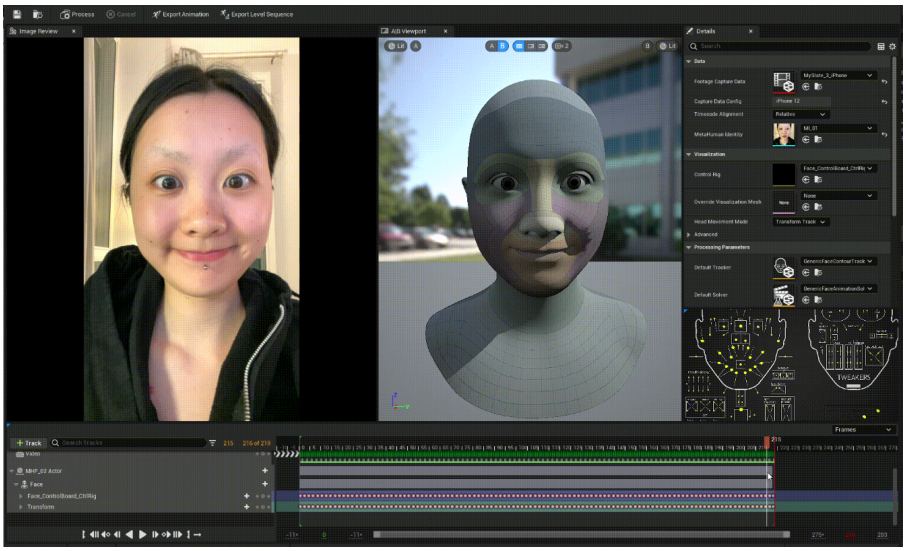


Figure 3.44: End Result: Metahuman Performance

Chapter 4

Generating Textures for Synthetic Humans

Creating realistic 3D models of human faces requires special attention to texture because it plays a crucial role in the overall appearance of a Synthetic Human.
The texture of a 3D human face is critical for making a Synthetic Human look realistic and believable. It is responsible for reproducing subtle facial details, including skin pores, moles, scars, and other distinctive elements that contribute to an individual's uniqueness. High-quality texture is essential to capture these details and ensure that the Synthetic Human closely resembles the real individual they are intended to represent.

4.1 Challenges in Texture Creation with FaceBuilder

During the Face Reconstruction phase, automated through the use of the ***"FB_Head_Reconstruction_Automatized.py"*** script for facial texture creation, we rely on the texture extraction algorithm provided by FaceBuilder. This algorithm integrates camera views, aligning the 3D facial mesh with previously uploaded photos, and generates the texture by projecting each pixel of the 3D mesh's UV map onto the photo according to the model's position. This process is repeated for each uploaded photo, and therefore, each camera aligned with the 3D model [22] [23].

The use of the FaceBuilder plugin significantly streamlines the texture creation process, especially when automated through the script. However, there are issues related to the resulting textures that render those obtained via the FaceBuilder algorithm unsuitable for direct use on Synthetic Humans. These issues can be attributed to two main factors:

- **Quality of Reference Images:** Creating a high-quality texture with FaceBuilder necessitates exceptionally high-quality reference images. These images should be sharp, high-resolution, and captured under consistent lighting conditions.
  Blurry, low-resolution, or differently lit images can lead to the generation of textures with artifacts and color inconsistencies, resulting in an unrealistic and less believable appearance.

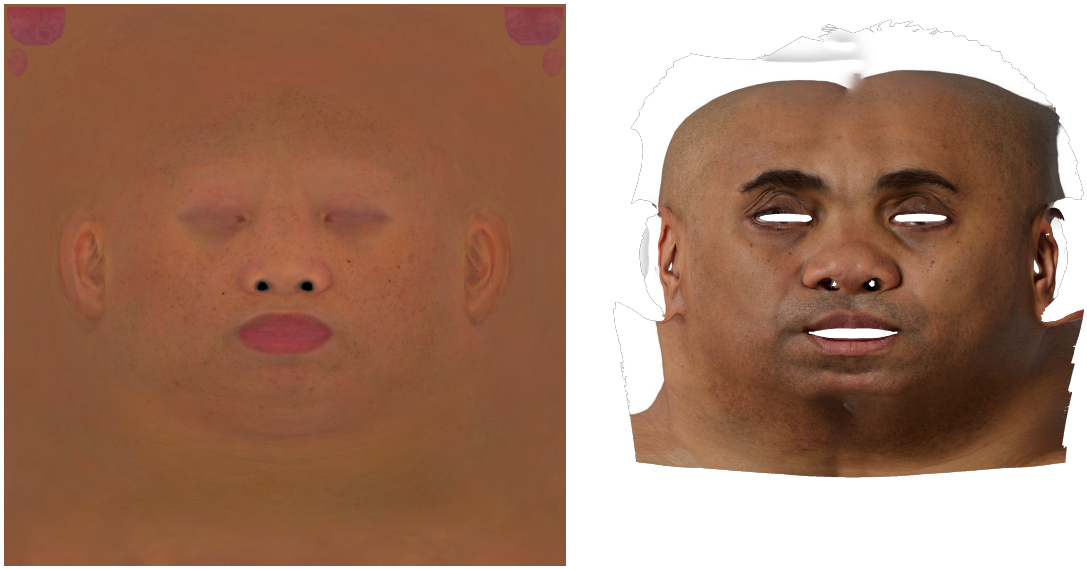
- **Coverage of Various Facial Angles:** Another critical factor involves ensuring that the images used for the creation of the 3D mesh cover various angles of the individual's face. The FaceBuilder texture generation algorithm relies on these images to align virtual cameras with the 3D model. Images that do not adequately cover all facial angles can lead to imperfect alignments and, as a result, distorted and irregular textures.

Assuming that these problems do not arise, i.e., having high-quality images covering the face from various angles, enabling a complete 360° reconstruction of the individual, the result would be an exceptional texture capable of capturing all the specific details of the individual's face.

4.2 Textures Generated by Metahuman Creator

Given the potential issues associated with using textures generated by the FaceBuilder algorithm, in some instances, it may be preferable to resort to textures generated by tools like Metahuman Creator. This is because textures generated by Metahuman Creator offer a realistic and uniform skin texture.

However, the problem arises when these textures appear "generic" in the sense that they do not incorporate specific facial details of the subject being represented, such as moles, scars, and other distinctive features. Neglecting these details could prevent the Synthetic Human from fully capturing the uniqueness of the individual's face.



(a) Texture Metahuman (b) Texture FaceBuilder

Figure 4.1: Example Texture

4.3 Hybrid Approach: Combining Texture from FaceBuilder and Metahuman Creator

One of the most promising solutions for addressing texture challenges in Synthetic Humans is the hybrid approach. This approach involves combining the texture generated by FaceBuilder with that obtained from Metahuman Creator. This strategy allows you to leverage the best of both worlds, capturing both the specific facial details (from FaceBuilder) and the overall realism of the skin (from Metahuman).

To effectively create these hybrid solutions, you can rely on advanced graphics tools like

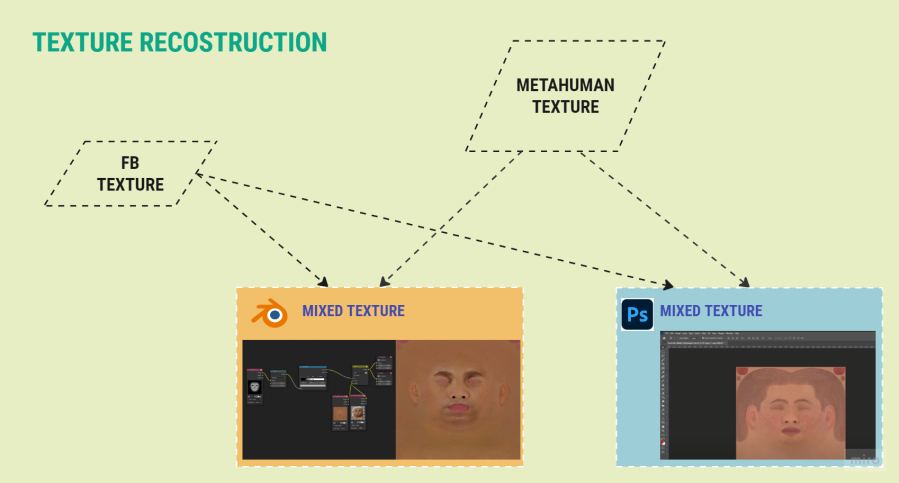


Figure 4.2: Combining Texture from FaceBuilder and Metahuman Creator

Adobe Photoshop, which allows you to overlay the textures obtained from FaceBuilder and those generated by Metahuman Creator by following a series of manual steps. Alternatively, it as been provided an in-house Blender project that enables the semi-automatic combination of the two textures.

Photoshop: Mixed Texture

In the process of merging the texture generated by FaceBuilder with the one from Metahuman Creator using Adobe Photoshop, you can follow these key steps:

1. **Project Settings:** Create a new project with dimensions of 2048x2048 pixels.
2. **Preparation of FaceBuilder (FB) Texture:** Load the FaceBuilder texture into a layer and rasterize the layer.
3. **Preparation of Metahuman (MH) Texture:** Create a different layer to place the Metahuman texture and also rasterize this layer.

4. **Adjustments to FB Texture:** Darken the layer containing the MH texture. Select the FB texture, go to "*Image" -> "Adjustment" -> "Shadow/Highlight"*, and increase the shadow to equalize the texture color.

5. **Creation of FB Texture Copies:** Create two copies of the FB texture and darken the original texture and one copy.

6. **Adjustments to the Copy of FB Texture:** Select one copy and apply "Filter" -> "Blur" -> "Gaussian Blur" at 16% to remove details and retain only the color information of the texture.

7. **Creation of a Detail Mask:** Activate the layer of the second copy of FaceBuilder. Position the second copy above the first, and select the second copy. Then go to *"Image" -> "Apply Image"* and *select "Subtract" for Blending Mode, "Scale" at 2, and "Offset" at 128.* This will create a mask containing only the details of the FB texture, without color information.
   Rename the second copy as ***"details"*** as it contains only detail information, and the first copy as ***"colors"*** as it contains color information.

8. **Blending Details with Main MH Texture:** In the Layer panel, change the blending mode from "Normal" to "Linear Light" for the "details" texture and turn off all layers except the "details" image and activate the MH texture.

9. **Creation of a Detail Mask:** Select "details" and create a mask. Using a black-colored brush, you can remove unwanted details like hair, ears, or imperfections from the FB texture.

10. **Applying Blurring to the Main MH Texture:** To apply the mask containing the relevant details to the Metahuman texture, selectively remove the details corresponding to the same areas of the face present in the Metahuman texture. In other words, you will selectively replace the details from FaceBuilder on the Metahuman texture.
    Proceed by selecting the mask and executing the *"Subtract Mask from Selection"* command. Select the MH texture and apply *"Filter" -> "Gaussian Blur"*, only on the portion defined by the mask. This way, you will remove the information from the MH texture, leaving only the color information.

11. **Texture Merging:** Merge the resulting MH texture and the "details" layer + mask.

The resulting image will be a texture with details from FaceBuilder and color uniformity from Metahuman.

Figure 4.3: Photoshop Example: Mixed Texture

Blender: Mixed Texture

The Blender project we've created will have a Compositing section with a series of nodes that make it possible to overlay textures from Metahuman and FaceBuilder. The result obtained through this process is an image where the two textures are overlaid based on a mask.

What makes this approach interesting is that the user has full control over the intensity of this overlay, allowing for extremely precise customization.

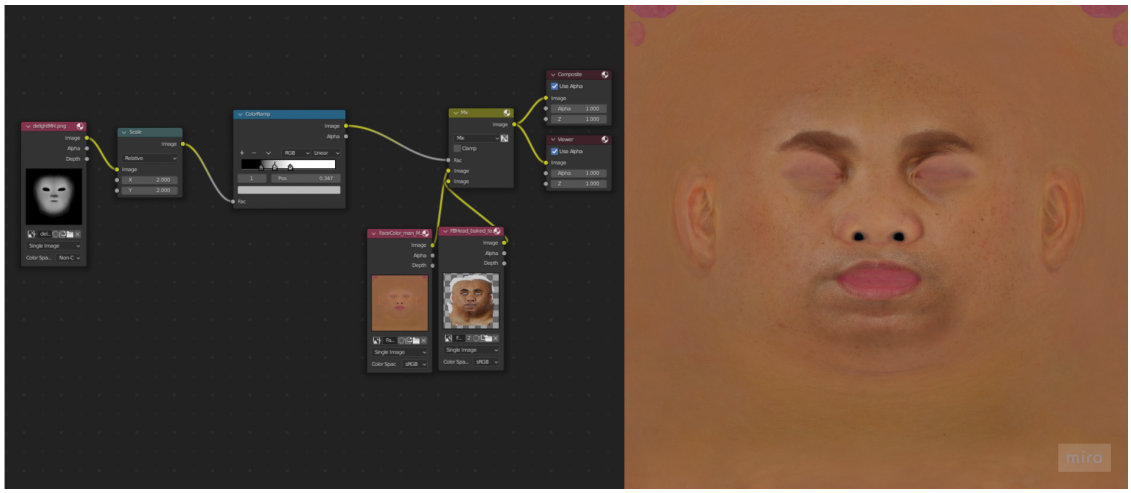


Figure 4.4: Blender Project

It's important to note that, compared to using Photoshop, which offers more detailed control, the Blender approach is more automated, but precision might be slightly lower. However, this doesn't necessarily mean the results are of lower quality. In fact, in many situations, this Blender method can produce good-quality textures, especially if the initial texture generated by FaceBuilder is of high quality.

The choice between the two methods can depend on personal preferences, the quality of the initial texture, and/or the desired results.

In the following Figure 4.5, the extended workflow up to this point is depicted, including the new Texture Reconstruction phase.
The process begins with the selection of material from the RAI archive, marking the initial phase. The acquired material is subsequently transmitted to the Orchestrator.

The Orchestrator oversees the initial "Frame Extraction" phase, where essential images for the subsequent face reconstruction are identified. Following this, these extracted images undergo a "Super Resolution" phase to enhance their quality. The culmination is the "Face Reconstruction" phase, where the processed images are used to generate a 3D mesh of the face along with the corresponding texture.

Subsequently, the 3D mesh and texture are integrated into Unreal Engine through the "Mesh To Metahuman" process to create the MetaHuman. Later on, the texture obtained from Metahuman Creator is combined with that derived from the "Face Reconstruction" phase. This fusion occurs through two distinct approaches: a manual one in Photoshop and a semi-automatic one in Blender.

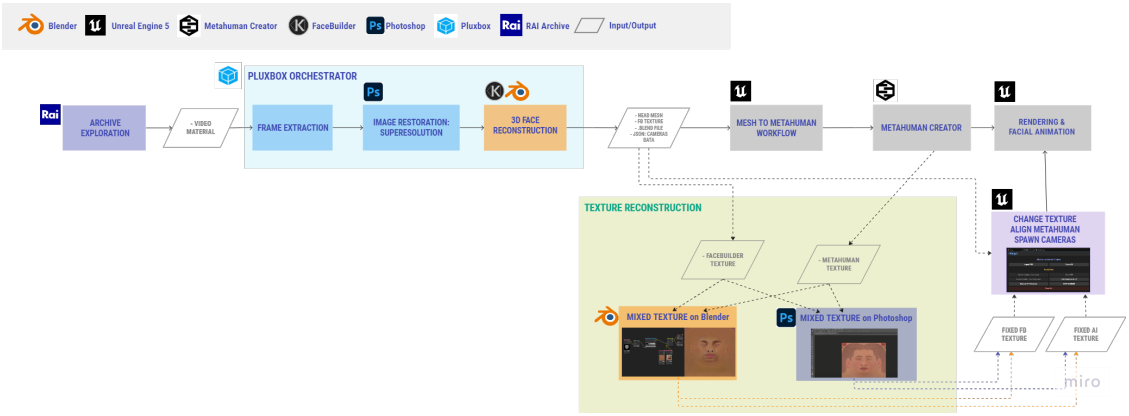


Figure 4.5: Synthetic Humans WorkFlow with Texture Reconstruction

4.4 Images with Generative AI

An innovative solution to address challenges in texture creation is the use of images generated through artificial intelligence (AI) as a starting point for applying in FaceBuilder. Leveraging AI-generated images can potentially overcome issues related to the quality and diversity of reference images for ***Texture Reconstruction***. These AI-generated images can encompass a wide range of angles and lighting conditions, ensuring a high level of quality. This approach enables the achievement of high-quality textures directly from FaceBuilder.

This solution does not exclude the use of FaceBuilder or Metahuman Creator but seamlessly integrates with them. AI-generated images serve as a starting point for FaceBuilder in texture generation, while textures created by Metahuman Creator can be utilized to enhance consistency and realism.

In the following sections, we will delve into the techniques and models used in image generation through the application of artificial intelligence (AI), providing a clear overview of the potential and challenges of these innovative methodologies. We will address three key aspects: **Stable Diffusion** [45] and the integration of **ControlNet with OpenPose** [46] [47] for pose definition, and the training of models based on the **LoRA Model** [48].

4.5 Stable Diffusion

Stable Diffusion [45] is a text-based latent diffusion model for image generation.
It was created by researchers and engineers from CompVis (at LMU Munich), Stability AI, and LAION.

Stable Diffusion is a **text-to-image model**, meaning that when provided with a text prompt, it can generate an AI-generated image that corresponds to the given text.

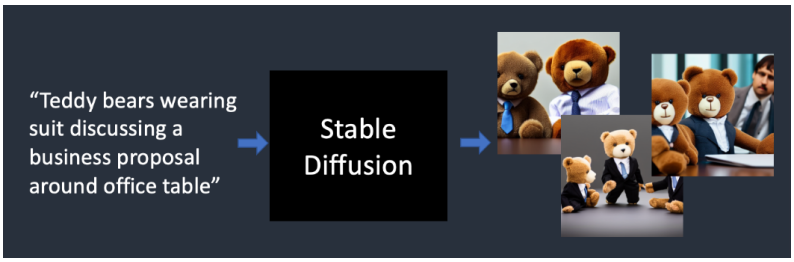


Figure 4.6: Stable Diffusion Example

4.5.1 Diffusion Model

Stable Diffusion is part of a class of deep learning models called **Diffusion Models**. These models are generative, meaning they are designed to generate new data similar to what they have seen during training. In the case of Stable Diffusion, the data in question is images.

The training of this diffusion model is divided into **Forward Diffusion** and **Reverse Diffusion**.

Forward Diffusion

The process of **Forward Diffusion**, also known as *direct diffusion*, represents the first part of the concept in the Stable Diffusion model.
During this process, each training image, in this case images of a cat or a dog, is gradually transformed by adding noise to it. The "forward diffusion" process gradually turns each starting image into a noisy image that lacks distinctive features.
The resulting image becomes a kind of "noise" that no longer reveals whether it initially depicted a cat or a dog.

Figure 4.7: Forward Diffusion of an image depicting a cat

Reverse Diffusion

"Reverse Diffusion", or inverse diffusion, constitutes the second fundamental part of the Stable Diffusion model. During this phase, starting from a noisy and meaningless image, the model can reverse the diffusion process and generate an image corresponding

to, for example, a cat in our previous case.
This result is determined by the "drift" component of the diffusion process, which guides the result towards a specific type of image.

Model Training

To realize the concept of "Reverse Diffusion," it is necessary to understand how much noise has been added to the image. To do this, a neural network model must be trained to predict the level of noise added. This model is known as the "noise predictor" within the context of Stable Diffusion and is implemented as a **"U-Net"** model.

The training process proceeds as follows:

1. A training image is selected.
2. An image containing random noise is generated.
3. The training image is compromised by adding noise to it for a certain number of steps.
4. The **noise predictor** is trained to determine how much noise has been added to the image, tuning its weights and providing it with the correct answers.

At the end of this training process, we obtain a **noise predictor** capable of accurately estimating the level of noise added to an image, as can you see in the figure 4.8.

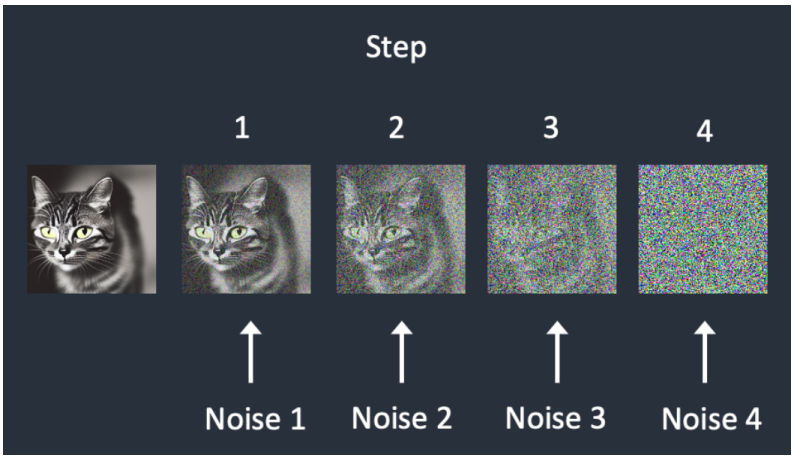


Figure 4.8: The noise is added sequentially at each step.

Once the **noise predictor** is trained, the next step is to understand how to use it. The process of **Reverse Diffusion** involves the following steps:

1. Firstly, a completely random and meaningless image is generated, and the **noise predictor** is asked to calculate the level of noise present.
2. Subsequently, the estimated noise is subtracted from the original image.

3. This process is repeated multiple times, ultimately producing a noise-free image.

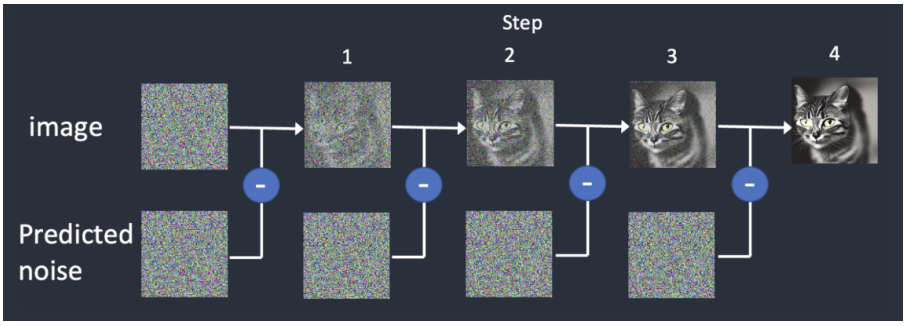


Figure 4.9: Reverse diffusion works by successively subtracting the predicted noise from the image.

It's important to note that currently, the image generation occurs without being guided by any specific request.

4.5.2 Stable Diffusion Model

However, a drawback of diffusion models is that the reverse denoising process is slow due to its sequential and repetitive nature. Additionally, these models consume a lot of memory because they operate in pixel space, which becomes vast when generating high-resolution images.

The **Stable Diffusion** is a **Latent Diffusion Model (LDM)** [45] that operates in a lower-dimensional space, as opposed to the pixel image space, known as the "latent space."

In more technical terms, Stable Diffusion employs an approach based on a neural network called a **Variational Autoencoder (VAE)**, where an image is compressed into a low-dimensional latent space. During the generation process, the network works within this latent space, gradually adding the necessary noise to produce the final image.
This approach significantly reduces computational complexity and enables efficient generation of high-resolution images, representing a significant advancement in the field of text-based image generation.

Variational Autoencoder (VAE)

The Variational Autoencoder (VAE) consists of two main components: an encoder and a decoder.

The encoder's task is to compress an image into a lower-dimensional representation within the latent space. Meanwhile, the decoder performs the inverse operation, reconstructing the image from the latent space back to its original form.

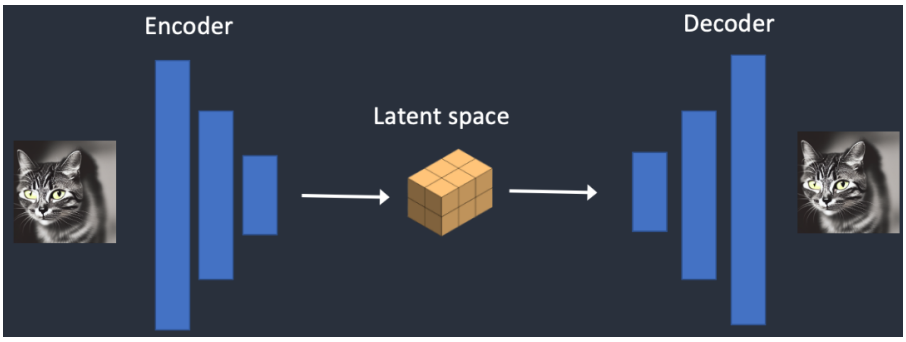


Figure 4.10: Stable Diffusion Example

In the Stable Diffusion model, the dimension of the latent space is 48 times lower than the pixel space of the original image.
During the training process, instead of generating a noisy image, the model produces a random tensor in the latent space, known as "latent noise." This significantly contributes to the efficiency of the generative process and the rapid generation of high-quality images.

The process of **Reverse Diffusion** in the latent space in Stable Diffusion follows these steps:

1. A random matrix is generated in the latent space.
2. The noise predictor estimates the noise of the latent matrix.
3. The estimated noise is then subtracted from the latent matrix.
4. Steps 2 and 3 are repeated for a specific number of sampling steps.
5. The VAE decoder converts the latent matrix into the final image.

4.5.3 Text Conditioning (text-to-image)

Currently, the understanding of how the Stable Diffusion model works is still incomplete because Stable Diffusion cannot yet generate images from text.
Conditioning is what solves this problem. Its purpose is to guide the noise predictor, through text, so that the subtraction of the predicted noise gives us what we desire from the image.
Let's see how a text prompt is processed and input into the noise predictor.

Tokenizer

First, the text prompt is tokenized by a CLIP tokenizer, which is a deep learning model developed by OpenAI to generate textual descriptions for any image.
The tokenizer converts each word in the prompt into a number called a token and is capable of recognizing or breaking down words it has learned during training.

Embedding (Textual Inversion)

Each token is then transformed into a 768-value vector called an embedding. The embeddings are fixed by the CLIP model, which is learned during training.
Using embeddings is important for words that can be used interchangeably.

Text Trasformer

The embedding needs further processing by the text transformer before being provided to the noise predictor. The transformer acts as a universal adapter for conditioning. In this case, it can take input text embedding vectors, but it can also receive other types of data, such as images and depth maps.
The transformer not only processes the data further but also provides a mechanism to include various modes of conditioning.

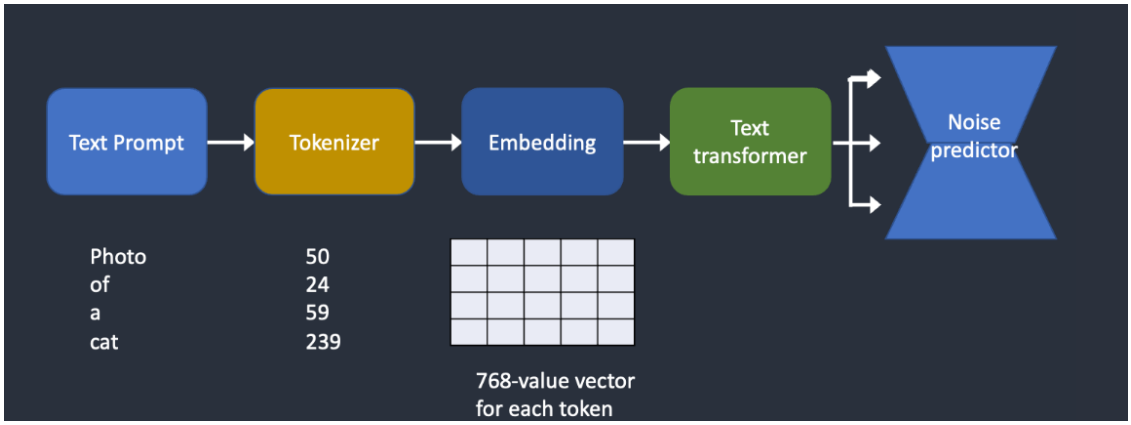


Figure 4.11: How text is processed and input to the Noise Predictor to guide image generation.

You can add or reduce the emphasis of words by enclosing them in parentheses, and you can also use **negative prompts** to specify what the model should avoid when generating the image.

Text prompts are not the only way to condition a Stable Diffusion model.
ControlNet can condition the noise predictor with information like human poses, achieving excellent control over image generation. This means there are many different ways to influence the image generation process, making the Stable Diffusion model highly flexible and controllable.

4.6 ControlNET

ControlNet is a neural network that controls image generation in Stable Diffusion by adding additional conditions [46].

As mentioned earlier, Stable Diffusion models are used for text-to-image generation, using a text prompt as a condition to guide image generation.
ControlNet adds an additional condition beyond the text prompt, which can take many forms, but for our purpose, we will focus on **human pose detection** using **OpenPose**.

4.6.1 OpenPose

OpenPose [47] is a quick keypoint detection model for the human body, capable of extracting positions of hands, legs, and the head, without capturing other details such as clothing, hairstyles, and backgrounds.

Below, in figure 4.12, is the workflow of ControlNet using OpenPose. Keypoints are extracted from the input image using OpenPose and saved as a control map containing the keypoint positions. These are then provided to Stable Diffusion as an additional condition, alongside the text prompt. Images are generated based on these two conditions.

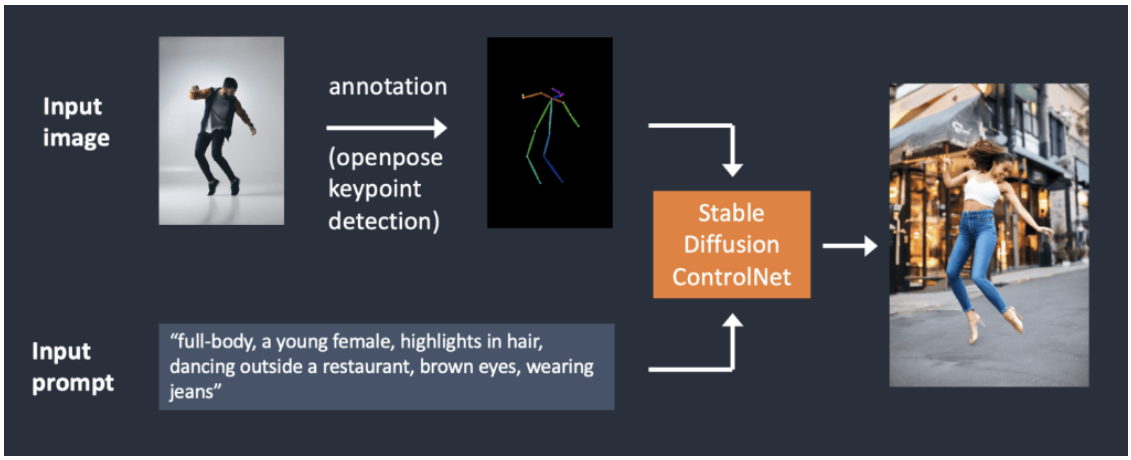


Figure 4.12: ControlNet Workflow using OpenPose

Given our ultimate goal of reproducing AI images of faces of the subjects for creating the Synthetic Human, we should focus on the preprocessor ***OpenPose_face***, which does everything that the OpenPose preprocessor does, such as detecting the positions of the eyes, nose, neck, shoulders, elbows, wrists, knees, and ankles, but it also detects additional facial details. Alternatively, you can choose the ***OpenPose_faceonly*** preprocessor if you are mainly interested in face detection and not other keypoints.

4.7 LoRA Model

The **LoRA Model** [48], which stands for **Low-Rank Adaptation Model**, are a training technique aimed at ***fine-tuning*** *Stable Diffusion models*, primarily used for

image generation.

The concept of **fine-tuning** models refers to the practice of training an image generation model to produce faithful representations of specific subjects or styles. Instead of starting a training process from scratch, with the associated costs and complexity, you begin with an existing model like Stable Diffusion and subject it to additional training using a much smaller dataset composed of images of the desired subject or style. This creates a model that excels at both generating realistic images in general and generating specific images of the chosen subject.

LoRA models are essential for the fine-tuning process of Stable Diffusion models. These smaller-sized models (typically ranging from 1 MB to 200 MB) are combined with pre-existing Stable Diffusion checkpoint models to introduce new concepts to the models, enabling them to generate images incorporating these concepts.
These new concepts can broadly fall into two main categories: *Subjects* and *Styles*.

- ***Subjects:*** "Subjects" can include representations of real or fictional people, facial expressions, poses, objects, environments, and more. For example, a LoRA model can be trained to generate realistic images of a specific individual.
- ***Styles:*** "Styles" encompass visual aesthetics, artistic styles, and characteristics of specific artists. This allows for the generation of images with a particular visual style.

A key element in the operation of LoRA models is optimizing the ***cross-attention module***, where the image and "prompt" meet within the Stable Diffusion model. Modifications to this module enable the model to generate images based on the newly introduced concepts. Refining the cross-attention module has proven to be an effective strategy, as researchers have found it sufficient for achieving successful training.

The cross-attention levels are represented by the yellow parts in the Stable Diffusion model's architecture in figure 4.13. These levels contain weights organized in matrices, and a LoRA model refines the base model by adding its weights to these matrices.
A distinctive feature of LoRA is the ability to decompose a matrix into two smaller, low-rank matrices, addressing storage space issues.

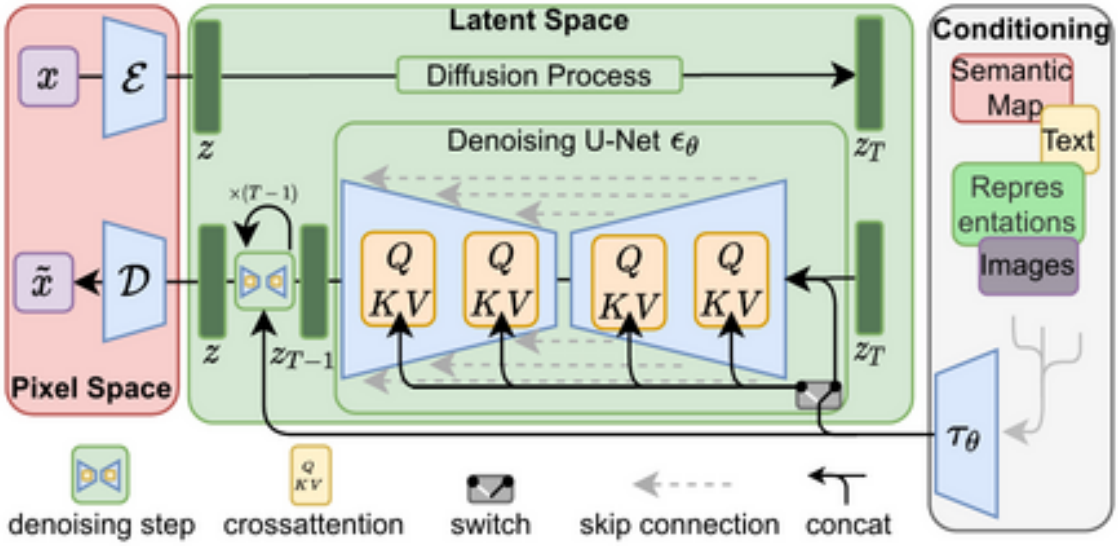


Figure 4.13: LoRA optimizes the cross-attention levels of the U-Net noise predictor. [49]

It's important to note that a LoRA model cannot function independently. It must be used in combination with a Stable Diffusion model checkpoint file since LoRA makes changes to the existing model's styles. In practice, when using a LoRA model, you need to specify both the LoRA model and the base Stable Diffusion model.
In the specific context, the training model known as ***"Realistic Vision V5.1"*** [50] was utilized to generate high-quality and detailed AI images, driven by the necessity to represent human faces realistically.

4.8 Image Generation for Texture Creation

After examining the technologies and methodologies used in image generation through artificial intelligence (AI), it is crucial to focus on the main goal of this process. The central objective is to obtain detailed, high-quality, and consistent images that serve as a solid foundation for creating Textures.
This phase of image generation aims to overcome various challenges and achieve several objectives:

- **High-Quality Texture:** The primary goal is to produce highly detailed, high-resolution textures. This is imperative to ensure that the final texture of the Synthetic Human is realistic and faithfully represents the reference subject.
- **Uniformity in Lighting Conditions:** It is essential to generate images with consistent lighting conditions to avoid color discrepancies within the texture, ensuring that the generated images are consistent in terms of lighting.
- **Flexibility in Angles:** To ensure the creation of 360° head models, it is crucial to have images from various angles of the subject. Image generation through AI,

using the mentioned techniques, especially ControlNET with OpenPose, allows the generation of subject images in the same poses given by the reference image.

- **Variation in Age and Conditions:** Artificial intelligence offers the ability to generate subject images at various desired ages and conditions, overcoming the limitation of the limited availability of photographic material.

To illustrate how we have applied the aforementioned techniques for creating new images for use in Texture Reconstruction, consider an example aimed at obtaining images depicting Maria Callas.

The process begins with the training of **LoRA models**, using a dataset consisting of 100 photos of Maria Callas and leveraging the **"Realistic Vision V5.1" checkpoint**. This checkpoint is known for its ability to generate extraordinarily realistic and detailed images.

Once the training is complete, we integrate the LoRA model into the **Stable Diffusion** prompt, defining a set of desired features for the resulting image in the textual prompt. For example, "Generate RAW images of Maria Callas at 40 years old, without hair, neutral expression, neutral lighting, and high quality..."

Simultaneously, in the negative prompt, you can specify the characteristics to exclude from the output image.

To ensure the generation of the output image with the subject in the specific desired poses, we leverage reference images depicting any subject in the same poses we want for Maria Callas. **OpenPose** plays an essential role in identifying the facial keypoints in these different poses, which are then passed to Stable Diffusion through **ControlNET**. This process ensures that the resulting image represents Maria Callas with the features expressed in the prompt and in the poses specified by the reference image.

At the end of this process, we obtain an image of Maria Callas with all the characteristics defined in the prompt, in high quality and in the desired pose. This concrete example demonstrates how the mentioned technologies and methodologies can be employed to generate high-quality and realistic images used for creating Textures for Synthetic Humans, ensuring uniformity, flexibility, and customization.

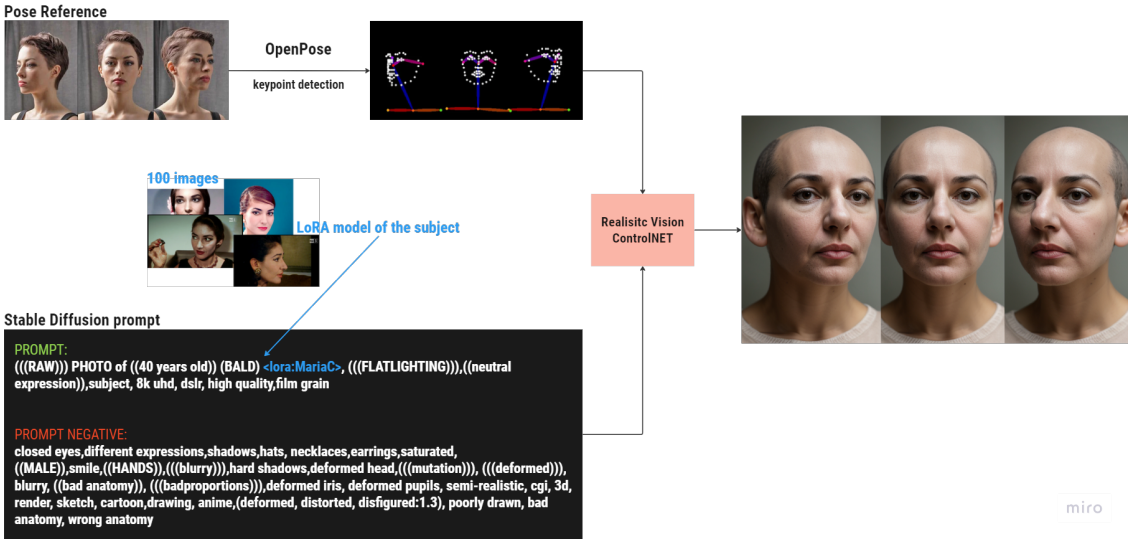


Figure 4.14: Workflow for creating AI images of Maria Callas with Stable Diffusion

Once the images generated through artificial intelligence (AI) are obtained, they can serve as a starting point in the automated phase of Face Reconstruction, utilizing FaceBuilder's Texture Reconstruction algorithm. Textures derived from these AI-generated images exhibit consistent lighting uniformity and reconstruction coherence, providing an initial high-quality texture that requires minimal adjustments for readiness on a Synthetic Human.

(a) Texture obtained from original images (b) Texture obtained from AI images

Figure 4.15: Textures compared

Potential improvements or modifications to the texture derived from AI-generated images through FaceBuilder can be implemented using software applications like Blender (semi-automated) or Photoshop and Procreate (manual).

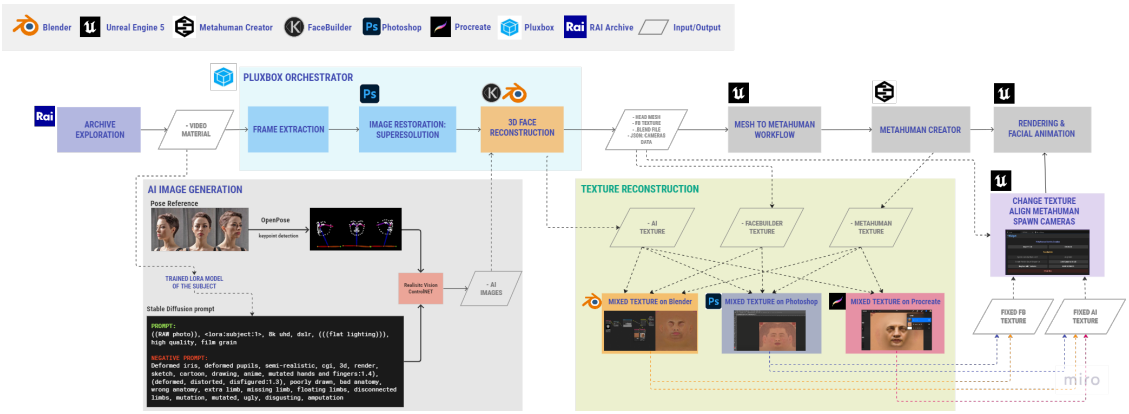


Figure 4.16: Completed Workflow for creating Synthetic Humans

Chapter 5

Model Validation

In this chapter, an evaluation is conducted to determine the reliability and robustness of the automated workflow proposed for generating Synthetic Humans.

The analysis will focus on the fidelity and similarity of the Synthetic Humans to the reference subjects. The evaluation will be conducted using both objective and subjective methods by comparing the original images depicting the subjects with the resulting Synthetic Humans.

5.1 Test Material Preparation

5.1.1 Representative Dataset

To conduct a thorough and meaningful validation of the model, it was crucial to define a representative dataset of Synthetic Humans.
This dataset was designed to encompass a variety of individuals representing different demographic characteristics, including varying ethnicities, ages, and genders. Additionally, for each subject, a collection of 80 to 100 pictures was assembled, depicting them in a range of poses and expressions, from close-ups to distant shots, without strict limitations on the variety of images.
Emphasis was given to Italian subjects, to create a more tailored dataset that aligns with the project's specific goals. The images were sourced from a blend of RAI's archive and online resources, they include a variety of historical periods, resulting in a diverse range of photographs from black and white, lower resolution images to contemporary, high-resolution ones.
The diversification of these features aims to ensure and demonstrate that the workflow is capable of generating Synthetic Humans reflecting the traits of a broad spectrum of individuals in the global population.

In Figure 5.1, photos depicting the subjects selected for the creation of the Synthetic Humans for the dataset are shown.

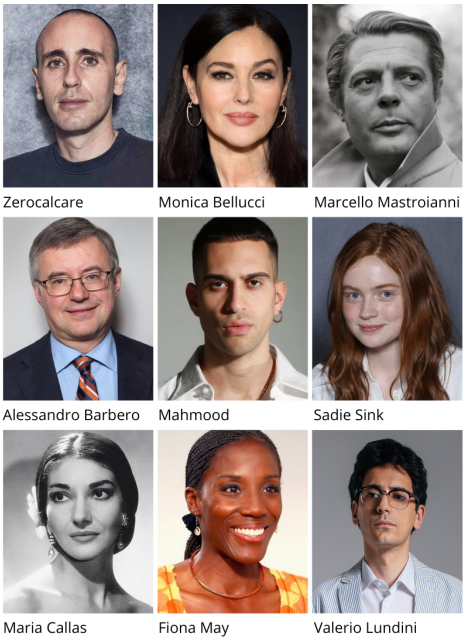


Figure 5.1: The selected subjects to which the workflow has been applied.

For each individual, the following phases were implemented as schematically illustrated in Figure 5.2.

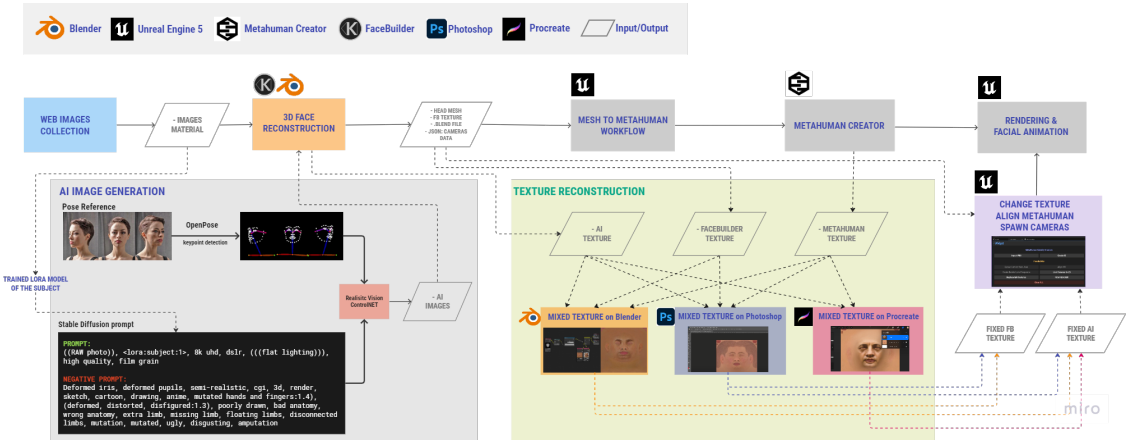


Figure 5.2: The workflow followed

Initially, a set of approximately ninety images depicting the subject of interest was collected. These images were utilized in training a LoRA model designed to generate images of the individual using generative networks. A portion of the images collected for LoRA model training was allocated to the **Face Reconstruction** phase.

It is important to note that, for the generation of Synthetic Humans in this context, the orchestrator developed in collaboration with PluxBox was not employed. Instead, the script named ***"FB_Head_Reconstruction_Automatized.py"***, the same script used in the orchestrator via Rest API, was locally used to automate the creation of the **3D head mesh**. This script automatically generates the 3D model of the individual's face, utilizing the subject's images.

The output of this process includes an ***fbx*** file containing the 3D head mesh, the extracted texture from the images used to build the 3D model, the **Blender scene** (in case additional manual modifications to the mesh are desired), and a **JSON file** containing camera information. The latter is crucial for replicating the same view in Unreal Engine, facilitating the generation of renders of Synthetic Humans and subsequent model validation.

After obtaining the 3D head model from the original images, the same script was used with the images generated by the Generative AI. From the resulting output, only the texture is extracted, which will be used in the testing phase for model validation.

The availability of the 3D head mesh and textures generated by FaceBuilder facilitated the workflow, which includes importing the FBX mesh into Unreal and executing the ***"Mesh To Metahuman"*** procedure. This process utilizes the head mesh obtained during the Face Reconstruction phase as a base for creating a Metahuman.

After obtaining the MH, several facial textures were generated through a series of blending operations. The base textures, that can see in figure 5.3, employed in these operations are as follows:

- **Original MH**: The base texture provided by the Metahuman framework, designed for high compatibility with its models.
- **Original FB**: Textures obtained from FaceBuilder mapping using the subject's web original images.
- **Original AI**: Textures generated by FaceBuilder mapping using the subject's Stable diffusion generated images.

The blending process involved combining the base textures to address any gaps and ensure a cohesive appearance. The Original MH texture was essential in this process as it helped fill in any missing details and provided a consistent base for the blend. The following table 5.1 provides an overview of the blending methods and the specific textures that were combined.

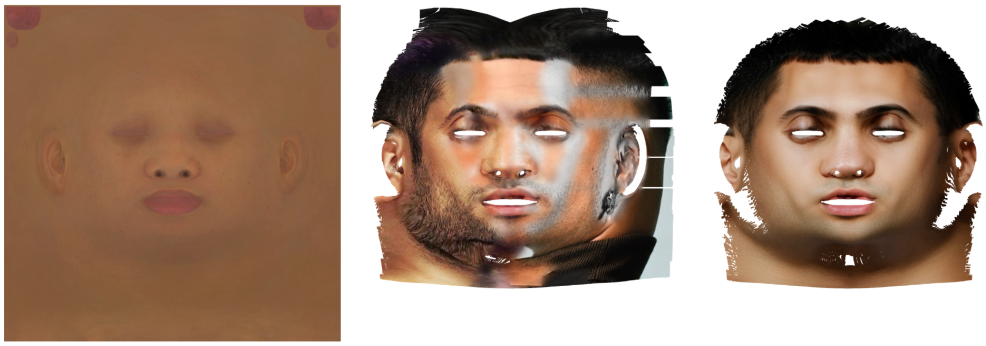


Figure 5.3: The base textures used for the various blending methods

Textures Generated from Original Images			
Name	**Blending Method**	**Texture A**	**Texture B**
Original_Blender_Mix	Blender	Original FB	Original MH
Original_Photoshop_Mix	Photoshop	Original FB	Original MH
Textures Generated from AI Images			
Name	**Blending Method**	**Texture A**	**Texture B**
Procreate_AI	Procreate	Original MH	Original AI
Blender_AI	Blender	Original MH	Original AI
Photoshop_AI	Photoshop	Original MH	Original AI

Table 5.1: Texture blending methods overview

During the testing phase, all obtained textures were used to objectively and subjectively evaluate which texture generation procedure is more effective in ensuring similarity with the target subject.

The blended textures were imported into Unreal Engine 5, and placed in the corresponding Metahuman project folder.
Through the implementation of a **Widget** in Unreal several processes were automated: loading of the Metahuman into the scene, instantiating cameras to match the views from the Face Reconstruction images (guided by the JSON file), and swapping textures on the Metahuman.
Consequently, this automation facilitated the Metahuman's positioning in the same poses used during the Face Reconstruction phase, allowing for a direct comparison between the original image and the Metahuman rendering.

5.1.2 Evaluation Metrics

Validating our Synthetic Humans generation model required a careful approach to identify metrics that would reflect human perception. A significant part of our work was dedicated to studying and understanding which metrics could effectively capture the human experience in evaluating generated faces.

In addition to objective metrics, we conducted a subjective validation through a **survey** conducted on a sample of **46 participants**. This allowed us to gain a human perspective on facial resemblance and to support our choice of objective metrics.

Survey: Subjective Evaluation

To obtain an assessment reflecting human perception of the generated Synthetic Humans and the intermediate stages used to achieve our goal, we conducted a survey with a significant number of participants. The survey questions were strategically formulated to mirror the same aspects examined through objective metrics. This approach provided us with a complementary and human perspective on the fidelity of synthetic faces.

Analysis of Objective Metrics

In the model validation process to measure the effectiveness of the system in generating Synthetic Humans, we focused on two objective metrics: the **Cosine Similarity** and the **Euclidean distance**.

Cosine Similarity Metric

The choice of the **Cosine Similarity metric** [51] is based on its ability to measure the similarity between high-dimensional vectors, making it particularly suitable for complex data such as facial images. The cosine similarity metric is a method for calculating the similarity between vectors based on the angle between them in the vector space. The calculation is performed using the cosine formula:

$$\cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}$$

This formula calculates the angle (θ) between two vectors ($\boldsymbol{a}$ and $\boldsymbol{b}$) and returns a value ranging from -1 to 1. A value of -1 indicates that the vectors point in opposite directions, while a value of 1 indicates that the vectors are perfectly aligned. A value of 0 indicates independence between the vectors.

This metric is widely used in the field of facial recognition because it reflects how humans perceive the similarity between faces. When two faces are very similar, we expect their cosine similarity score to approach 1, making it an ideal tool for evaluating the similarity between subjects' faces and Synthetic Humans.
Furthermore, the robustness of the cosine similarity metric to variations in lighting conditions and different facial poses ensures that small changes in the subject's appearance

or pose do not significantly affect the score.

One of the most significant applications of the cosine similarity metric is through the **ArcFace** algorithm.

ArcFace: Maximizing Cosine Similarity

ArcFace [52] is an advanced deep learning-based facial recognition algorithm. Its distinctive feature is the use of the cosine metric to calculate the similarity between features extracted from faces. During the training phase, ArcFace is configured to optimize a specific loss function based on the cosine metric.
ArcFace leverages a loss function called "Cosine Margin Product," designed to maximize the similarity between feature vectors of faces from the same individual (intra-class) and minimize the similarity between feature vectors of different individuals (inter-class). In simple terms, the algorithm calculates the cosine angle between the feature vector extracted from a facial image and vectors representing class centers.
During training, the model learns to position feature vectors of faces from the same person closer to each other in the feature space.
ArcFace is one of the most advanced facial recognition models due to its ability to robust face detection, recognition, and landmark detection capabilities.

InsightFace Library

ArcFace within the InsightFace library [53] is utilized to extract the "landmark_2d_106" array, which consists of 106 pairs of (x,y) coordinates.
Each pair represents the position of a specific facial landmark on the 2D image plane, as can be seen in figure 5.4, covering various facial features such as eyes, eyebrows, nose, mouth, and the contour of the face.

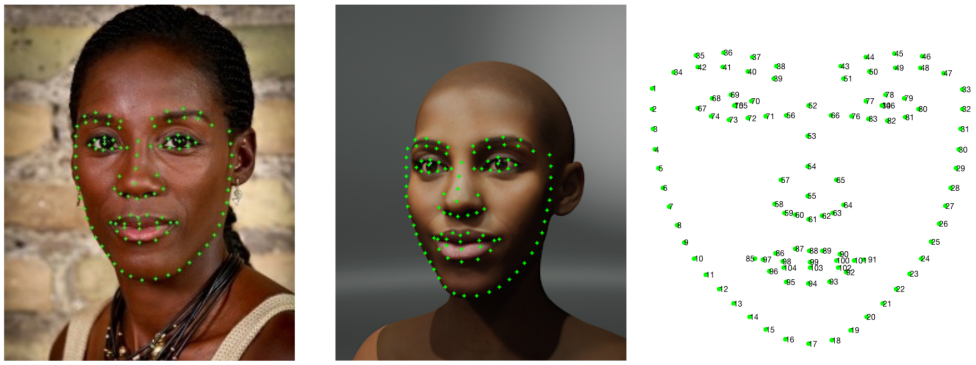


Figure 5.4: Facial landmarks delineated by InsightFace using ArcFace module, showcasing 106 key points for detailed facial feature analysis.

The following code snippet illustrates the process of employing it for extracting high-dimensional embeddings from facial images and calculating the cosine similarity between them:

```
# Path to the input images
sint1 = "<path>/1.png"
sint2 = "<path>/2.png"

# Read the first image and detect face embeddings and 2D
   landmarks
img1 = cv2.imread(sint1)
faces1 = app.get(img1)
face1 = faces1[0]
emb_sint1 = face1.embedding
landmark_2d_1 = face1.landmark_2d_106

# Draw 2D landmarks on the first image
for point in landmark_2d_1:
    x, y = point
    cv2.circle(img1, (int(x), int(y)), 1, (0, 255, 0), -1)
         # Draw a green circle at the landmark point

# Read the second image and detect face embeddings and 2D
   landmarks
img2 = cv2.imread(sint2)
faces2 = app.get(img2)
face2 = faces2[0]
emb_sint2 = face2.embedding
landmark_2d_2 = face2.landmark_2d_106

# Draw 2D landmarks on the second image
for point in landmark_2d_2:
    x, y = point
    cv2.circle(img2, (int(x), int(y)), 1, (0, 255, 0), -1)
         # Draw a green circle at the landmark point

# Calculate the cosine similarity between the embeddings
   of the two faces
cosine_similarity = np.dot(emb_sint1, emb_sint2) / (norm(
   emb_sint1) * norm(emb_sint2))
```

The cosine similarity metric, computed using the dot product of the embeddings divided by the product of their norms, serves as an indicator of the similarity between the two

faces in the embedding space. A higher cosine similarity score suggests a closer match between the facial features represented by the embeddings.

Euclidean Distance

The **Euclidean distance** represents a measure of distance between two points by the length of the vector connecting them. In the specific context of facial recognition between images, Euclidean distance emerges as a crucial tool for assessing the similarity between vector representations of faces extracted from visual data.

Before applying Euclidean distance, facial recognition models adopt advanced feature extraction processes. These processes often involve deep neural networks for identifying and representing the salient features of a face. The result of this process is a multidimensional numerical vector that uniquely captures the peculiarities of the given face.

The Euclidean distance between two such feature vectors is calculated according to the standard formula of Euclidean geometry.

$$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

The sum of the squared differences between corresponding components of the two vectors is then square-rooted to obtain the Euclidean distance. This process allows for a numerical evaluation of the dissimilarity or similarity between the faces represented by the vectors.

One of the main challenges of Euclidean distance in the context of facial recognition lies in its sensitivity to variations. Factors such as changes in lighting conditions, different face angles, and variations in facial expressions can significantly influence the results of Euclidean distance.
In more complex situations, relying solely on Euclidean distance may not be sufficient. Therefore, more advanced similarity metrics, such as the cosine metric, are often integrated to overcome the limitations of Euclidean distance. The robustness of such advanced metrics to environmental variations and subject conditions makes them preferable in more challenging facial recognition scenarios.

As shown in figure 5.5, each row showcases a comparison between an original photograph of a subject (on the left) and a synthetic image generated by a Stable Diffusion model (on the right). For each pair, the Cosine Similarity and Euclidean distance was computed to quantify resemblance, also considering different poses of the subject.
Notably, the results from the Euclidean distance metric predominantly return false, except for the first image of Maria Callas. This outcome indicates that the computed distance for most pairs exceeds the threshold of 1.13, and if the Euclidean distance is greater than the threshold, the algorithm determines that the images are of different person (false). Conversely, a distance below the threshold would suggest that the images

are of the same person (true). This threshold is empirically determined through machine learning techniques, specifically decision trees [54].
Contrasting these Euclidean distance findings, both the cosine similarity measures and survey responses indicate a higher perceived resemblance for the SD generated images.
In the survey, 46 participants were asked to rate the similarity of AI-generated images to the original subjects on a scale from 1 to 5. The survey results were as follows:

- For Valerio Lundini, 67.4% of participants rated the similarity as '5' and 26.1% rated it as '4'.
- For Monica Bellucci, 45.7% of participants rated the similarity as '5' and 54.3% rated it as '4'.
- For Marcello Mastroianni, 70.5% of participants rated the similarity as '5' and 29.5% rated it as '4'.
- For Maria Callas, 63.6% of participants rated the similarity as '5' and 36.4% rated it as '4'.

These ratings highlight a significant discrepancy between the quantitative Euclidean distance measures and the qualitative assessments of human observers. It was concluded that **the cosine similarity metric**, through the **ArcFace model**, was **more aligned with human perception** than Euclidean distance.

Therefore, the subsequent steps of analysis and presentation, will focus exclusively on the results obtained from this metric.

5.2 Results of Workflow Automation

Before proceeding with the analysis and comparison of Synthetic Humans derived from the selected subjects in our dataset, we will thoroughly examine some intermediate stages within the Synthetic Human creation process. This in-depth exploration will allow us to demonstrate step by step the robustness of the methods employed, culminating in the realization of the MetaHuman.

To objectively assess each of these stages, we will use the **objective metric** of **cosine similarity**, employing the facial recognition algorithm **ArcFace**.
As previously mentioned, ArcFace stands as a solid and human-perception-conforming approach to evaluate the fidelity of the generated Synthetic Humans.

Simultaneously, we will also consider a **subjective metric** based on the results of a **survey** conducted with a sample of N participants. As mentioned before, the survey questions have been structured to reflect the same aspects examined through the objective metrics, offering a complementary perspective to the evaluation of the fidelity of the generated Synthetic Humans.

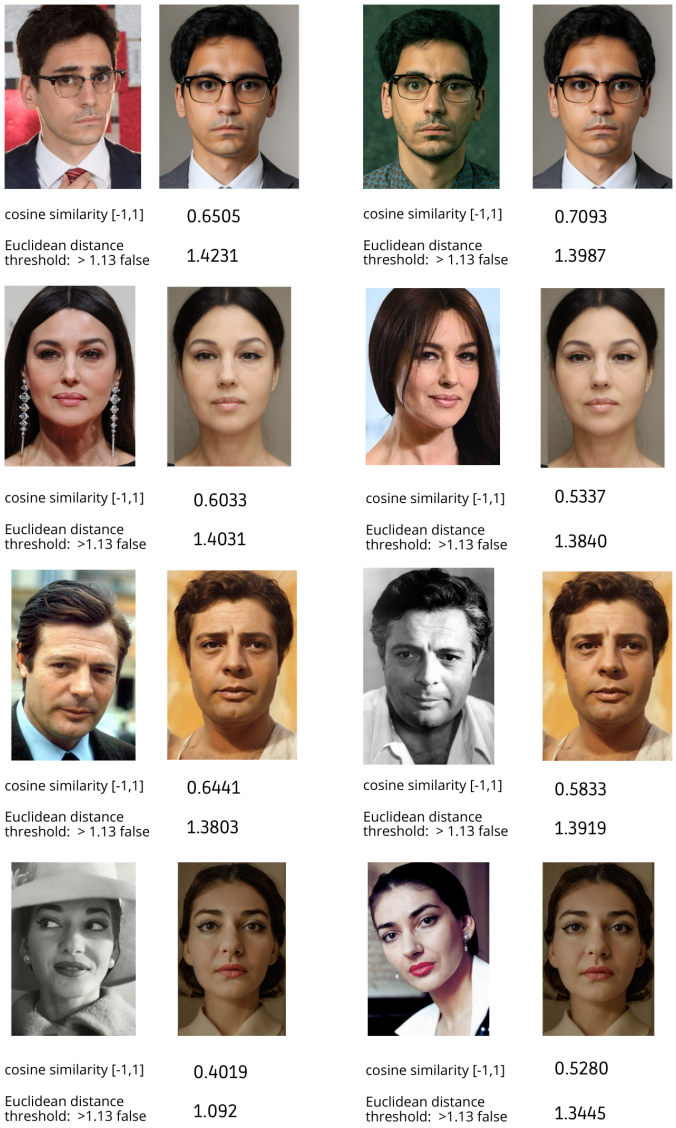


Figure 5.5: Some Cosine Similarity and Euclidean distance results.

5.2.1 Validity of Images Generated through Artificial Intelligence

First and foremost, let's focus on the validation and comparison of images generated through Stable Diffusion generative network with the corresponding original images of the individual. This comparison was made using the cosine similarity metric and collecting opinions through a survey on the resemblance of the original subject to the images generated by artificial intelligence, you can observe some results in the figure 5.6.
The results obtained through the cosine similarity metric consistently exceeded 0.5, a significant threshold beyond which the probability that two subjects are the same person is very high since it suggests a high likelihood of facial match [52].

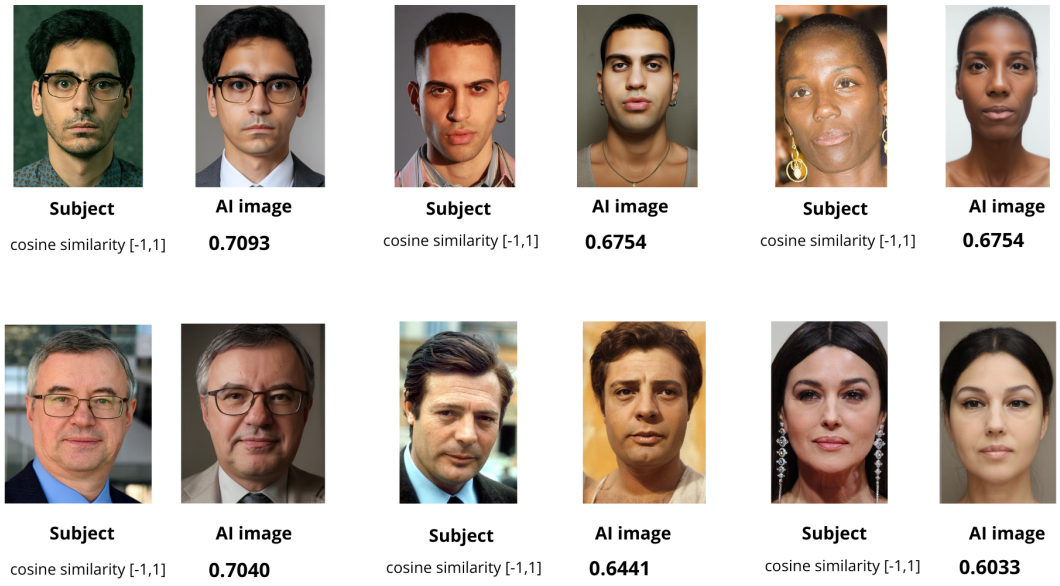


Figure 5.6: Some Results of the cosine similarity metric using ArcFace

In parallel with the cosine similarity assessment, as said before we conducted a survey asking participants to *"Assign a score from 1 to 5 to express how much you think the AI-generated image resembles the original character"*. Based on the responses of 46 participants, the score of 5, which represents the highest level of resemblance, was given by approximately 68% of respondents when averaged across all subjects.

This demonstrates the remarkable capability of the generative AI in producing images that closely match the original characters in appearance.

Subject	**Cosine Similarity [-1,1]**
Alessandro Barbero	0.7040
Monica Bellucci	0.6033
Maria Callas	0.5280
Fiona May	0.6754
Marcello Mastroianni	0.6441
Mahmood	0.6754
Sadie Sink	0.6619
Valerio Lundini	0.7093
Zerocalcare	0.7065

Table 5.2: Cosine Similarity Scores of the AI generated Images

5.2.2 Validity of 3D Meshes Obtained during the Face Reconstruction Phase

In this section, we will carefully examine the validity of the 3D meshes obtained during the automated face reconstruction phase, using the script *"FB_Head_Reconstruction_Automatized.py"*.

To conduct this evaluation, we will focus on the comparison between an original image of the subject and the corresponding generated 3D mesh, maintaining the same pose as the subject in the photograph.

This comparison, similar to the previous case, was performed using the cosine similarity metric, measuring the affinity between the rendered 2D image of the 3D head mesh and the original images of the subject, you can observe the results in the figure 5.7 and in the table 5.3.
As before, we collected opinions through a survey asking *"Assign a score from 1 to 5 to express how faithful you think the 3D model of the face is to the original character"*. This was done to evaluate the perceived accuracy of the 3D facial models generated during the face reconstruction process compared to the original subjects.

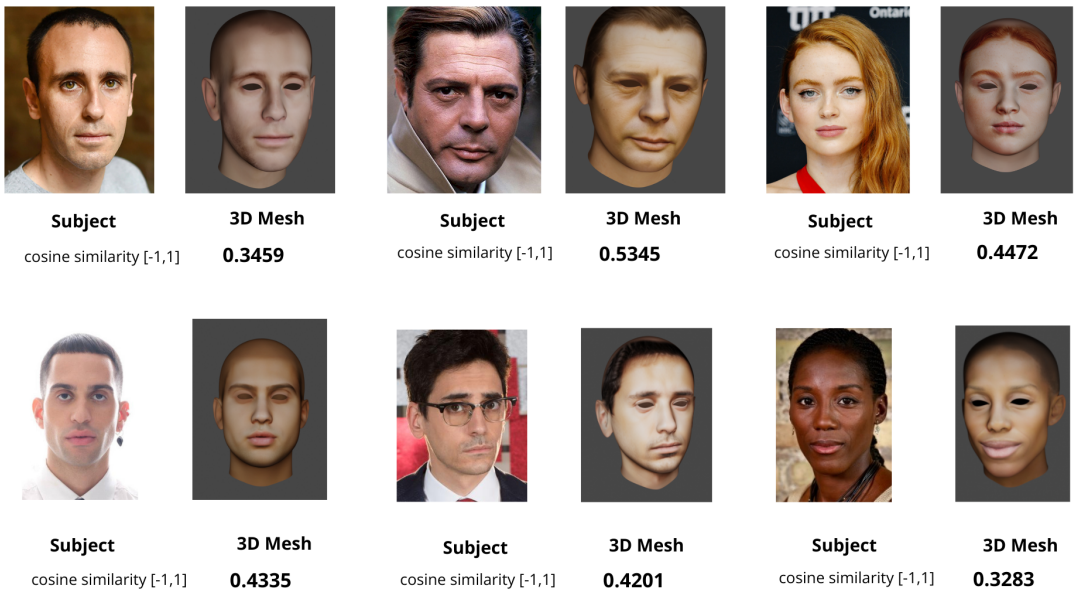


Figure 5.7: Some Results of the Cosine Similarity Metric Using ArcFace for 3D Meshes

The cosine similarity scores between the original 2D images and their corresponding 3D mesh reconstructions generally approached the 0.5 mark, which is indicative of a reasonable resemblance.

Subject	Cosine Similarity [-1,1]
Alessandro Barbero	0.3545
Monica Bellucci	0.2857
Maria Callas	0.1211
Fiona May	0.3283
Marcello Mastroianni	0.5345
Mahmood	0.4335
Sadie Sink	0.4472
Valerio Lundini	0.4201
Zerocalcare	0.3459

Table 5.3: Cosine Similarity Scores of the 3D Mesh

It's noteworthy that the slight decrease in cosine similarity when comparing 2D images to 3D meshes can be attributed to several factors like the inherent difference between a 2D image and a 3D representation. Moreover, the original images are not always captured in a frontal pose, Arcface model is susceptible to head rotation, which can impact the direct comparison. Despite these challenges, the 3-D meshes maintain a high level of fidelity to the original subjects, as the survey results also show.
In fact, the survey results give correlated results with those obtained with cosine similarity: 33.82% of participants rated the similarity with a score of 5, while a notable 29.95% assigned a score of 4. This demonstrates that a significant majority of respondents recognize a strong likeness between the 3D models and the original images

5.2.3 Validity of Synthetic Humans

In this section, we will proceed with the evaluation of the Synthetic Humans in the dataset, comparing them with an original image of the represented subject. In particular, we will test the different textures obtained from the previously analyzed processes on the MetaHuman.
This analysis aims to identify the texture creation process that produces the most satisfactory results in determining which of these procedures proves most effective in maintaining similarity to the reference subject in the MetaHuman.

This analysis, as in previous cases, was conducted using the cosine similarity metric, measuring the affinity between the face of the original subject and the face of the MetaHuman in relation to different textures. Simultaneously, we collected opinions through a survey aimed at evaluating the texture that performs better, is more realistic, and is more similar to the original subject.

Before proceeding with the analysis of the results related to the Synthetic Humans, generated by the previously examined 3D head models that have shown good performance, it is important to note that, once imported into MetaHuman, they undergo changes that can influence the face's geometry. These changes may result in the loss of specific details.

Variations in the shape of the MetaHuman's face compared to the original 3D meshes can be attributed to the "Mesh to Metahuman" process that transforms an FBX 3D mesh of a face into a MetaHuman. This process has some limitations in the customization options offered by MetaHuman Creator, which may affect the complexity and variety of replicable facial shapes. The default nature of MetaHuman Identity implies that customization options for face shape may be more limited than the richness of details contained in the original mesh. During conversion, the software tries to adapt the existing geometry to the structure of a MetaHuman, based on the available customization options, causing discrepancies in face shape during the conversion process.

After this clarification, which will partly determine the results obtained from objective metrics, let's analyze and comment on the values obtained.

Subject	**Or. MH**	**Or. FB**	**Or. Blender Mix**	**Or. Photoshop Mix**
Alessandro Barbero	0.1038	0.2688	0.1847	0.0852
Monica Bellucci	0.0832	0.3444	0.2929	0.1939
Maria Callas	0.0702	0.1760	0.1145	0.0950
Fiona May	0.0311	0.3166	0.1874	0.0651
Marcello Mastroianni	0.0687	0.3442	0.2447	0.1005
Mahmood	0.0517	0.4258	0.3413	0.1271
Sadie Sink	0.1194	0.2930	0.1879	0.0983
Valerio Lundini	0.1556	0.3447	0.3306	0.1377
Zerocalcare	0.0359	0.3108	0.2166	0.0201
Mean	0.0800	**0.3138**	0.2334	0.1025

Table 5.4: Cosine Similarity Values for Subjects Across Different Methods using Original FB and Original MH as base textures (for textures refer to the table 5.1)

Subject	**Original AI**	**Procreate AI**	**Blender AI**	**Photoshop AI**
Alessandro Barbero	0.1539	0.2084	0.1292	0.0845
Monica Bellucci	0.3155	0.3333	0.2857	0.1989
Maria Callas	0.1125	0.0957	0.0649	0.1285
Fiona May	0.2565	0.2110	0.1103	0.0912
Marcello Mastroianni	0.5506	0.4180	0.3820	0.2940
Mahmood	0.4032	0.3781	0.3150	0.1524
Sadie Sink	0.2384	0.3095	0.1737	0.1249
Valerio Lundini	0.3127	0.3480	0.2377	0.1391
Zerocalcare	0.2469	0.1945	0.1453	0.0297
Mean	**0.2878**	**0.2774**	0.2049	0.1381

Table 5.5: Cosine Similarity Values for Subjects Across Different Methods using Original AI and Original MH as base textures (for textures refer to the table 5.1)

As highlighted in Tables 5.4 and 5.5, the results obtained through the cosine similarity metric are not always good, one of the reasons for this is certainly the loss of information about face geometry once the MetaHuman is generated, as anticipated earlier.

The ***Original_FB*** 5.4 textures, stands out with an average cosine value of 0.31, which keeps most of information from the original web photos used to build the model, but results visually not particularly realistic as it is a collage of multiple photos;
The ***Procreate_AI*** and ***Original_AI*** 5.5 also show noteworthy performance, with average cosine values of 0.27 and 0.28, respectively. The latter contains all the information obtained from AI-generated images mapping, meanwhile the other one is similar but with some inaccuracies that are fixed manually on Procreate.

In the survey section, participants were presented with the task *"Select one or more images that you think represent a realistic facial texture from the available options"*, where they could choose up to three textures per subject, the render used for each subject in the survey are like the ones showed in 5.8.

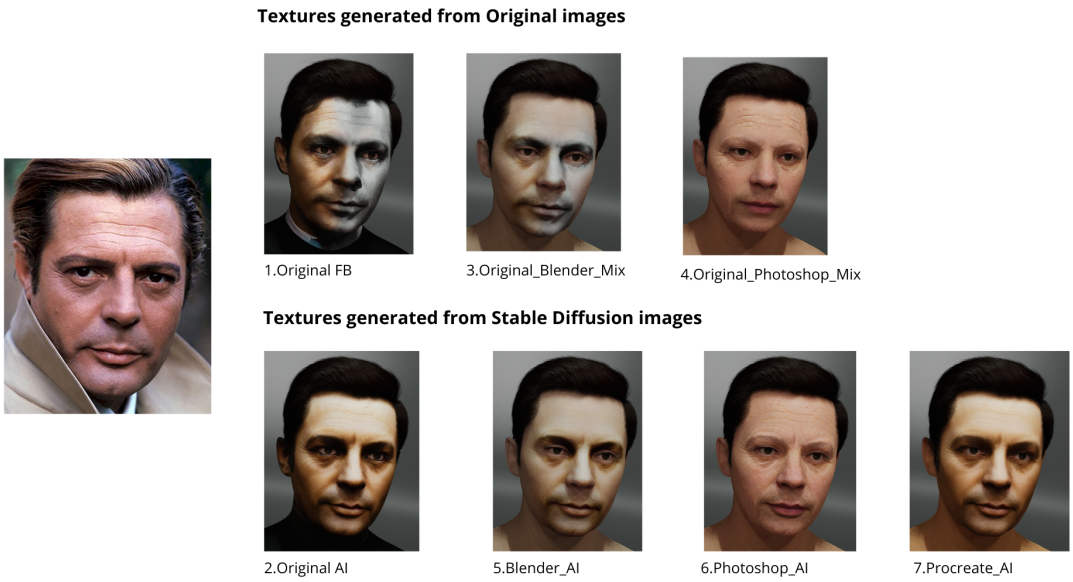


Figure 5.8: An Example of the proposed MH with different textures applied in the survey.

In the survey each MH render is presented following a random order and numbered from 1 to 7, with no reference to the specific texture used. This approach was designed to avoid biasing participants' choices, allowing an assessment solely based on visual appeal and perceived realism, rather than preconceived notions about the textures themselves.
As for the survey results, statistically, the preferred textures in terms of realism and resemblance to the original individual are Texture 6 **(Photoshop_AI)** and Texture 7

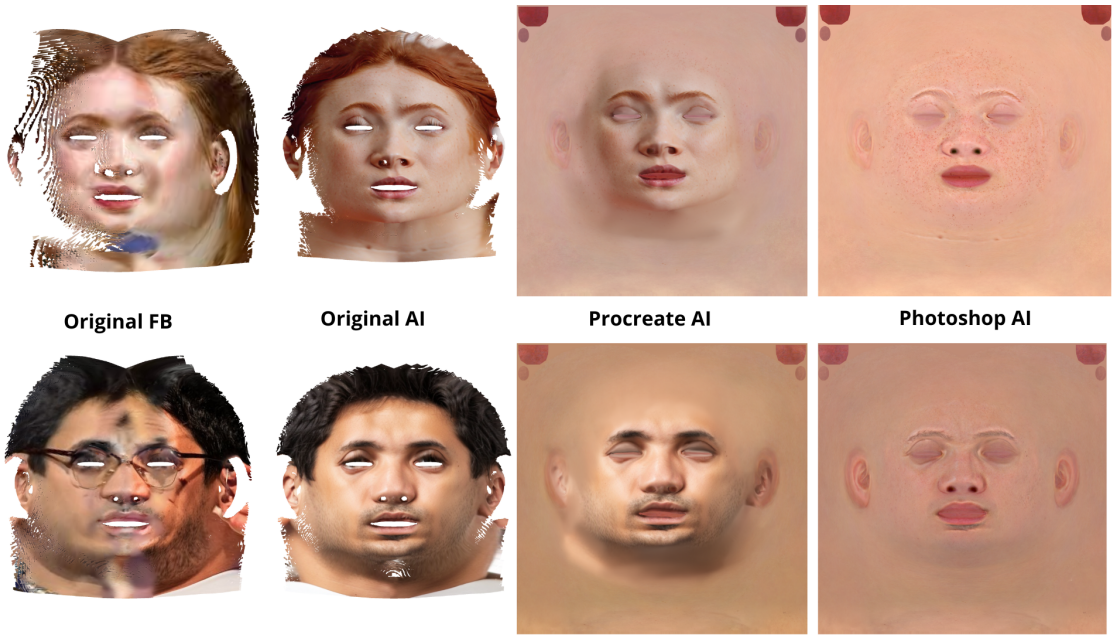


Figure 5.9: Some of the best performing facial texture maps

(Procreate_AI).
Given that the total number of votes for this section was 690 with, Photoshop_AI received 183 votes, resulting in 26.52% and Texture 7 Procreate_AI received 206 votes, leading to 29.86%.
This indicates that despite the initial higher cosine similarity of Original_FB textures in representing details from the original photos, participants showed a stronger preference for textures that realistically blend AI-generated details with manual corrections.
Also, it emerges that the texture that ranks among the best in both metrics is **Procreate_AI.**

Metahuman Creator VS Character Creator 3: Comparative Analysis

In the context of creating photorealistic avatars, **MetaHuman Creator** and **Character Creator 3** [36] stand out as the main options.

In light of the not entirely satisfactory performance of MetaHuman, Character Creator 3 was considered since it's renowned for its features and flexibility, presented itself as a promising alternative.
The avatars resulting from the creation process with Character Creator 3 (CC3) underwent the same evaluation protocol as used earlier, ensuring a consistent and fair assessment across different avatar generation platforms.
In contrast to MH, which are shown to the voter "wearing" the Procreate_AI facial texture, CC3 employs an automated texture generation approach. This process utilizes the original AI-generated photos of the subjects, produced by Stable Diffusion, to create the textures. Furthermore, CC3 avatars have a facial UV mapping that differs from the MH one, as shown in Figure 5.10. This automatic texture generation in CC3, starting directly from the subject's AI-generated images, contrasts sharply with the manual texture fixing approach used in MH.

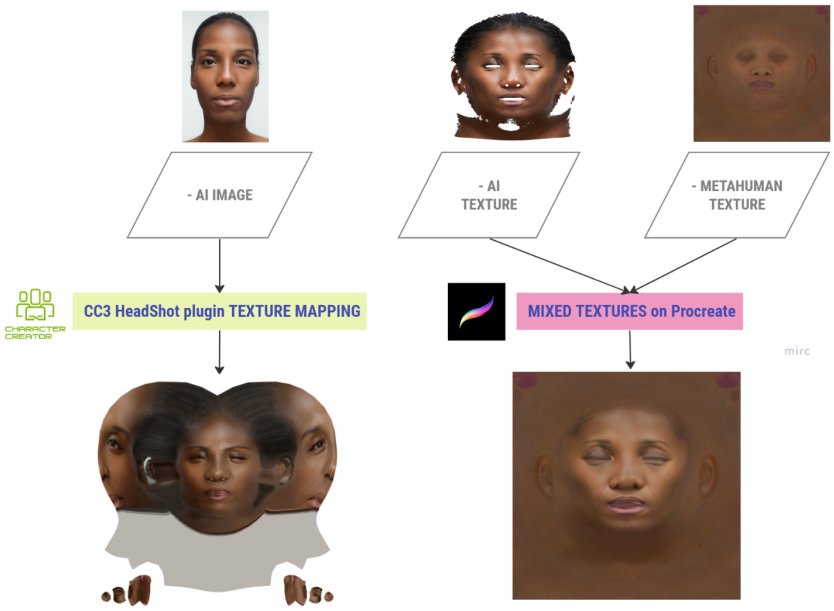


Figure 5.10: Displayed side-by-side, the distinct approaches to facial texture mapping: CC3's methodology is showcased on the left, while MH's technique is on the right

The results derived from the cosine similarity metric, shown in Table 5.6 and in figure 5.11, reveal that avatars generated using Character Creator often demonstrate comparable or higher values in comparison to those produced by MetaHuman.

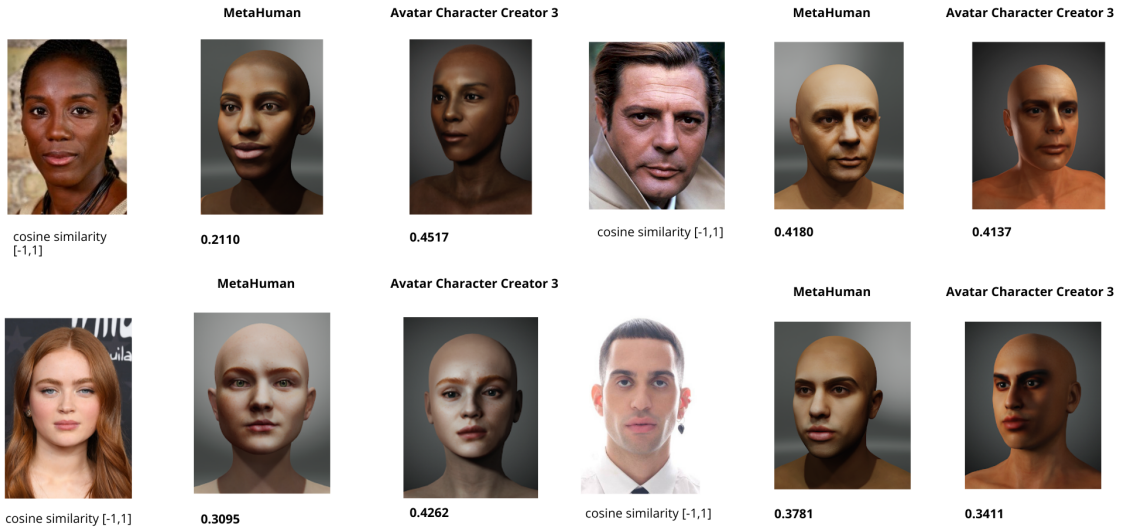


Figure 5.11: Some Results of the Cosine Similarity Metric Using ArcFace for MetaHumans and Avatars Obtained with Character Creator 3.

Subject	**MetaHuman**	**Character Creator 3 Avatar**
Alessandro Barbero	0.2083	0.2394
Monica Bellucci	0.3136	0.3817
Maria Callas	0.0957	0.2210
Fiona May	0.0211	0.4517
Marcello Mastroianni	0.4180	0.4137
Mahmood	0.3781	0.3411
Sadie Sink	0.3095	0.4262
Valerio Lundini	0.3480	0.4417
Zerocalcare	0.1945	0.5108
Mean	**0.2541**	**0.3808**

Table 5.6: Cosine Similarity Scores of the Final Synthetic Humans Mesh

Nevertheless, the conducted survey asking *"Which of the two Avatars most looks like the original subject?"* revealed mixed preferences among participants regarding Synthetic Humans presented as MetaHuman (MH) and Character Creator (CC3) avatars.
Out of 414 total responses (46 voters each choosing between two avatar types for 9 subjects), avatars created with Character Creator were preferred in approximately 57.73% (239 votes) of instances, while MetaHuman Creator was favored in 42.27% (175 votes). This preference distribution is indicating a inclination towards CC3's avatars, yet also

revealing a significant appreciation for MH's ones.

Despite the variability in user preferences, this information has enabled us to conduct a thorough comparison between the performance of Character Creator 3 and MetaHuman, marking a significant stride in the quest for more suitable solutions. This lays the groundwork for further investigation and enhancements in the Synthetic Human creation process, outlining avenues for the future evolution of this technology.

5.2.4 Results Considerations

The detailed analysis conducted in the Model Validation chapter has brought to light several key aspects regarding the effectiveness of our automated workflow in generating Synthetic Humans. Before examining the results in detail, it is essential to emphasize the importance of our generative network-based approach.

The use of generative networks for subject image generation has proven to play a significant role in improving the quality and similarity of Synthetic Humans compared to traditional methods. Generative networks allow capturing complex details and nuances present in the original images, producing realistic textures that maintain fidelity to the reference individual.

Briefly summarizing the main results obtained from different phases of the process:

1. **Validity of AI-Generated Images:**
   Images generated by generative networks have proven to be valid and realistic, with cosine similarity values consistently above 0.5, confirming their resemblance to the original counterparts. Survey results further supported the high fidelity of the generated images.

2. **Validity of 3D Meshes Obtained during Face Reconstruction:**
   3D meshes obtained during the face reconstruction phase using the script *"FB_Head_Reconstruction_Automatized.py"* maintained a high level of fidelity to the reference individual. Cosine similarity confirmed consistency between 3D meshes and original images.

3. **Validity of Synthetic Humans:**
   Cosine similarity metric results for Synthetic Humans showed a variety of values, with a tendency to have higher values in textures preserving more details of the original face. The survey highlighted a subjective preference for textures obtained through *Procreate_AI* and *Photoshop _AI*.

4. **Comparison between MetaHuman Creator and Character Creator 3:**
   Character Creator 3 has proven to be a promising alternative to MetaHuman Creator, with cosine similarity results often similar or superior. However, the survey did not reveal a clear user preference for either approach.

The generative network-based approach, therefore, offers greater realism in textures, preserving specific details of the original faces. However, it is important to note that some

challenges persist, especially during the 3D mesh-to-MetaHuman conversion phase within the Metahuman Creator workflow, potentially resulting in the loss of geometric facial information.

Although MetaHumans produced via semi-automated workflows are rapidly created, they have not yet attained the nuanced quality of 3D models of Synthetic Humans meticulously handcrafted or partially manually sculpted.
Nevertheless, this methodology establishes a reliable starting point for the development of 3D meshes and facial texturing.
While our automated methods provide a substantial starting point, manual refinement, through targeted sculpting of the 3D model and meticulous texture adjustments, can elevate the final product to an even higher standard of realism. The approach proposed in this thesis serves as a substitute for the more tedious manual tasks, streamlining the process significantly, especially valuable in crafting the subject's human head, a critical element for recognition.
These results provide a solid foundation for further research and developments in Synthetic Human creation.

Chapter 6

Conclusions

6.1 Summary and Final Considerations

This thesis, conducted in close collaboration with **RAI R&D**, represents a significant step towards the optimization and automation of the photorealistic avatar creation process in the context of the broadcast industry. The main objective was to explore and implement an advanced workflow for the generation of Synthetic Humans, aiming to minimize human intervention and ensure an efficient and flexible process.

The motivation behind this thesis was the challenge of revitalizing RAI's extensive archive of images and videos, transforming it into a valuable resource for the creation of photorealistic 3D avatars. The evolution of the broadcast industry demands increasingly engaging and interactive content, and the transformation of two-dimensional resources into photorealistic avatars of significant personalities, such as historical figures or past celebrities, offers an innovative approach to enhancing television programming with virtual duets, interviews, shows, and more.

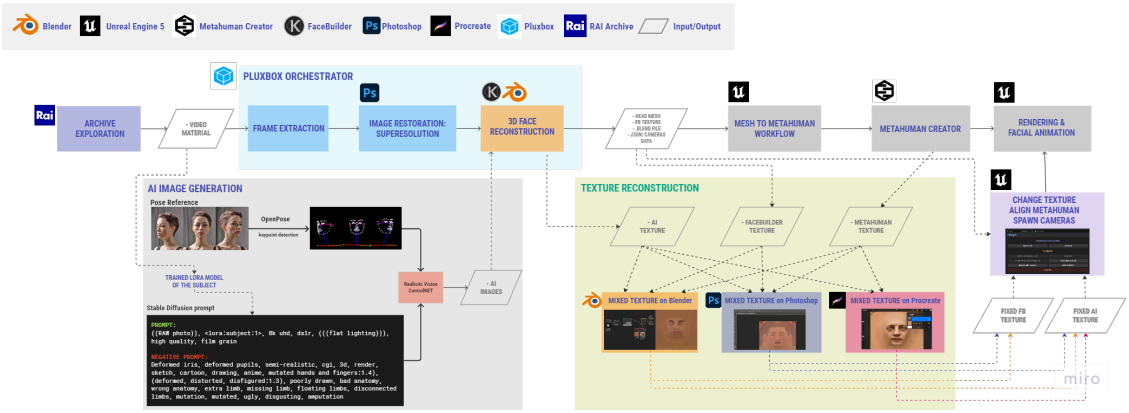


Figure 6.1: Implemented Workflow for Synthetic Human Creation

As shown in Figure 6.1, illustrating the implemented workflow in this thesis, the initial

phase focused primarily on the complete automation of the Face Reconstruction stage. This was made possible by the Orchestrator, developed in collaboration with PluxBox, a partner of RAI in the IBC2023 Accelerator project.

The Orchestrator, a sophisticated technological solution, played a central role in integrating various stages into a single block, from image selection to super resolution, and finally, three-dimensional face reconstruction. The automation of Face Reconstruction was facilitated by a script, made available to the Orchestrator through Rest APIs, utilizing the functionalities of the Blender FaceBuilder plugin to generate a 3D face mesh from 2D images.

Subsequently, emphasis was placed on improving textures through the adoption of Stable Diffusion to generate images of the reference subject with the help of Artificial Intelligence. The use of AI-generated images proved more suitable for texture production than traditional methods. This phase was influenced by the refinement of LoRA training models, enabling the generation of realistic and detailed images of the target subjects, crucial for creating uniform textures in the Synthetic Human generation process.

The in-depth analysis conducted in the Model Validation chapter aims to assess the effectiveness of various steps within the workflow.
Images generated through generative neural networks demonstrated significant similarity to their original counterparts, supported not only by cosine similarity metrics but also by survey results confirming the high fidelity of the generated images.
The fidelity of 3D meshes obtained during the automated Face Reconstruction phase remained at a high level, confirmed by both objective metrics and survey values.
However, the similarity comparison phase between the obtained MetaHuman and the original reference subject presented some critical issues.

Considering the previous data, it is evident that the use of generative networks for creating subject images allows for greater realism in textures, preserving specific details of the original faces. Moreover, the validation of the automation part of 3D mesh creation from images represents a significant step forward in the automation process of Synthetic Human creation. However, the studied process is not without challenges, as it was not possible to fully automate the entire workflow due to limitations in Metahuman APIs, even though the implementation of Widgets within Unreal Engine allowed for semi-automation of this phase. The main issue encountered relates to the conversion of 3D mesh to MetaHuman, where some cases experience a loss of facial geometry information.

Nevertheless, these results provide a starting point for further research, exploration, and developments in Synthetic Human creation.

6.2 Future Developments

Looking to the future, this thesis lays the groundwork for significant developments in the broadcast industry, marking a substantial shift in the approach to photorealistic avatar

creation. The automation of Synthetic Human generation is just the beginning of a potentially revolutionary transformation in audience interaction with visual content, paving the way for extensive integration into broadcasting, the metaverse, and video games contexts. However, to reach this stage, it is imperative to dedicate further efforts to in-depth studies, testing, and the exploration of new technologies, emphasizing the progressive nature and the early development path of this ambitious goal.

The following are potential directions for future developments, for further improvements in the Synthetic Human generation process and related automations. This analysis aims to identify key directions for subsequent research and development, outlining opportunities that can contribute to further consolidating and refining photorealistic avatar creation in the broadcast industry.

6.2.1 Optimization of the MetaHuman Conversion Process

A potential future development involves optimizing the conversion process from 3D mesh to MetaHuman. The current process, following the MetaHuman plugin workflow for Unreal Engine, known as ***Mesh To MetaHuman***, offers an intuitive solution, but variations in facial geometry may occur during the conversion to Metahuman. This variation is attributed to limitations in customization options offered by MetaHuman Creator during the conversion process.

Due to this issue, after executing the "Mesh to Metahuman" process, it is recommended to explore advanced customization options provided by MetaHuman Creator. This post-conversion step offers the possibility to further refine facial shape through sculpting tools and manual adjustment features within Metahuman Creator. Navigating through these advanced options can be crucial for adapting MetaHuman to specific aesthetic requirements, allowing for more detailed facial customization.

However, it is essential to note that if MetaHuman Creator has a limited range of customization options for facial shape compared to your original mesh, discrepancies in facial shape may still occur. In other words, MetaHuman Creator may not precisely replicate the face shape of your original mesh. The default nature of MetaHuman Identity implies that the variety of customizable facial shapes may be limited compared to the complexity of the original mesh.

Use of External 3D Sculpting Tools

In cases where MetaHuman Creator does not fully meet desired customization needs, an alternative approach is the use of external 3D sculpting tools. Through this methodology, the MetaHuman mesh can be exported and subsequently imported into platforms such as ZBrush, Blender, or Maya. These software, known for their advanced facial geometry manipulation capabilities, can offer an additional level of control over facial shape, allowing for refined modeling and more detailed adaptation to the original mesh concept.

A particularly relevant tool for Metahuman customization in Maya is **Metapipe (Custom Metahuman & Expressions Tool)** [55] [56].
Metapipe, integrated directly into Maya, provides 3D artists with a powerful tool for advanced Metahuman customization, positioning itself as a promising starting point for future developments aimed at overcoming limitations in the "Mesh to Metahuman" workflow of MetaHuman Creator, offering a level of control and flexibility beyond current constraints. It allows complete control over the mesh, an automated workflow, and the use of official Epic Games Calibration DNA Codes, significantly simplifying the customization process. Therefore, Metapipe appears as a potential future exploration, as it features characteristics that enable overcoming constraints in the standard "Mesh to Metahuman" process.

6.2.2 Exploration of Alternative Solutions to MetaHuman

Considering the challenges and limitations identified in the Metahuman workflow, such as the loss of geometric information during conversion, along with automation process restrictions due to the lack of APIs for remote use of Metahuman services, the exploration of alternative software for Photorealistic Avatar creation, such as **Character Creator 4** [36], could be evaluated.
In the latest update, Character Creator 4, augmented by a newly released Headshot plugin version, introduces a workflow similar to the 'Mesh to MetaHuman' process. Unlike the previous version, it directly uses the imported 3D mesh rather than solely depending on 2D image data. As demonstrated in Figure 2.12, Character Creator offers smooth integration with leading industry tools, enabling users to import characters seamlessly into their projects. Although Character Creator does not currently provide tools to complete automation, it could represent a fertile area for further developments, offering an alternative to the existing process.

This option is also motivated by the results in the Model Validation chapter, where both the survey and cosine similarity metric indicated values often superior between the original subject and the avatar generated by Character Creator 3 compared to those obtained with avatars generated by Metahuman. Exploring this direction could lead to significant improvements in Synthetic Human creation, reducing limitations observed in the current process.

6.2.3 Optimizations in the Use of Generative Networks

Optimizing generative networks could allow even more accurate rendering of textures, eliminating the need for manual post-generation modifications. Exploring new training models or advanced generation techniques could contribute to achieving increasingly convincing and faithful results to the originals.

In the pursuit of enhancing generative networks for faster texture rendering, in Figure 6.2 the **"Head Texture Map"** [57] LoRA model, presented on Civitai, offers a notable example. Based on Stable Diffusion 1.5, this model has been fine-tuned for the specific task of generating facial textures. Its size is only 144MB and uses trigger words such as

Head texture and *<lora:Head Texture Map_v.1.0:1>*.
The produced Facial UV map, is not yet compatible with MetaHuman's facial UV mapping, but it signifies a step towards more automated, AI-driven texture that may soon be tailored to project-specific requirements, potentially bypassing traditional tools like FaceBuilder for texture creation.

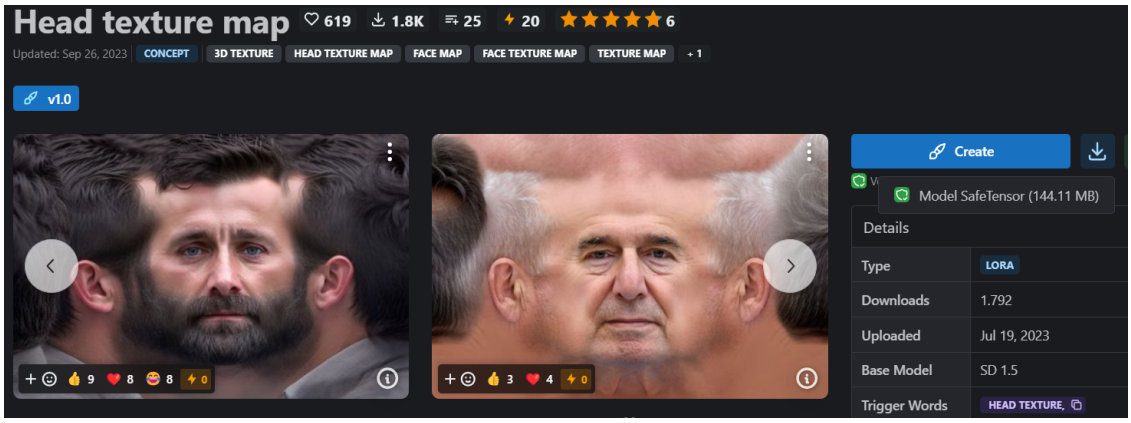


Figure 6.2: The Civitai web interface showcasing the 'Head Texture Map' model available for download

Another future development is the advent of **PanoHead** [58], a pioneering 3D generative model that synthesizes and reconstructs comprehensive 3D human heads from any viewpoint. PanoHead's innovative approach in using unstructured 2D images for training, while maintaining 3D consistency across broad viewing angles, signifies a substantial evolution beyond the capabilities of traditional 3D GANs.
This could revolutionize digital human modeling by creating detailed and diverse 3D heads, including intricate hairstyles, potentially bypassing the need for manual texturing and 3D sculpitng processes.

Integrating AI Texture Generation with the Orchestrator

The potential to integrate AI-driven texture generation into scripted workflows is a compelling research and development frontier. Pluxbox's orchestrator, renowned for its powerful integrations and open API architecture, could potentially facilitate seamless interactions with a local deployment of Stable Diffusion's API. The synergy between these platforms could significantly streamline the Synthetic Human production process, not only by enhancing the 3D head reconstruction process from raw video data but also integrating consistent facial texture. Another scenario could be to integrate AI texture generation directly inside Blender after the FaceBuilder phase. Stability AI provides a dedicated integration with Blender [59]. This allows users to directly apply AI-driven image generation and manipulation within Blender, offering capabilities such as Image-to-Image transformations on rendered frames or the creation of textures from textual descriptions.

Despite the analysis of these potential future developments, as we are still in the early stages of this project, aiming to create a fully automated and highly efficient workflow for broadcast industry adoption, it is essential to adopt a continuous evaluation and adaptation approach. This approach involves constant monitoring of technological developments and flexibility in modifying the workflow in response to new opportunities and challenges in the technological and industrial landscape.

In conclusion, this study not only establishes a base for future massive integration of Synthetic Humans in media and virtual contexts but also opens new perspectives for visual storytelling and interactive content experiences. Continuous commitment to research and development will fully leverage the potential of this technology, contributing to shaping the future of the broadcast industry.

Bibliography

[1] Human digital twins: Creating new value beyond the constraints of the real world. https://www.rd.ntt/e/ai/0004.html, 2023.

[2] An era of digital humans: Pushing the envelope of photorealistic digital character creation. https://developer.nvidia.com/blog/an-era-of-digital-humans-pushing-the-envelope-of-photorealistic-digital-character-creation/, 2021.

[3] Gillen McAllister. The story behind the last of us part 2's staggeringly realistic in-game character facial animation. https://blog.playstation.com/2020/08/28/the-story-behind-the-last-of-us-part-iis-staggeringly-realistic-in-game-character-facial-animation/, 2020.

[4] John Linneman. Detroit: Become human is a different kind of tech showcase. https://www.eurogamer.net/digitalfoundry-2018-detroit-become-human-tech-analysis, 2018.

[5] THE THIRD FLOOR, Inc. Virtual visualization series – the mandalorian. https://thethirdfloorinc.com/4206/virtual-visualization-series-the-mandalorian/, 2020. Online; accessed November 17, 2023.

[6] lilmiquela. Miquela's instagram profile. https://www.instagram.com/lilmiquela/, 2023. Instagram profile.

[7] Abba voyage official website - 2023 abba concert in london. https://abbavoyage.com/.

[8] Brian Caulfield. Nvidia, bmw blend reality, virtual worlds to demonstrate factory of the future. https://blogs.nvidia.com/blog/nvidia-bmw-factory-future/, 2021.

[9] Lex Fridman. Transcript for mark zuckerberg: First interview in the metaverse. https://lexfridman.com/mark-zuckerberg-3-transcript, 2023.

[10] Amelia Tait. 'here is the news. you can't stop us': Ai anchor zae-in grants us an interview. *The Guardian*, Oct 2023. URL https://www.theguardian.com/tv-and-radio/2023/oct/20/here-is-the-news-you-cant-stop-us-ai-anchor-zae-in-grants-us-an-interview.

[11] Digital human. https://www.alibabacloud.com/solutions/digital-human, 2023.

[12] Jake Bickerton. Digital human created for tour de france. *Broadcast Now*, Jun 2023. URL https://www.broadcastnow.co.uk/production/digital-human-created-for-tour-de-france/5183644.article.

[13] Digital humans. https://triplesense.it/digital-humans, 2023.

[14] Lu Chen, Sida Peng, and Xiaowei Zhou. Towards efficient and photorealistic 3d human reconstruction: A brief survey. *Visual Informatics*, 5(4):11–19, 2021. ISSN 2468-502X. doi: https://doi.org/10.1016/j.visinf.2021.10.003. URL https://www.sciencedirect.com/science/article/pii/S2468502X21000413.

[15] Adobe super resolution. Online, n.d. https://www.adobe.com/products/photoshop-lightroom/super-resolution.html.

[16] Topaz labs gigapixel ai. Online, n.d. https://www.topazlabs.com/gigapixel-ai.

[17] Zitong Wang, Jue Zhang, Ting Chen, Wen Wang, and Ping Luo. Restoreformer++: Towards real-world blind face restoration from undegraded key-value pairs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1–15, 2023. https://doi.org/10.1109/tpami.2023.3315753.

[18] Chan K. Li C. Zhou, S. and C. Loy. Towards robust blind face restoration with codebook lookup transformer. https://arxiv.org/pdf/2206.11253.pdf.

[19] Intel realsense. Online, n.d. https://www.intelrealsense.com/.

[20] Capturing reality. Online, n.d. https://www.capturingreality.com/.

[21] Skanect by structure. Online, n.d. https://structure.io/skanect.

[22] Facebuilder for blender | keentools. n.d. https://keentools.io/products/facebuilder-for-blender.

[23] KeenTools. Facebuilder for blender guide. 2021. https://medium.com/keentools/facebuilder-for-blender-guide-cbb10c717f7c.

[24] Reallusion character creator - headshot. Online, n.d. https://www.reallusion.com/character-creator/headshot/.

[25] Daz 3d - face transfer unlimited. Online, n.d. https://www.daz3d.com/face-transfer-unlimited.

[26] Avatar sdk. Online, n.d. https://avatarsdk.com/.

[27] Facegen. Online, n.d.. https://facegen.com/.

[28] Apostolos Lattas, Stylianos Moschoglou, Baris Gecer, Stylianos Ploumpis, Vassilis Triantafyllou, Aneesh Ghosh, and Stefanos Zafeiriou. Avatarme: Realistically renderable 3d facial reconstruction 'in-the-wild'. n.d. https://arxiv.org/pdf/2003.13845.pdf.

[29] Taras Martyniuk, Oleksandr Kupyn, Yurii Kurlyak, Ivan Krashenyi, Jiri Matas, and Viktoriia Sharmanska. Dad-3dheads: A large-scale dense, accurate and diverse dataset for 3d head alignment from a single image. n.d. https://arxiv.org/pdf/2204.03688.pdf.

[30] Baris Gecer, Jia Deng, and Stefanos Zafeiriou. Ostec: One-shot texture completion. n.d. https://arxiv.org/pdf/2012.15370.pdf.

[31] Reallusion iclone. Online, n.d. https://www.reallusion.com/iclone/.

[32] Nvidia omniverse - audio2face. Online, n.d. https://www.nvidia.com/it-it/omniverse/apps/audio2face/.

[33] Facewaretech. Online, n.d.. https://facewaretech.com/.

[34] Unreal engine - metahuman. Online, n.d. https://metahuman.unrealengine.com/.

[35] New metahuman animator feature set to bring easy high-fidelity performance capture to metahumans. n.d. https://www.unrealengine.com/en-US/blog/new-met.

[36] Character creator. Online, n.d. https://www.reallusion.com/character-creator/.

[37] Ibc2023 accelerator project: Synthetic humans. Online, 2023. https://www.ibc.org/features/ibc2023-accelerator-project-synthetic-humans/9944.article.

[38] Synthetic humans in the metaverse. Online, 2023. https://show.ibc.org/accelerator-media-innovation-programme/synthetic-humans-metaverse.

[39] Pluxbox. Online, n.d.. https://pluxbox.com/.

[40] KeenTools. Keentools facebuilder x metahuman guide. Medium, n.d. https://medium.com/keentools/keentools-facebuilder-x-metahuman-guide-81bc193ef2a.

[41] Unreal engine - metahuman. Online, n.d. https://www.unrealengine.com/en-US/metahuman.

[42] Metahuman plugin. Online, n.d.. https://www.unrealengine.com/marketplace/en-US/product/metahuman-plugin.

[43] Mesh to metahuman quick start in unreal engine. Online, n.d. https://dev.epicgames.com/documentation/en-US/metahuman/mesh-to-metahuman-quick-start-in-unreal-engine.

[44] Unreal Engine. Delivering high-quality facial animation in minutes: Metahuman animator is now available. Online, n.d. https://www.unrealengine.com/en-US/blog/delivering-high-quality-facial-animation-in-minutes-metahuman-animator-is-now-available.

[45] CompVis. Stable diffusion. GitHub repository, n.d. https://github.com/CompVis/stable-diffusion.

[46] L. Zhang and M. Agrawala. Adding conditional control to text-to-image diffusion models. Online, 2023. https://arxiv.org/pdf/2302.05543.pdf.

[47] CMU-Perceptual-Computing-Lab. Openpose. GitHub repository, n.d. https://github.com/CMU-Perceptual-Computing-Lab/openpose.

[48] E. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. Online, n.d. https://arxiv.org/pdf/2106.09685.pdf.

[49] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. Online, 2022. https://arxiv.org/pdf/2112.10752.pdf.

[50] CIVITAI (se disponibile). Realistic vision v12. Online, n.d. https://civitai.com/models/4201/realistic-vision-v12.

[51] X. Wei, H. Wang, B. Scotney, and H. Wan. Cosface: Large margin cosine loss for deep face recognition. *Pattern Recognition*, 97:107012, 2020. doi: 10.1016/j.patcog.2019.107012.

[52] Jing Yang Niannan Xue Irene Kotsia Jiankang Deng, Jia Guo and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. *Journal of LaTeX Class and Files*, 14(8), 2015. https://arxiv.org/pdf/1801.07698v4.pdf.

[53] InsightFace Developers. InsightFace: 2d and 3d face analysis project. https://github.com/deepinsight/insightface, 2023. Accessed: 2023-03-20.

[54] Sefik Ilkin Serengil. Deep face recognition with arcface in keras and python. https://sefiks.com/2020/12/14/deep-face-recognition-with-arcface-in-keras-and-python/, 2020.

[55] Road to custom metahuman: Metapipe custom metahuman expressions tool. ArtStation Marketplace, n.d. https://www.artstation.com/marketplace/p/PmpWr/road-to-custom-metahuman-metapipe-custom-metahuman-expressions-tool.

[56] Arts and spells documentation. Online, n.d. https://www.artsandspells.com/documentation.

[57] Head texture map - v1.0 | stable diffusion lora. https://civitai.com/models/112287/head-texture-map.

[58] Sizhe An, Hongyi Xu, Yichun Shi, Guoxian Song, Umit Y. Ogras, and Linjie Luo. Panohead: Geometry-aware 3d full-head synthesis in 360deg. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2023.

[59] Stability for blender. https://platform.stability.ai/docs/integrations/blender.