Journal of the Brazilian Computer Society, 2026, 32:1, doi: 10.5753/jbcs.2026.8232

FacePipe: A Low-Cost Solution for 3D Facial Animation

Artur Tavares de Carvalho Cruz [**Universidade Federal de Pernambuco** | ***artur.cruz@ufpe.br***]
Calíope Corrêa de Araújo [**Universidade Federal de Pernambuco** | ***caliope.araujo@ufpe.br***]
Willams de Lima Costa [**Voxar Labs, Centro de Informática, UFPE** | ***wlc2@cin.ufpe.br***]
Fabiana Frata Furlan Peres [**Universidade Estadual do Oeste do Paraná** | ***fabiana.peres@unioeste.br***]
Walter F M Correia [**Universidade Federal de Pernambuco** | ***walter.franklin@ufpe.br***]
Alexandre Siqueira [**University of Florida**| ***agomesdesiqueira@ufl.edu***]
Fatima L S Nunes [**Universidade de São Paulo** | ***fatima.nunes@usp.br***]
João Marcelo Teixeira [**Universidade de São Paulo, Universidade Federal de Pernambuco**| ***jmxnt@cin.ufpe.br***]

Center of Informatics, Universidade Federal de Pernambuco, Av. Jornalista Aníbal Fernandes, s/n, Cidade Universitária, Recife, PE, 50740-560, Brazil.

Received: 16 May 2026 • **Accepted:** 05 August 2026 • **Published:** 13 August 2026

Abstract. This paper presents FacePipe, a low-cost 3D facial capture and animation solution based on a conventional RGB camera and integrated with Blender. The system was developed from design requirements derived from an exploratory review, workflow analysis, and technology-selection criteria focused on low-cost hardware availability, interoperability, usability, openness, and maintainability. FacePipe consists of a MediaPipe-based capture module, a structured local data-management and export workflow, and a dedicated Blender add-on that converts captured blendshape coefficients into editable animation data. The solution was evaluated through a Hierarchical Task Analysis comparing FacePipe, FaceCap, and a MetaHuman-based workflow, and through a between-subjects usability study comparing FacePipe and FaceCap with an adapted UEQ+ questionnaire. The evaluation addresses workflow structure and perceived user experience, but it does not establish objective tracking-accuracy equivalence between FacePipe and FaceCap. Results indicate that both tools were evaluated positively, but with distinct workflow profiles. FaceCap was rated more favorably in setup, perceived dependability, and Blender integration, while FacePipe showed a descriptive advantage in file export and organization and comparable perceived usability during recording. These findings suggest that FacePipe is a viable low-cost alternative for editable facial animation workflows, especially in contexts that value widely available RGB-camera-based capture, structured local file management, open implementation, and integration with Blender.

Keywords: Facial animation, Facial motion capture, Usability.

1 Introduction

Digital animation is a profitable entertainment industry, but it is also characterized by long, complex, and resource-intensive production processes. According to Pixar animator Bruce Kuei, animated feature films may take between four and seven years to be completed [Hausman, 2019]. The production of animated content involves extensive workflows, including modeling, rigging, animation, rendering, and multiple review stages, requiring significant technical, computational, and financial resources [Beane, 2012; Tschang and Goldstein, 2004; Mitchell, 2017]. These requirements create barriers for independent creators, small studios, and educational contexts, particularly in countries with limited access to specialized infrastructure and high-end equipment [Gatti Jr. *et al*., 2014].

In recent years, audiovisual production has increasingly incorporated generative artificial intelligence systems capable of producing videos from text, images, and multimodal inputs. Platforms such as Veo/Flow and Kling demonstrate advances in controllable video generation through mechanisms involving camera control, motion conditioning, keyframes, and visual references [Google, 2025; Kling AI, 2025]. Current research has progressively shifted from purely text-driven generation toward controllable systems that integrate structured signals such as pose, depth, and motion [Ma *et al*., 2025].

However, in the context of 3D facial animation, generating visually plausible videos alone is insufficient. Production pipelines require editable and reusable facial representations compatible with rigs, blendshapes, and retargeting systems. Professional facial animation workflows still depend on complex processes involving geometry generation, rigging, and motion transfer [Zhang *et al*., 2025]. In addition, facial retargeting remains strongly dependent on semantic consistency and parametric facial structures [Zhu and Joslin, 2024a,b].

In this context, low-cost and widely available facial capture solutions remain relevant despite recent advances in generative AI. Rather than replacing animation pipelines, these technologies reinforce the importance of systems capable of producing controllable and interoperable facial data for 3D animation workflows. Therefore, this paper presents FacePipe, a low-cost facial capture tool based on conventional RGB cameras and integrated with Blender, designed to support editable facial animation workflows without relying on proprietary capture hardware or dedicated depth sensors.

The theoretical foundation of this work is structured around three complementary areas: Computer Science, Design, and Psychology. Computer Science contributes concepts from computer graphics, computer vision, and machine learning,

including facial tracking, 3D reconstruction, and parametric animation methods [Parke, 1972; Hernandez *et al*., 2012]. Design contributes methods related to digital artifact development, prototyping, and usability evaluation [Baxter, 1995]. Finally, Psychology contributes the Facial Action Coding System (FACS), which provides a semantic framework for describing facial expressions and supports the parametrization of facial animation systems [Ekman and Friesen, 1978].

The relevance of this research lies in addressing facial animation not only as a technical problem of motion capture and visual synthesis, but also as a digital artifact design problem embedded within production pipelines. While many existing studies emphasize geometric precision or visual realism, less attention has been given to workflow integration, usability, and economic availability in independent, educational, and low-infrastructure contexts. In this sense, the present work contributes by investigating how a low-cost facial capture system can be designed and evaluated within an editable facial animation workflow while preserving interoperability and parametric control.

Thus, this paper focuses on the development and evaluation of FacePipe as a low-cost facial animation pipeline integrated with Blender. The research combines three complementary dimensions: the use of an exploratory review and workflow analysis to support design requirements, the development of a low-cost RGB-camera-based facial capture solution, and the evaluation of the proposed system through workflow and usability analysis. The literature review is used as contextual and design-supporting evidence rather than as a systematic-review contribution. More specifically, the study aims to:

1. Define technical and functional requirements for a low-cost facial capture system based on an exploratory review and workflow analysis;
2. Develop a functional prototype integrated into Blender workflows; and
3. Evaluate its usability and workflow characteristics in comparison with existing solutions.

Because the study focuses on workflow and user experience, it does not compare the geometric or temporal tracking accuracy of FacePipe and FaceCap. Claims about tool comparison are therefore limited to usability, perceived workflow quality, and integration effort.

The remainder of this paper is structured as follows. Section 2 presents the theoretical background, discussing facial animation, parametric facial representation, markerless capture, and recent advances in neural and blendshape-based approaches. Section 3 describes the methodological procedures adopted in the study, including the design of FacePipe, its system architecture, facial capture pipeline, Blender integration, task analysis, and usability evaluation protocol. Section 4 presents the HTA and usability results. Section 5 discusses the workflow trade-offs, scope of comparison, hardware variability, and limitations. Section 6 concludes the paper.

2 Theoretical Background

3D facial animation involves the representation, parametrization and synthesis of human facial movement for digital characters and virtual avatars. Because facial expressions are directly associated with emotion, communication and perception of realism, facial animation systems must balance geometric accuracy, expressive control and computational efficiency. Early approaches include muscle-based parametrization methods proposed by Parke [1974] and physically based facial models developed by Terzopoulos and Waters [1990]. Since then, the field has expanded through techniques involving facial motion capture, audio-driven animation, blendshape systems and machine learning approaches [Lewis, 1991; Sturman, 1994; Pighin and Lewis, 2006].

Contemporary facial animation pipelines are commonly structured around stages such as modeling, rigging, motion acquisition and deformation transfer [Kerlow, 2009; Ping *et al*., 2013]. Within these pipelines, rigs operate as intermediary control structures between motion data and facial geometry, frequently relying on blendshapes, bones or hybrid systems [Orvalho *et al*., 2012; de Carvalho Cruz and Xavier Natario Teixeira, 2021]. Blendshape-based rigs remain particularly relevant due to their compatibility with animation software and real-time applications, allowing facial expressions to be represented through interpolated parametric targets derived from a shared topology [Parke, 1982; Villar, 2014].

A central theoretical reference for semantic facial representation is FACS, proposed by Ekman and Friesen [1978]. FACS organizes facial expressions through Action Units (AUs), describing observable muscular movements that can be mapped to facial rigs and animation controllers. Because of its semantic interpretability and modular structure, FACS became widely adopted in facial animation, facial capture and performance transfer systems [Parke and Waters, 2008]. More recent revisions, such as the F-M FACS proposed by Freitas-Magalhães [2018], expanded the framework to include head, eye and tongue movements, reinforcing its applicability to digital facial performance.

Recent advances in computer vision and machine learning significantly transformed facial animation research. Markerless facial tracking based on facial landmarks and monocular RGB cameras enabled lower-cost capture pipelines compared to traditional marker-based motion capture systems [Baltrušaitis *et al*., 2018]. At the same time, deep learning approaches introduced new possibilities for facial reconstruction, performance transfer and neural rendering.

A complementary line of work investigates speech- and audio-driven facial animation. These approaches are not direct substitutes for FacePipe, because they typically synthesize facial motion from speech rather than capturing RGB-based facial performance for subsequent Blender editing. However, they are important for positioning the proposed pipeline within the broader facial animation ecosystem. Representative examples include audio-driven animation with learned pose and emotion control, generalized speech animation, implicit emotional awareness, animator-centric viseme control, learned speaking styles, cross-modal speech-to-mesh animation, transformer-based speech-driven animation, and discrete motion-prior approaches [Karras *et al*., 2017; Taylor

et al., 2017; Pham *et al*., 2017; Zhou *et al*., 2018; Cudeiro *et al*., 2019; Richard *et al*., 2021; Fan *et al*., 2022; Xing *et al*., 2023; Chhatre *et al*., 2025]. Together, these works show that neural speech-driven animation has advanced substantially, but also reinforce the importance of editable intermediate representations, such as mesh vertices, blendshapes, or animation curves, when the goal is integration with production pipelines.

The background review used in this work should be understood as an exploratory and design-oriented review, not as a systematic literature review. Its purpose was to identify representative technical approaches, design constraints, and interoperability issues relevant to the development of FacePipe. The search combined peer-reviewed sources with a limited set of preprint sources used only to identify emerging trends. When preprints were considered, they were treated as non-peer-reviewed evidence and were not used as the basis for claims of consolidated scientific consensus.

Current research can be broadly divided into two major groups. The first consists of parametric approaches based on FACS, blendshapes and morphable face models, prioritizing editability, semantic control and compatibility with established workflows. In this context, FACS remains a central reference for facial representation and expression parametrization.

Among the most influential parametric models are FLAME [Li *et al*., 2017], DECA [Feng *et al*., 2021], EMOCA [Danecek *et al*., 2022] and FaceVerse [Wang *et al*., 2022]. These methods use statistical and parametric facial representations capable of reconstructing identity, pose and expression while preserving compatibility with animation rigs and blendshape-based systems. DECA and EMOCA further extend these approaches by incorporating fine facial details and emotionally consistent expressions.

The second major group involves neural implicit representations, frequently inspired by Neural Radiance Fields (NeRFs) and Gaussian Splatting techniques. Methods such as AVFace [Chatziagapi and Samaras, 2023], ImFace++ [Zheng *et al*., 2025] and 3D Gaussian Blendshapes [Ma *et al*., 2024] prioritize visual realism and high-fidelity reconstruction using neural fields and volumetric rendering. Although these approaches achieve impressive visual quality, they often reduce direct editability and semantic transparency when compared to traditional parametric systems.

Recent literature also indicates a convergence between these paradigms. Many contemporary systems combine interpretable parametric facial controls with neural refinement layers, preserving compatibility with existing animation workflows while improving realism and reconstruction quality. In this context, editable parameters such as blendshape coefficients and morphable model representations continue to function as interoperable components across different stages of facial animation pipelines [Li *et al*., 2017; Wang *et al*., 2022].

In parallel with academic research, commercial solutions such as Apple ARKit, Epic Games MetaHuman, Faceware and NVIDIA Audio2Face have significantly influenced contemporary production workflows. These systems provide real-time facial capture, retargeting and integration with animation software, but frequently depend on proprietary ecosystems, specialized hardware or high computational requirements.

Overall, the literature demonstrates that current facial animation pipelines balance two competing demands: visual fidelity and parametric control. While neural methods improve realism, parametric approaches remain essential for interoperability, editability and integration into production workflows. This context reinforces the relevance of low-cost and open-source solutions such as FacePipe, which prioritize widely available facial capture and reusable parametric outputs compatible with Blender-based animation pipelines.

3 Method

This study adopted a design-based, comparative, and experimental methodology focused on the development and evaluation of a low-cost facial animation pipeline. The method was organized into three complementary phases.

First, a literature review and parametric analysis of 3D facial capture and animation techniques were conducted to identify technical requirements, interoperability constraints, and limitations of existing solutions. These findings supported the definition of design guidelines and technology-selection criteria for the prototype, following principles of product design, interaction design, and usability evaluation [Baxter, 1995].

Second, the resulting prototype, named FacePipe, was designed as an RGB camera-based facial capture solution compatible with ARKit-inspired blendshape structures and integrated with Blender. The design process translated the requirements identified in the review into a functional pipeline for capture, data organization, export, and animation reuse.

Third, the prototype was evaluated through two complementary procedures. A Hierarchical Task Analysis (HTA) compared FacePipe with FaceCap and a MetaHuman-based workflow, focusing on the tasks and operations required to transform facial performance capture into reusable facial animation assets. In addition, a between-subjects usability study was conducted in which participants used either FacePipe or FaceCap to perform facial capture and animation tasks. User experience was assessed through a questionnaire adapted from the UEQ+ framework [Schrepp *et al*., 2017], and the results were interpreted together with the HTA findings to relate workflow structure to perceived usability.

3.1 FacePipe Design

The prototype developed in this research was guided by User-Centered Design (UCD) principles. In this work, UCD is treated as a methodological basis for aligning the system with users' tasks, constraints, and production contexts. Following Mao *et al*. [2001], the design process considered user needs, task structure, iterative development, and usability evaluation as central elements for interactive system development.

The design process also incorporated Norman's distinction between visceral, behavioral, and reflective levels of user experience [Norman, 2004]. This perspective supported attention to both immediate interface perception and the practical experience of controlling, recording, organizing, and reusing facial animation data. In addition, the iterative lifecycle proposed by Benyon [2013], involving context analysis,

requirements definition, design, and evaluation, informed the organization of the prototype development process, as shown in Figure 1.

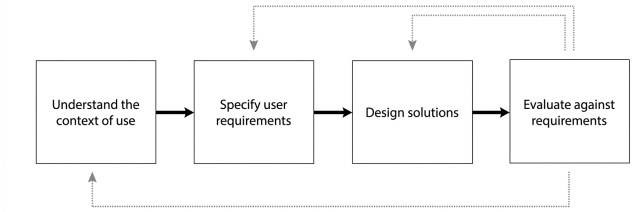

Figure 1. Stages of the User-Centered Design process defined by Benyon [2013]

FacePipe was developed as an end-to-end system for real-time facial animation capture and processing. Rather than being conceived as a set of isolated components, the system was designed as a continuous workflow that transforms live facial input into structured animation data compatible with blendshape-based pipelines.

The prototype operates through four main stages: facial data acquisition using a standard RGB camera, real-time facial analysis, structured data generation, and export for use in 3D animation software. This organization reflects the central design concerns of the project: reducing dependency on specialized hardware, minimizing manual file-management operations, and preserving compatibility with established facial animation workflows.

In this sense, FacePipe functions as a bridge between low-cost facial capture and editable 3D animation workflows. Its purpose is not to replace high-end capture systems, but to provide a widely available low-cost pipeline for independent creators, educational contexts, and small production environments that require reusable facial animation data.

The following sections describe the design guidelines, technology-selection criteria, system architecture, capture process, data format, and Blender integration implemented in the prototype.

3.1.1 Design Guidelines

Based on the theoretical foundations and the parametric analysis of the state of the art, a unified set of design guidelines and technology-selection criteria was defined to guide the development of FacePipe. These elements translated high-level design goals into operational decisions, ensuring feasibility, usability, and long-term sustainability.

The main design guidelines were defined as follows:

- **Hardware and economic availability**: the system should operate with low-cost and widely available hardware, avoiding dependency on specialized capture devices and lowering the entry barrier for independent users, small studios, and educational contexts.
- **Interoperability**: the generated animation data should follow facial animation conventions compatible with established workflows, particularly ARKit-inspired blendshape structures, supporting reuse in tools such as Blender, Unity, and Unreal Engine.
- **Usability**: the workflow should reduce cognitive load, provide predictable interaction, and minimize repetitive or error-prone tasks, especially during capture organization, export, and integration with 3D software.
- **Openness**: the software stack, source code, data format, and Blender integration should be transparent and modifiable, supporting academic inspection, adaptation, reuse, and future extension.
- **Sustainability**: the implementation should favor maintainable and evolvable technologies, with active documentation, modular architecture, and long-term development potential.

The technology-selection criteria derived from these guidelines were:

- **RGB camera input**: use of standard webcams or smartphone cameras, eliminating the need for specialized sensors.
- **Blendshape compatibility**: adherence to ARKit/FACS-based structures to support interoperability with existing facial animation pipelines.
- **Open-source infrastructure**: preference for tools with permissive licenses, active documentation, and strong community support.
- **Maintainability**: selection of technologies with active development and modular architecture to facilitate future updates.

Alternative solutions such as FaceVerse v4, NVIDIA Audio2Face, and Mocap4Face were analyzed but not adopted due to limitations related to proprietary ecosystems, hardware dependencies, incomplete alignment with the target blendshape workflow, or discontinued development.

Given these constraints, Google's MediaPipe was selected as the core facial tracking solution. Its face mesh and face landmarking modules enable real-time facial tracking and blendshape-like coefficient estimation from standard RGB cameras, providing a low-cost, cross-platform, and well-documented foundation for implementation and future extensibility.

3.1.2 System Architecture and Data Pipeline

FacePipe is a cross-platform application developed using Electron, integrating web technologies (HTML, CSS, and JavaScript) with Node.js to provide a real-time facial capture interface based on MediaPipe. Its architecture was designed to support the capture, organization, visualization, and export of facial animation data through a structured workflow. A high-level overview of the system is presented in Figure 2.

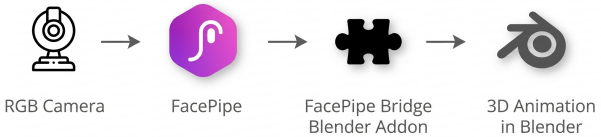

Figure 2. High-level overview of the FacePipe system workflow.

When launched, FacePipe automatically creates a root directory named *FacePipe* inside the user's Documents folder. Captures are organized into sequentially numbered scene folders (e.g., Scene 1, Scene 2), which function as groupings for

related recordings. Each recording generates a corresponding `.json` file stored in the active scene directory, enabling users to manage multiple captures without manually creating or organizing folders. Through the interface, users can rename files and scene folders, create new scenes, delete data, and access directories directly in the system file explorer.

During capture, the system continuously estimates the camera frame rate (FPS), which may vary depending on hardware and recording conditions. The average FPS is computed in real time and stored with the capture data. This value is later used by the FacePipe Bridge add-on to synchronize the imported animation in Blender according to the original recording rate, preserving temporal consistency between capture and playback.

Each recording session results in a `.json` file containing 52 blendshape coefficients per frame, stored as floating-point values between 0.0 and 1.0. Early versions of FacePipe used a CSV-based format, where each row represented a frame and FPS values were redundantly repeated. Although functional, this structure lacked semantic clarity and introduced unnecessary redundancy.

The current implementation uses the JSON (JavaScript Object Notation) format for its structured, self-describing nature. Each file contains three main fields: `fps`, which stores the average frame rate of the recording; `blendshapes`, which defines the ordered list of facial coefficients; and `frames`, which stores sequential arrays of blendshape values for each captured frame. This structure eliminates redundant FPS entries, improves readability, enables direct mapping to Blender shape keys, and preserves extensibility for future metadata without breaking compatibility with the pipeline.

3.1.3 Facial Capture and Processing Technologies

FacePipe is built on the MediaPipe framework, a collection of machine learning libraries developed by Google to support real-time perception pipelines across web, mobile, and Python environments. Its modular architecture enables efficient deployment of vision-based solutions with low computational cost, making it suitable for real-time facial capture applications.

The system is based on the MediaPipe Face Landmarker task, which integrates two core components: the FaceMesh and Blendshape models. Together, these components form a two-stage pipeline that extracts 3D facial geometry and converts it into semantic facial expression parameters compatible with the ARKit convention.

As illustrated in Figure 3, the pipeline begins with the FaceMesh model, which processes RGB images or video frames and outputs a dense set of 478 3D facial landmarks. This model is implemented using a convolutional neural network optimized for real-time performance. Before inference, each frame undergoes a preprocessing step in which the facial region is detected, cropped with a 25% margin, and normalized to a fixed resolution of 256×256 pixels. This standardization improves numerical stability and reduces computational cost while maintaining robustness under moderate variations in pose and illumination [LeCun *et al.*, 2015; Sze *et al.*, 2017].

The Face Landmarker operates through two inference modes. When a face has already been detected, the system uses the previous frame to guide tracking and reduce computational cost. When the face is lost, occluded, or appears for the first time, the system activates a BlazeFace-based reacquisition stage to relocate the face before continuing landmark estimation. This tracking acquisition behavior is relevant for real-time capture because it affects how the system responds to interruptions, partial occlusions, and changes in head position.

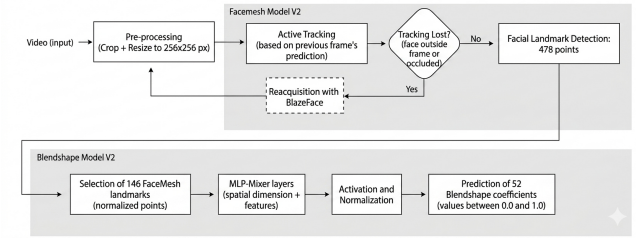

Figure 3. Processing pipeline of MediaPipe FaceMesh and Blendshape models.

The landmarks generated by FaceMesh are then passed to the Blendshape model, which estimates 52 continuous facial expression coefficients. This second stage is implemented using an efficient MLP-Mixer architecture [Tolstikhin *et al.*, 2021], which processes a reduced subset of 146 landmarks derived from the original 478-point mesh. The output values represent semantically meaningful facial actions, such as mouth opening, blinking, cheek puffing, and other expressive deformations, with intensities normalized to 0.0-1.0.

This separation between geometric reconstruction, performed by FaceMesh, and semantic expression estimation, performed by the Blendshape model, enables a modular and reusable pipeline. The same landmark representation can support multiple downstream tasks, including animation, avatar control, and real-time interaction systems. In FacePipe, this design allows the system to maintain a lightweight architecture while producing animation-ready facial coefficients suitable for low-cost workflows, including VTubing and real-time character animation.

Overall, the pipeline shown in Figure 3 illustrates how raw RGB input is progressively transformed into structured geometric data and then into blendshape coefficients, forming the basis of the facial capture system used in this work.

3.1.4 Desktop Application and User Interface

The FacePipe desktop application was developed using Electron, a cross-platform framework that combines web technologies such as JavaScript, HTML, and CSS with operating system resources through Node.js and the Chromium rendering engine. This architecture allows the same codebase to be packaged for Windows, macOS, and Linux without requiring separate platform-specific implementations.

The adoption of Electron was relevant for three main reasons.

First, FacePipe can be installed and run locally as a desktop application, enabling offline use without a constant internet connection. This characteristic is relevant for independent creators, educational settings, and small studios working in environments with limited connectivity.

Second, Electron provides access to the user's file system through Node.js APIs, enabling FacePipe to automatically cre-

ate organized folder structures for recordings, manage scene files, generate and export `.json` files containing blendshape data, and provide direct access to exported captures through the operating system's file explorer.

Third, the distribution process was simplified through Electron Forge, which automates the generation of installation packages for multiple operating systems, allowing end users to install the software without configuring dependencies or development environments.

The graphical user interface was initially planned using low-fidelity wireframes in Figma, enabling rapid iteration on layout organization, visual hierarchy, and navigation flow before implementation. The final interface was implemented in HTML and CSS using the Bootstrap framework, which provides a responsive grid system based on flexbox and breakpoints. This approach follows responsive design principles discussed by Marcotte [2011], enabling the interface to reorganize according to screen size while preserving readability and usability.

Two responsive layout configurations were considered during development: one optimized for widescreen monitors, displaying blendshape values beside the capture video, and another designed for narrower displays, positioning the blendshape panel below the video feed. By leveraging Bootstrap's responsive grid system, the application adapts to different resolutions and window sizes while maintaining interface consistency and real-time visual feedback. The final implementation of FacePipe running as an Electron desktop application is shown in Figure 4.

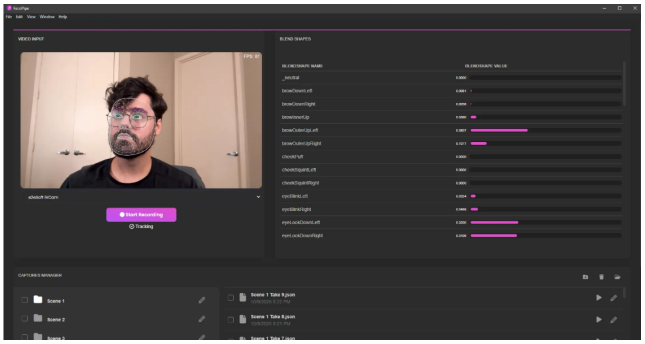

Figure 4. FacePipe implemented in Electron and running on Windows 11.

3.1.5 Blender Integration and Add-on Development

To enable the facial capture data generated by FacePipe to be applied to three-dimensional characters, a dedicated add-on for Blender was developed, named *FacePipe Bridge*. Blender is an open-source 3D modeling and animation software distributed under the GNU General Public License (GPL), whose Python-based API enables automation, customization, and integration with external tools. This extensibility makes Blender suitable for interoperability within computer graphics pipelines and animation workflows.

The FacePipe Bridge add-on acts as an integration layer between the prototype and Blender by introducing a custom `.json` import option into the *File* > *Import* menu. When a FacePipe `.json` file is selected, the add-on reads the stored blendshape list, frame rate, and animation frame matrix, converting the captured data into shape key animation inside Blender. The system creates or maps blendshapes to compatible mesh objects following an ARKit-inspired naming convention, including side-specific naming adjustments when necessary, and inserts keyframes for each blendshape value along the animation timeline.

The resulting animations are stored as Blender Actions named according to the source file and capture FPS, facilitating reuse across different scenes and characters. Since the animation is converted into native Blender keyframes and animation curves, users can further refine expressions, smooth transitions, and edit timing directly in the Graph Editor and keyframe editor, as illustrated in Figure 5.

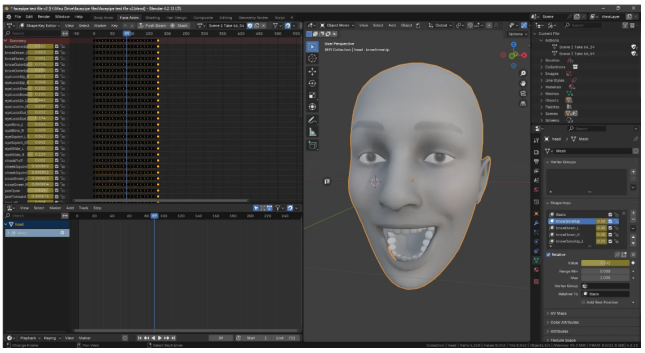

Figure 5. Manipulation of imported animation keyframes in FacePipe Bridge.

Internally, the add-on was implemented in Python using Blender's API architecture for extensions and operators. The import operator uses Blender's `ImportHelper` utility to open a file-selection dialog and filter compatible `.json` files. The code is organized into modular functions responsible for reading and preparing capture data, extracting the source filename, formatting the FPS value, creating and mapping shape keys, generating Actions, inserting animation keyframes, adapting animations to the scene frame rate, and registering or removing the import command from Blender's interface. A dedicated temporal baking function recalculates keyframe positions according to the ratio between the capture FPS and the scene FPS, preserving the original blendshape intensities while maintaining temporal synchronization. Table 1 summarizes the main functions and operators implemented in the add-on.

A supplementary tutorial[1] and video[2] demonstrate the complete FacePipe workflow, including application launch, capture setup, facial recording, JSON export, Blender import through FacePipe Bridge, Action generation, and keyframe inspection. The video clarifies the sequence of operations and the distinction between runtime capture and post-capture Blender animation import.

3.2 Hierarchical Task Analysis Procedure

A Hierarchical Task Analysis (HTA) was conducted to compare the operational structure of three facial animation workflows: FacePipe, FaceCap, and a MetaHuman/Unreal Engine workflow integrated with Blender through the MetaHuman

[1] `https://sircruxstudios.notion.site/FacePipe-Windows-and-Mac-OS-Installation-28b0e370f6b9801aba62f72c190`

[2] `https://drive.google.com/file/d/1ZtLOddNUtv63xS8MF5dm0oZ6SEYUwxYC/view?usp=sharing`

Table 1. Main functions and operators of the FacePipe Bridge add-on.

Component	Description
`read_json_file()`	Reads FacePipe `.json` capture files and returns structured data, including blendshapes, FPS, and animation frames.
`get_json_filename()`	Extracts the base filename used to name the resulting Blender Actions.
`format_fps()`	Formats the captured FPS value for Action naming and identification.
`create_shape_keys()`	Creates and maps blendshapes to Blender shape keys, ensuring compatibility with ARKit-inspired naming conventions.
`create_shape_key_action()`	Generates Blender Actions from capture data and inserts keyframes for each blendshape across the animation timeline.
`bake_action_to_scene_fps()`	Adjusts animation timing to match the Blender scene FPS by recalculating keyframe positions while preserving blendshape intensities.
`JSON_IMPORT_OT_shape_key`	Operator responsible for the import workflow, including file selection, data parsing, shape key creation, Action generation, and temporal baking.
`menu_func_import()`	Adds the FacePipe Bridge operator to Blender's *File* > *Import* menu.
`register()` / `unregister()`	Registers and unregisters the add-on in Blender, integrating or removing it from the user interface.

DNA Add-on. The analyzed task was the complete workflow required to transform a facial performance into reusable facial animation data in Blender.

The HTA was performed by the research team through exploratory execution of each workflow, inspection of official documentation, tutorials, and community resources. Participants in the usability study did not perform the HTA; instead, the HTA served as an analyst-generated workflow model used to define questionnaire blocks and to contextualize the usability results.

For each workflow, the global task was decomposed into four macro-stages: setup and preparation, facial capture, file export and organization, and integration with Blender. Within each macro-stage, the analysis identified user operations, conditional steps, software-mediated operations, and potential points of friction. The resulting operation counts were not treated as direct measures of usability, but as descriptive indicators of workflow structure and operational complexity.

The present study did not measure end-to-end latency or import-time performance quantitatively. Therefore, the reported evaluation should be interpreted as a workflow and usability assessment rather than a runtime performance benchmark.

3.3 Usability Evaluation

To compare the user experience of FacePipe with a commercial facial capture tool, a usability evaluation was conducted using standardized user experience instruments as methodological references. Among the methods initially considered were the System Usability Scale (SUS) [Brooke, 1996], UMUX-LITE [Lewis *et al.*, 2013], and SUPR-Q [Sauro, 2015]. Although widely used in usability assessment, these instruments were not adopted directly because the goal of this study was not limited to obtaining a global usability score, but rather to compare user experience across specific stages of the facial animation pipeline.

Therefore, a questionnaire was developed based on the modular structure of the User Experience Questionnaire Plus (UEQ+), proposed by Schrepp and Thomaschewski [2019] and operationalized in the UEQ+ supporting material [Schrepp *et al*., 2021]. The flexibility of the UEQ+ allowed the selection of constructs more suitable for the evaluated workflow, which included distinct stages such as tool configuration, facial capture recording, file export, and Blender integration.

The adapted questionnaire was organized into five thematic blocks derived from the task analysis: Configuration and Preparation, Facial Capture Recording, Export and File Organization, Blender Integration, and Overall User Experience. Instead of using generic adjective pairs, the selected UEQ+ constructs were translated into contextualized statements directly related to facial capture and animation tasks. Because the instrument was adapted to the facial animation workflow, the results were interpreted as UEQ+-based comparative measures rather than as benchmarked UEQ+ scores. Table 2 presents the relationship between the thematic blocks, questionnaire statements, UEQ+ constructs, and item polarity.

3.3.1 Participants, procedure, and statistical analysis

Because the study was conducted remotely and unsupervised, participants did not use identical rooms, lighting conditions, camera models, camera resolutions, or physical setups. This design reflects a realistic low-infrastructure use scenario but also introduces uncontrolled variability. Consequently, differences between FacePipe and FaceCap should not be attributed exclusively to software design. They may also reflect hardware availability, camera quality, lighting, device performance, and local setup differences. This limitation is especially relevant for RGB-based facial capture, whose stability can be affected by illumination, face visibility, resolution, and head pose.

The usability evaluation followed a between-subjects design, in which each participant used only one of the two evaluated tools: FacePipe or FaceCap. This design avoided immediate comparison effects and reduced fatigue associated with using multiple systems sequentially. The experiment was approved by the Institutional Review Board of the University of Florida under protocol ET00047563.

A total of 160 undergraduate Computer Science students participated in the study. Participants were divided into two independent groups: 108 evaluated FacePipe and 52 evaluated FaceCap. The unequal group sizes resulted from an operational limitation related to device availability. FacePipe could be used with a standard RGB camera, whereas FaceCap required an iOS device compatible with facial capture. Thus, the imbalance between groups reflected the availability of compatible devices during data collection rather than post-hoc exclusion, performance filtering, or selective removal of responses.

Participants were recruited through institutional communi-

Table 2. Thematic blocks, questionnaire statements, UEQ+ constructs, and item polarity.

Block	Statement	UEQ+ construct	Polarity
Config. and Preparation	It was easy to configure the tool to start capturing.	Perspicuity / Efficiency	+
	I had difficulties starting the capture session.	Perspicuity	–
	The instructions were clear at the beginning of use.	Clarity	+
Facial Capture Recording	The recording process worked smoothly and reliably.	Efficiency / Dependability	+
	The tool responded in an unstable or unpredictable manner.	Dependability	–
	The interface during recording made it easier to understand what was being captured.	Clarity	+
	I felt in control throughout the recording process.	Dependability	+
Export and File Org.	Exporting data (FBX or JSON) was simple and clear.	Efficiency / Clarity	+
	The generated file structure (folders, names) made organization difficult.	Efficiency	–
	The exported format (FBX/JSON) met my expectations for later use.	Clarity	+
Blender Integration	I was able to import the data into Blender easily.	Efficiency / Clarity	+
	The data maintained fidelity to the original expression after import.	Dependability	+
	Integration with Blender required complex or unintuitive steps.	Perspicuity	–
Overall User Experience	The visual interface of the tool was pleasant.	Stimulation	+
	I felt frustrated at some point while using the tool.	Stimulation / Dependability	–
	The complete process was efficient and well resolved.	Efficiency	+
	I would use this tool again in a real production environment.	Stimulation / Clarity	+

cation channels and completed the experiment remotely in an unsupervised setting. They followed a predefined task script covering facial capture and application of the resulting animation data. The chronological sequence was as follows. First, participants accessed the study instructions and consent information. Second, they installed or opened the assigned tool according to their experimental condition. Third, they prepared a capture session using the provided instructions. Fourth, they recorded a short facial performance. Fifth, they exported the resulting animation data using the format supported by the assigned tool: `.json` for FacePipe or `.fbx` for FaceCap. Sixth, they opened Blender and imported the exported data, using the FacePipe Bridge add-on in the FacePipe condition or Blender's native FBX import workflow in the FaceCap condition. Seventh, they inspected whether the animation was applied to the 3D character. Finally, they answered the adapted UEQ+ questionnaire. The functional structure of the task was equivalent across conditions, although operational instructions were adapted to the specific workflow of each tool.

Questionnaire responses were collected using a 7-point Likert scale. For the adapted UEQ+ scale analysis, responses were converted to the bipolar -3 to $+3$ UEQ metric. Positively worded items were converted using $UEQ = Likert - 4$, while negatively worded items were reverse-scored using $UEQ = (8 - Likert) - 4$. This ensured that higher values consistently represented more favorable evaluations.

Because the study used independent groups with unequal sample sizes, comparisons between FacePipe and FaceCap were analyzed using Welch's t-test. Levene tests were used to inspect homogeneity of variance, and effect sizes were reported using Cohen's d. All statistical analyses were conducted in Jamovi 2.5, adopting a 5% significance threshold.

4 Results

The segmented analysis of the workflow provided a more nuanced understanding of user experience across different stages of the facial capture pipeline. Rather than relying only on aggregated averages, which can obscure relevant variation, the decomposition into thematic blocks revealed how each tool was perceived under specific interaction conditions. Overall, both FacePipe and FaceCap received positive evaluations, with relatively small differences emerging at distinct stages of the process.

4.1 Hierarchical task analysis results

Following Hackos and Redish [1998], the analysis focused on representative user goals rather than isolated interface functions. Each workflow scenario covered the complete process, from initial configuration to generating reusable facial animation in Blender. Hierarchical decomposition was then applied to break down goals into macro-stages, tasks, and subtasks, making explicit both physical actions and cognitive decisions, such as selecting export formats, interpreting system feedback, and organizing files. This approach helped reveal how operations that may appear trivial to experienced users can become barriers for novice users or for production contexts involving repeated captures and large volumes of files.

HTA enabled a structured comparison of how each solution transforms facial capture into usable Blender animation. The FaceCap workflow required multiple low-level file management operations, including accessing iOS directories, transferring FBX files, converting formats, and manually organizing folders. These operations increase the number of steps between the user's intention and the final animation asset, reflecting Norman's concept of the "gulf of execution" [Norman, 2013]. From Nielsen's usability perspective, this workflow also depends heavily on user memory and prior technical knowledge [Nielsen, 1993].

The FacePipe workflow simplified several of these operations through automatic folder creation, consistent naming conventions, and exporting to a `.json` format, which is handled by the FacePipe Bridge add-on in Blender. According to Hackos and Redish [1998], redistributing recurrent operational tasks from the user to the system can reduce unnecessary intermediary actions. Rogers *et al*. [2019] describe this

type of design decision as a closer alignment between system structure and the structure of the user's actual task.

The MetaHuman/Unreal Engine workflow integrated with the MetaHuman DNA Add-on presented the highest level of technical complexity. In addition to facial capture and animation, users needed to configure Unreal Engine plugins, use Live Link Face, export DNA and texture files, manage Control Rig tracks, and reconstruct characters in Blender. While this workflow offers greater flexibility and robust animation capabilities, it also creates a substantial operational barrier for less-experienced users.

Considering the three workflows together, HTA revealed different strategies for distributing tasks between the user and the system. FaceCap emphasized manual file and format management, FacePipe centralized and automated recurring organizational tasks, and the MetaHuman workflow demanded a more complex production infrastructure. Table 3 summarizes the quantitative comparison between the pipelines. FacePipe required 19 operations, FaceCap required 22, and the MetaHuman + PolyHammer BlenderDNA Add-on workflow required up to 92 operations, in addition to multiple conditional tasks related to plugin configuration, DNA files, and integration between Unreal Engine and Blender.

Table 3. Comparison of operation counts in the analyzed facial animation workflows.

Workflow	Operations	Min.	Max.	Conditional tasks
FacePipe	19	19	19	1
FaceCap	22	20	22	1
MetaHuman workflow	92	9	92	11

Note. MetaHuman workflow refers to the MetaHuman + PolyHammer BlenderDNA Add-on pipeline. This workflow includes multiple conditional tasks distributed across the HTA plans.

Rather than establishing a hierarchy between solutions, this comparison highlights how different design decisions produce distinct profiles of operational effort, potential error risk, and learning curve. In line with Stanton [2006] and Hackos and Redish [1998], task analysis is used here as a methodological tool to make these differences explicit and to support discussion of which workflow may be more suitable for specific production and research contexts.

4.2 Workflow-stage usability results

When all questions are considered together, the mean difference between the tools (FacePipe – FaceCap, after polarity correction) was approximately -0.13 points on the 1-7 scale. This indicates a small overall descriptive advantage for FaceCap, but not a substantial difference. Both tools were evaluated predominantly positively, with mean scores ranging from roughly 4.67 to 5.95 across the different competencies.

The detailed results show that this small overall difference reflects distinct workflow profiles rather than a general superiority of one tool over the other. FaceCap received higher scores in the initial stages of the process and in Blender integration, whereas FacePipe showed slightly better results in file export and organization while maintaining comparable performance in the recording stage.

4.2.1 Setup and preparation

The Setup and Preparation block evaluated the initial stage of interaction with the tools, including ease of configuration, clarity of initial instructions, and difficulty in starting a capture session. This phase is important because it represents the user's first contact with the system and may influence their willingness to continue using the tool.

The overall mean scores for this block were approximately 5.59 for FacePipe and 5.95 for FaceCap on a 1-7 scale. Both values indicate a positive experience. The difference of about 0.36 points suggests a descriptive advantage for FaceCap in perceived setup and preparation, including software installation, plugin configuration, and file conversion workflows.

At the item level, participants tended to evaluate FaceCap as easier to configure and clearer in its initial guidance. However, the item-level inferential results indicate that these differences should be interpreted cautiously: the setup-related items did not reach the conventional 5% significance threshold, although the item related to ease of configuration approached significance ($p = 0.057$) and showed a small effect size ($d = 0.32$). The remaining setup items also favored FaceCap descriptively, but with small effects.

This pattern can be partly explained by differences in the initial installation and preparation workflows. FaceCap is distributed through the Apple App Store on iOS, whereas FacePipe requires installation on Windows or macOS and also depends on the FacePipe Bridge add-on for JSON-based integration with Blender. In addition, FaceCap's FBX output can be imported directly into Blender, whereas FacePipe requires a compatible Blender setup with the dedicated import add-on.

From a user experience perspective, these findings suggest that FaceCap may offer a smoother entry curve, especially for users who prioritize rapid onboarding. However, because the observed differences in this block were small and not statistically conclusive, this result should be interpreted as a tendency in perceived setup experience rather than as strong evidence of superiority.

4.2.2 Facial capture recording

The Facial Capture Recording block focused on the user experience during recording, including workflow smoothness, perceived stability, and a sense of control throughout the session.

The overall mean scores for this block were approximately 5.58 for FacePipe and 5.63 for FaceCap, indicating very close and positive evaluations for both tools. The polarity-corrected mean difference was approximately -0.05 points (FacePipe – FaceCap), suggesting a negligible difference in practical terms.

At the item level, FacePipe showed a slight descriptive advantage in the item related to interface clarity during recording. This may be associated with its structured scene organization and the possibility of playing back multiple stored captures, which can help users understand what has been recorded. In contrast, FaceCap showed slightly higher descriptive scores in items related to recording fluidity, perceived stability, and sense of control during the process.

However, the inferential results indicate that these item-level differences should not be interpreted as substantive. None of the recording-stage items reached the conventional 5% significance threshold, and the observed effect sizes were negligible to very small. This supports the interpretation that the recording experience was broadly comparable between the two tools.

Overall, although FacePipe and FaceCap differ in specific interface and workflow features, participants perceived the actual facial capture stage in a highly similar way. Both tools were evaluated as sufficiently stable and usable during recording.

4.2.3 Exporting and organizing files

The File Export and Organization block evaluated the experience of exporting capture data and organizing the generated files, including the clarity of export options, ease of locating files, and perceived order in the workflow after recording.

In this block, the overall mean scores were approximately 5.31 for FacePipe and 5.14 for FaceCap, resulting in a difference of about 0.17 points (FacePipe - FaceCap). This suggests a slight descriptive advantage for FacePipe in this stage of the workflow.

At the item level, participants tended to evaluate FacePipe more favorably for export clarity and file organization. This pattern may be related to FacePipe's built-in mechanisms for organizing captures into scene folders and renaming files directly within the application. In contrast, FaceCap relies more heavily on external file management and transfer mechanisms, such as mobile file managers, cloud services, or AirDrop, with more limited post-capture organization within the tool itself.

However, the inferential results indicate that these differences should be interpreted cautiously. None of the items in this block reached the conventional 5% significance threshold, and the observed effect sizes were small. Thus, the results do not demonstrate a statistically robust advantage for FacePipe, but they do suggest that its design decisions may provide a more organized export workflow from the user's perspective.

This tendency is relevant for production contexts involving repeated captures or large volumes of files, where structured export paths, consistent naming, and direct access to generated data can support workflow efficiency and traceability.

4.2.4 Integration with Blender

The Blender Integration block evaluated users' experience when transferring facial capture data into Blender, including import ease, perceived data fidelity, and the smoothness of the transition between the capture tool and the 3D environment.

The mean scores for this block were approximately 4.66 for FacePipe and 5.11 for FaceCap. The polarity-corrected mean difference was about -0.45 points (FacePipe – FaceCap), representing the largest gap among the evaluated workflow stages and indicating a descriptive advantage for FaceCap in Blender integration.

The item-level results provide stronger support for this pattern. Although the difference in ease of importing data into Blender was not statistically significant, FaceCap received significantly higher scores for perceived fidelity of the imported expression ($p = 0.029$, $d = 0.37$) and for lower perceived complexity of the integration process ($p = 0.047$, $d = 0.34$). These effect sizes are small to moderate, indicating that the Blender integration stage was the clearest point of differentiation between the tools.

This result can be explained by differences in the import workflow. FaceCap exports FBX files that can be imported directly into Blender and include an embedded facial rig, whereas FacePipe exports `.json` data that requires the FacePipe Bridge add-on and a compatible character setup with corresponding blendshapes. As a result, FacePipe may demand additional knowledge of Blender's shape key and animation assignment workflow, increasing perceived complexity during integration.

Additionally, FaceCap's ARKit-based capture may have contributed to a stronger perceived expression fidelity upon import. However, this interpretation should be treated cautiously, because the study measured perceived fidelity rather than objective tracking accuracy. The results, therefore, support a difference in user experience during Blender integration, not a definitive claim about the technical precision of capture.

In practical terms, Blender integration is a critical stage in facial animation workflows, and additional steps during import can significantly affect perceived usability. However, these results are based on a controlled scenario involving a single import workflow. In more realistic production contexts, both tools may require additional adjustments, such as adapting or replacing default assets, preparing compatible character rigs, or repeating imports across multiple captures. Future studies should therefore examine repeated imports and more complex production pipelines to better reflect real-world usage conditions.

4.2.5 Overall user experience

The Overall User Experience block summarized the general evaluation of the tools, including perceived interface quality, frustration, process efficiency, and willingness to use the tool again in a production scenario.

The mean scores for this block were approximately 4.96 for FacePipe and 5.00 for FaceCap. The polarity-corrected mean difference was approximately -0.05 points (FacePipe – FaceCap), indicating a negligible difference between the tools in the overall assessment.

At the item level, FacePipe showed a descriptive advantage in the perceived pleasantness of the visual interface. In contrast, FaceCap showed slightly higher descriptive scores in items related to lower frustration, overall process efficiency, and willingness to use the tool again in a real production scenario. However, none of these item-level differences reached the conventional 5% significance threshold, and the observed effect sizes were small.

These results indicate that the item-level variations tend to balance out when considered as a block. Thus, there is no clear dominance of one tool over the other in terms of overall user experience. Both FacePipe and FaceCap provided a satisfactory global experience, with differences appearing more clearly in specific workflow stages than in the overall evaluation.

This pattern is consistent with the aggregated findings presented in Table 4, which summarizes the polarity-corrected mean values across thematic blocks on the 1 to 7 Likert scale.

4.3 UEQ+ dimension results

The evaluation of the adapted UEQ+ scales was conducted using the consolidated dimensional table derived from the raw questionnaire responses, separately considering the groups that evaluated FacePipe and FaceCap. For each dimension—Clarity, Perspicuity, Efficiency, Dependability, and Stimulation—mean scores on the -3 to $+3$ scale, standard deviations, number of observations, and 95% confidence intervals were calculated. Negatively worded items were reverse-scored prior to analysis, so higher values consistently indicate a more favorable evaluation.

An important methodological aspect is that the mapping of questions to UEQ+ dimensions was not strictly exclusive. Some items contributed simultaneously to more than one dimension, reflecting how a single interaction aspect may affect different facets of user experience. For instance, questions related to the initial setup process were associated with both Perspicuity and Efficiency, since they influence not only ease of understanding and onboarding, but also workflow fluency. Consequently, the same response could contribute to multiple dimensional aggregates, creating intentional conceptual and statistical overlap. For this reason, the dimensions should be interpreted as complementary perspectives on the user experience rather than as fully independent measures.

The results indicate that both tools were positively evaluated across all investigated dimensions. Mean scores for FacePipe and FaceCap remained above zero in Clarity, Perspicuity, Efficiency, Dependability, and Stimulation, suggesting an overall favorable experience with both solutions. No neutral or clearly negative dimensions were identified, reinforcing the viability of both tools for facial performance capture workflows.

Nevertheless, the comparison between the tools reveals distinct usage profiles. FaceCap achieved slightly higher scores in dimensions associated with initial understanding, predictability, and system stability. In Perspicuity, the results suggest fewer difficulties related to setup and onboarding. In Clarity, associated with interface transparency and system feedback, FaceCap also achieved slightly higher average scores. A similar pattern was observed in Dependability, indicating a stronger perception of stability and control during data capture and export processes.

In Stimulation, however, FacePipe and FaceCap produced nearly equivalent results, indicating similar levels of enjoyment, motivation to continue using the tool, and intention to reuse the tool. In Efficiency, although small numerical differences were observed, the confidence intervals of both tools overlapped considerably, suggesting comparable perceived performance throughout the workflow.

The 95% confidence intervals presented in Table 5 help contextualize these differences. In several dimensions, the confidence intervals of FacePipe and FaceCap overlap, indicating that the observed differences should not be interpreted as large practical distinctions. Still, the overall pattern is consistent with the block-level analysis: FaceCap tends to be perceived as clearer, more predictable, and more reliable, whereas FacePipe remains positively evaluated across all dimensions.

Overall, the dimensional analysis reinforces the workflow-stage results: FaceCap shows a consistent descriptive tendency toward higher perspicuity and dependability, whereas FacePipe remains positively evaluated across all dimensions, with no statistically conclusive dimensional disadvantage. The findings suggest that the choice between FacePipe and FaceCap should be guided more by contextual requirements than by assuming one tool is objectively superior. In scenarios where initial clarity, predictability, and a sense of control are priorities, the results favor FaceCap. In contrast, for workflows that value structured capture management and an open Blender-oriented pipeline, FacePipe remains a viable alternative, maintaining strong engagement and an overall positive user experience.

5 Discussion

The results indicate that FacePipe and FaceCap should not be interpreted as tools positioned along a single continuum from worse to better. Instead, they represent different design trade-offs within the facial animation pipeline. FaceCap benefits from a mature ARKit-based ecosystem, tighter coupling between hardware and software, and a more consolidated capture workflow. FacePipe, by contrast, prioritizes low-cost hardware availability, open implementation, structured file management, and direct integration with Blender through a dedicated add-on. The central finding of this study is therefore not that one tool is categorically superior, but that different design decisions redistribute user effort across different stages of facial animation work.

5.1 Workflow trade-offs between FacePipe and FaceCap

The workflow-stage results suggest that the main differences between the tools occur around the capture process rather than exclusively within the act of facial recording itself. The recording stage was perceived as broadly comparable between FacePipe and FaceCap, which indicates that participants did not experience the RGB-based capture process of FacePipe as substantially less usable during the immediate performance-recording task. This is an important result because it suggests that a low-cost RGB-camera-based tool can provide an acceptable interaction experience for basic facial capture workflows, even without relying on specialized capture hardware.

However, FaceCap was evaluated more favorably in setup, perceived dependability, and Blender integration. These advantages are consistent with its reliance on Apple ARKit and compatible iOS hardware, where the hardware, tracking framework, and capture application are more tightly integrated. From the user's perspective, this integration may reduce uncertainty during initial configuration and may contribute to a stronger sense of predictability. The results related to Blender integration are particularly relevant because they show that the final stage of the pipeline can strongly affect the perceived quality of the entire workflow. Even when capture

Table 4. FacePipe vs. FaceCap by thematic workflow blocks with item-level Welch comparisons.

Workflow block / Item	FacePipe $N = 108$	FaceCap $N = 52$	Δ FC–FP	t	df	p	d
Setup and preparation							
1. It was easy to configure the tool to start capturing.	5.69 (1.31)	6.10 (1.24)	0.41	1.93	105.8	.057	0.32
2. Starting the capture session was not difficult.	5.45 (1.61)	5.80 (1.72)	0.35	1.25	95.2	.215	0.21
3. The instructions were clear at the beginning of use.	5.62 (1.24)	5.94 (1.39)	0.32	1.42	91.3	.160	0.24
Block mean	**5.59**	**5.95**	**0.36**	–	–	–	–
Facial capture recording							
4. The recording process worked smoothly and reliably.	5.72 (1.40)	5.77 (1.41)	0.05	0.20	100.2	.843	0.03
5. The tool did not respond in an unstable or unpredictable manner.	5.44 (1.46)	5.63 (1.52)	0.19	0.75	96.9	.455	0.13
6. The interface during recording made it easier to understand what was being captured.	5.63 (1.32)	5.52 (1.57)	-0.11	-0.44	87.2	.662	-0.08
7. I felt in control throughout the recording process.	5.53 (1.50)	5.60 (1.45)	0.07	0.28	104.3	.782	0.05
Block mean	**5.58**	**5.63**	**0.05**	–	–	–	–
File export and organization							
8. Exporting data (FBX or JSON) was simple and clear.	5.69 (1.26)	5.50 (1.53)	-0.19	-0.80	85.7	.428	-0.14
9. The generated file structure supported file organization.	5.09 (1.61)	4.77 (1.69)	-0.32	-1.15	96.9	.253	-0.20
10. The exported format (FBX/JSON) met my expectations for later use.	5.16 (1.35)	5.15 (1.55)	-0.00	-0.01	89.6	.989	-0.00
Block mean	**5.31**	**5.14**	**-0.17**	–	–	–	–
Integration with Blender							
11. I was able to import the data into Blender easily.	4.88 (1.74)	5.08 (1.90)	0.20	0.63	93.5	.529	0.11
12. The data maintained fidelity to the original expression after import.	4.91 (1.64)	5.48 (1.49)	0.57	2.21	110.0	.029	0.37
13. Integration with Blender did not require complex or unintuitive steps.	4.22 (1.70)	4.79 (1.65)	0.57	2.01	103.8	.047	0.34
Block mean	**4.67**	**5.12**	**0.45**	–	–	–	–
Overall user experience							
14. The visual interface of the tool was pleasant.	5.68 (1.16)	5.27 (1.68)	-0.41	-1.57	75.1	.120	-0.28
15. I did not feel frustrated while using the tool.	4.26 (1.78)	4.65 (1.75)	0.39	1.33	102.4	.187	0.22
16. The complete process was efficient and well resolved.	5.13 (1.45)	5.23 (1.60)	0.10	0.39	92.1	.701	0.07
17. I would use this tool again in a real production environment.	4.76 (1.63)	4.85 (1.49)	0.09	0.34	109.9	.738	0.06
Block mean	**4.96**	**5.00**	**0.04**	–	–	–	–

Note. Values are means with standard deviations in parentheses, computed on a 1–7 Likert scale after polarity correction, so higher scores always indicate a more favorable evaluation. Δ = FaceCap mean minus FacePipe mean. Welch's t-test was used because the groups were independent and unequal in size. Effect size is Cohen's d. Block means are descriptive summaries of the corresponding items; inferential statistics are reported at the item level only.

Table 5. Adapted UEQ+ dimension results for FacePipe and FaceCap with Welch comparisons.

UEQ+ dimension	FacePipe M (SD)	FaceCap M (SD)	Δ FC–FP	95% CI_{Δ}	t	df	p	Interpretation
Clarity	1.29 (1.48)	1.34 (1.61)	0.05	[-0.48, 0.58]	0.19	93.6	.851	Nearly equivalent scores; no evidence of a relevant difference.
Dependability	1.17 (1.64)	1.43 (1.57)	0.26	[-0.27, 0.79]	0.97	104.9	.336	Descriptive advantage for FaceCap, but the difference is small and not statistically conclusive.
Efficiency	1.37 (1.51)	1.41 (1.62)	0.04	[-0.49, 0.57]	0.15	94.7	.881	Practically equivalent perceived efficiency across tools.
Perspicuity	1.12 (1.67)	1.56 (1.64)	0.44	[-0.11, 0.99]	1.58	102.5	.117	Largest descriptive advantage for FaceCap, consistent with smoother onboarding, but not statistically conclusive at $\alpha = .05$.
Stimulation	0.90 (1.65)	0.92 (1.65)	0.02	[-0.53, 0.57]	0.07	100.8	.943	Nearly identical scores; both tools produced similar perceived stimulation and willingness to continue use.

Note. UEQ+ dimension scores are reported on the -3 to $+3$ scale after polarity correction, so higher values indicate a more favorable evaluation. N(FacePipe) = 108 participants; N(FaceCap) = 52 participants. Δ = FaceCap mean minus FacePipe mean. Welch's t-test was used because the groups were independent and unequal in size. Effect sizes were small across all dimensions: Clarity, $d = 0.03$; Dependability, $d = 0.16$; Efficiency, $d = 0.03$; Perspicuity, $d = 0.27$; Stimulation, $d = 0.01$. Some items contributed simultaneously to more than one UEQ+ dimension; therefore, dimensions are not completely independent and should be interpreted as complementary perspectives on user experience rather than as separate psychometric scales.

itself is usable, users may judge the tool less favorably if the transition from recorded data to editable animation inside Blender is perceived as complex, uncertain, or insufficiently transparent.

FacePipe showed its main descriptive strength in export and file organization. This result is coherent with the design of the tool, which emphasizes structured capture management, local data organization, and a `.json`-based export format handled by the FacePipe Bridge add-on. In practical terms, this suggests that FacePipe may be especially useful in contexts where users perform multiple captures, need to organize scenes and takes, or want inspectable animation data that can be reused and edited within Blender. Therefore, FacePipe's contribution lies less in eliminating all workflow friction and more in shifting part of the workflow toward an open and locally manageable structure.

The HTA results reinforce this interpretation. Operation counts and task decomposition should not be treated as direct measurements of usability, since fewer steps do not automatically mean a better user experience. Some steps may be cognitively demanding, while longer workflows may be acceptable when they are predictable and well supported by the software. Nevertheless, HTA helps reveal how each tool distributes operational effort. FacePipe reduces dependence on proprietary capture hardware and supports a relatively direct path from RGB capture to Blender editing. FaceCap provides a more polished capture ecosystem but depends on compatible devices and external transfer/integration procedures. The MetaHuman-based workflow, in turn, represents a more complex production-oriented pipeline whose operational richness may be appropriate for high-fidelity character workflows, but less aligned with the low-cost and lightweight use cases targeted by FacePipe.

5.2 Scope of comparison and tracking-accuracy trade-off

A central limitation of the present study is that it evaluates workflow structure and perceived user experience, not objective facial tracking accuracy. Therefore, the results should not be interpreted as evidence that FacePipe and FaceCap produce equivalent capture accuracy, expression fidelity, or temporal stability. This distinction is essential because FaceCap and FacePipe rely on different technical foundations. FaceCap is based on Apple ARKit and compatible iOS devices, whereas FacePipe uses generalized MediaPipe models and conventional RGB cameras. These design choices imply a clear trade-off between capture robustness and hardware availability.

The user study included items related to perceived fidelity after import, but perceived fidelity is not equivalent to objective tracking accuracy. Participants judged whether the resulting animation appeared acceptable within the task context, but the study did not compare captured expressions against a ground-truth facial performance, did not compute geometric error, did not analyze expression-level correspondence, and did not measure temporal latency or frame-level stability. Consequently, claims about the comparison between FacePipe and FaceCap must remain limited to the evaluated dimensions: workflow usability, perceived integration quality, perceived dependability, and overall user experience.

This clarification is important because the absence of objective tracking metrics could otherwise suggest that tracking accuracy was omitted because FacePipe performed worse. The more accurate interpretation is that the study was designed to answer a different research question. FacePipe is not presented here as a replacement for high-end or tightly integrated facial capture systems when the primary requirement is maximum tracking precision. Rather, it is presented as a low-cost, open, and editable facial animation pipeline for contexts in which hardware availability, transparency, and Blender-based workflow integration are central requirements. Users who prioritize capture robustness, micro-expression fidelity, fast speech articulation, or depth-assisted tracking may still prefer FaceCap or other specialized systems. Users who prioritize low-cost deployment, open-source modification, structured data export, and local Blender editing may find FacePipe more appropriate.

5.3 Hardware availability, openness, and terminology

The results also suggest that the value of FacePipe should be framed in terms of hardware and economic availability. FacePipe lowers the entry barrier to facial capture by using conventional RGB cameras and open software components, but this does not mean that the system is inherently accessible to users with sensory, motor, or cognitive impairments.

The distinction between hardware availability and openness is also important. Hardware availability refers to the ability to use common and low-cost input devices, such as webcams or integrated laptop cameras. Openness refers to the transparency, modifiability, and reproducibility of the software pipeline, including source code, export format, and Blender integration. These two principles are complementary but not identical. A closed tool may run on widely available hardware, while an open tool may still require specialized devices. FacePipe combines these two design priorities by using widely available RGB input and by providing an open pipeline that can be inspected, modified, and extended.

This positioning is especially relevant for educational, experimental, and independent production contexts. In these contexts, the most important requirement is not always maximum tracking fidelity, but the ability to understand, adapt, teach, and reuse the workflow. FacePipe supports this type of use by making the capture data explicit, storing animation information in a structured format, and importing it into Blender as editable animation curves. These characteristics may be less visible in a purely accuracy-centered benchmark, but they are central to the design problem addressed in this paper.

5.4 Remote evaluation and ecological validity

The remote and unsupervised nature of the usability study is another important factor in interpreting the results. Participants did not use identical rooms, lighting conditions, cameras, camera resolutions, or physical setups. This variability is particularly relevant for RGB-based facial capture, since

illumination, camera quality, face visibility, compression, frame rate, and head pose can affect tracking stability. Therefore, differences between FacePipe and FaceCap cannot be attributed exclusively to interface design or workflow structure. They may also reflect differences in local hardware and environmental conditions.

At the same time, this limitation also gives the study a degree of ecological validity. FacePipe is intended for low-infrastructure contexts, where users are unlikely to have standardized capture rooms, controlled lighting, or homogeneous hardware. A fully controlled laboratory setup would reduce variability, but it would also move the evaluation away from the conditions in which a low-cost RGB-based tool is likely to be used. The present results should therefore be interpreted as evidence of perceived usability under realistic but uncontrolled conditions, not as a controlled performance benchmark.

This interpretation helps explain why FaceCap may have been perceived as more dependable. Its dependence on compatible iOS hardware constrains the range of possible setups, whereas FacePipe accepts a broader range of RGB cameras. The same flexibility that increases hardware availability also increases variability. Thus, part of the FacePipe trade-off is structural: broad compatibility and low entry cost may come at the expense of more heterogeneous capture quality and less predictable performance across users.

5.5 Implications for low-cost facial animation pipelines

The findings have implications for the design of low-cost facial animation tools. First, they show that usability should be evaluated across the entire production pipeline, not only at the point of capture. Setup, recording, export, file organization, import, and post-capture editing all contribute to the user's perception of the tool. A capture system that performs well during recording may still be perceived negatively if export or integration is confusing. Conversely, a technically simpler capture system may remain valuable if it provides transparent data organization and editable output.

Second, the results suggest that open facial animation tools should invest not only in tracking quality but also in workflow communication. Users need to understand what the tool captures, what is exported, how the data maps to the character rig, and what can be edited after import. The FacePipe Bridge add-on is a step in this direction because it makes the transition from capture file to Blender animation more direct. However, the lower scores related to Blender integration indicate that this stage still requires refinement. Better onboarding, clearer import feedback, automated validation of compatible rigs, and more visible error messages could improve perceived dependability and perspicuity.

Third, the comparison reinforces the importance of designing for different user priorities. A commercial tool may be preferable when users require a polished ecosystem, rapid setup, and robust device-level tracking. An open RGB-based tool may be preferable when users need affordability, modifiability, and integration with educational or experimental workflows. Rather than treating these priorities as mutually exclusive, future development should aim to reduce the gap between them by improving FacePipe's robustness while preserving its low-cost and open characteristics.

5.6 Limitations and future evaluation

Several limitations should guide the interpretation of the study and the design of future evaluations. The between-subjects design reduced fatigue and avoided direct comparison effects, but it also means that individual participants did not experience both tools. Future work could adopt a counterbalanced within-subjects design with controlled hardware conditions to compare perceived workflow quality more directly. In addition, the unequal group sizes were caused by device availability constraints, but future studies should aim for balanced groups.

The participant sample was composed of undergraduate Computer Science students, which may not represent professional animators, technical artists, educators, or independent creators. Since facial animation pipelines are strongly shaped by production experience, future evaluations should include users with different levels of expertise in Blender, rigging, animation, and motion capture. Expert evaluations could reveal workflow issues that novice users may not perceive, especially in repeated production scenarios involving multiple characters, retargeting, correction of animation curves, and integration into larger scenes.

Future work should also include objective technical benchmarks. These should measure tracking accuracy, expression-level correspondence, temporal stability, frame rate, end-to-end latency, import time, and robustness under different lighting conditions, camera resolutions, head poses, occlusions, and speech speeds. Such benchmarks would make it possible to quantify the accuracy trade-off between FacePipe and ARKit-based alternatives. They would also allow the system to be improved in a more targeted way, for example by identifying which facial regions, expressions, or movement conditions produce the largest errors.

Finally, future versions of FacePipe should improve the Blender integration stage, since this was one of the points where FaceCap obtained stronger evaluations. Potential improvements include automatic rig compatibility checks, clearer mapping between captured coefficients and shape keys, visual previews before import, more robust handling of missing or renamed blendshapes, and better documentation for first-time users. These improvements would directly address the main usability gap identified in the study while preserving FacePipe's central design goal: providing a low-cost, open, and editable facial animation pipeline based on widely available RGB hardware.

6 Conclusion

This paper presented FacePipe, a low-cost real-time facial capture tool based on conventional RGB cameras and open-source technologies, without requiring dedicated depth sensors or proprietary capture hardware. In a context where commercial solutions such as FaceCap depend on Apple devices and ARKit infrastructure, FacePipe offers an openly

available, low-cost and multiplatform alternative for educational, experimental, and independent production contexts.

The study showed that facial animation pipelines involve complex and interdependent stages, making it important to evaluate not only capture functionality, but also how users move through setup, recording, export, file organization, and integration with 3D animation software. The comparison between FacePipe and FaceCap, conducted through an adapted UEQ+-based evaluation segmented by workflow stages, indicated that both tools were evaluated positively across most dimensions.

The results do not support the categorical superiority of one tool over the other. Instead, they indicate different workflow profiles. FaceCap showed stronger results in setup, initial clarity, perceived dependability, and especially Blender integration, where item-level differences related to perceived fidelity and integration complexity were more evident. FacePipe, in turn, showed a descriptive strength in export and file organization, supported by its scene-based folder structure, built-in file management, and structured `.json` output handled through the FacePipe Bridge add-on.

The recording stage was perceived as broadly comparable between the two tools, suggesting that the main differences were not concentrated in the act of facial capture itself, but in the surrounding workflow. This reinforces the importance of evaluating facial animation tools as complete pipelines rather than isolated capture mechanisms.

Several limitations must be acknowledged. The study used a between-subjects design with unequal group sizes, due to the availability constraints of iOS-compatible devices for FaceCap. Participants were undergraduate Computer Science students, and the experiment was conducted remotely in an unsupervised setting, which may have introduced variability in hardware, lighting, camera quality, and local setup. This environmental variability applied to both conditions, although in different ways: in the FacePipe condition it spans the full RGB capture chain, since participants used whatever webcam or integrated camera they had available, whereas in the FaceCap condition the capture sensor itself is standardized through Apple's TrueDepth hardware, even though the iOS device, lighting, and physical setup still varied. Because this variability was present in both groups, no systematic environmental advantage was conferred on either tool, and the comparison reflects realistic, rather than idealized, usage that mirrors the independent and educational deployment context the tools target. The Blender integration task was also based on a controlled single-import scenario, which may not fully represent repeated production use with multiple characters, captures, or retargeting requirements.

Technical limitations of FacePipe also remain, and the scope of this study must be stated plainly. The evaluation measured perceived user experience across workflow stages through an adapted UEQ+ instrument, not capture accuracy through a benchmark. This distinction is not incidental but structural: each participant interacted with only one of the two tools. As a result, no participant was in a position to directly compare the capture precision of the two pipelines, and the instrument was never intended to elicit such a comparison. Objective tracking accuracy, for example ground-truth landmark or blendshape error, cannot be recovered from this design and would require a separate study, whether a within-subjects protocol or an instrumented comparison against reference data. Assessing capture accuracy was therefore outside the objectives and the methodological reach of the present work. That said, because FacePipe is based on generalized models and monocular RGB input, while FaceCap relies on a dedicated depth sensor, FaceCap is expected to provide higher capture precision, particularly for micro-expressions, rapid speech movements, occlusions, extreme lighting, and large head rotations. The higher perceived fidelity reported during Blender integration (Section 4.2.4) is consistent with this expected advantage of depth-based capture, but, being a subjective and within-tool impression rather than an objective measurement, it does not by itself establish it. These limitations define the current scope of the tool: FacePipe is best understood as an openly available, low-cost, and editable facial animation pipeline, not as a replacement for high-end facial capture systems.

From a scientific perspective, this work contributes by relating user experience to concrete stages of facial animation workflows. The findings highlight that interface clarity, file organization, interoperability, and integration effort are central factors in the adoption of facial capture systems, alongside capture quality and visual fidelity. The work also reinforces the relevance of low-cost, widely available, and open solutions within the facial animation ecosystem, particularly in scenarios that require editable and controllable animation data.

Future work should investigate objective tracking accuracy, temporal stability, repeated import workflows, larger and more diverse participant samples, and production scenarios involving professional animators. Further development may also explore improved tracking robustness, broader interoperability with animation software and game engines, and integration with generative AI approaches for facial animation. Overall, FacePipe demonstrates that low-cost and open facial capture pipelines can provide viable alternatives for specific educational, experimental, and independent production contexts.

Declarations

Authors' contributions

ATCC: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Writing – original draft. **CCA**: Validation, Visualization, Writing – original draft. **WLC**: Validation, Visualization, Writing – review & editing. **FFFP**: Formal analysis, Methodology, Writing – review & editing. **WFMC**: Formal analysis, Methodology, Writing – review & editing. **AS**: Methodology, Project administration, Writing – review & editing. **FLSN**: Project administration, Writing – review & editing. **JMXNT**: Project administration, Resources, Supervision, Validation, Writing – review & editing. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Acknowledgements

The authors thank the participants who took part in the usability evaluation and contributed to the assessment of the FacePipe prototype.

Funding

The authors received no funding for this research.

Availability of data and materials

The FacePipe project website is publicly available at `https://facepipe.sircrux.com/`. The source code and related materials are publicly available in the project repository at `https://github.com/artutava/facepipe-app`.

References

Baltrušaitis, T. *et al*. (2018). OpenFace 2.0: Facial behavior analysis toolkit. In *Proceedings of the 13th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2018)*, pages 59–66, Xi'an, China. IEEE. DOI: 10.1109/FG.2018.00019.

Baxter, M. (1995). *Product Design*. CRC Press, Boca Raton, FL, 1 edition. DOI: 10.1201/9781315275246.

Beane, A. (2012). *3D Animation Essentials*. John Wiley & Sons, Hoboken, NJ, 1 edition. DOI: 10.3389/fcomp.2025.1598099.

Benyon, D. (2013). *Designing Interactive Systems: A Comprehensive Guide to HCI, UX and Interaction Design*. Pearson, Harlow, UK. Book.

Brooke, J. (1996). SUS: A "quick and dirty" usability scale. In Jordan, P. W. *et al*., editors, *Usability Evaluation in Industry*, pages 189–194. Taylor & Francis, London. DOI: 10.1201/9781498710411.

Chatziagapi, A. and Samaras, D. (2023). AVFace: Towards detailed audio-visual 4D face reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16878–16889, Los Alamitos, CA. IEEE Computer Society. DOI: 10.1109/CVPR52729.2023.01619.

Chhatre, K., Guarese, R., Matviienko, A., and Peters, C. E. (2025). Evaluation of generative models for emotional 3d animation generation in vr. *Frontiers in Computer Science*, Volume 7 - 2025. DOI: 10.3389/fcomp.2025.1598099.

Cudeiro, D., Bolkart, T., Laidlaw, C., Ranjan, A., and Black, M. J. (2019). Capture, learning, and synthesis of 3d speaking styles. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10093–10103. DOI: 10.1109/CVPR.2019.01034.

Danecek, R. *et al*. (2022). EMOCA: Emotion driven monocular face capture and animation. *arXiv preprint arXiv:2204.11312*. DOI: 10.48550/arXiv.2204.11312.

de Carvalho Cruz, A. T. and Xavier Natario Teixeira, J. M. (2021). A review regarding the 3d facial animation pipeline. In *Proceedings of the 23rd Symposium on Virtual and Augmented Reality*, pages 192–196. DOI: 10.1145/3488162.3488226.

Ekman, P. and Friesen, W. V. (1978). *Facial Action Coding System: A Technique for the Measurement of Facial Movement*. Consulting Psychologists Press, Palo Alto, CA. DOI: 10.1037/t27734-000.

Fan, Y., Lin, Z., Saito, J., Wang, W., and Komura, T. (2022). Faceformer: Speech-driven 3d facial animation with transformers. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18749–18758. DOI: 10.1109/CVPR52688.2022.01821.

Feng, Y. *et al*. (2021). Learning an animatable detailed 3D face model from in-the-wild images. *arXiv preprint arXiv:2012.04012*. DOI: 10.48550/arXiv.2012.04012.

Freitas-Magalhães, A. (2018). *Facial Action Coding System 3.0: Manual of Scientific Codification of the Human Face*. Escrytos, Lisboa. Book.

Gatti Jr., W., Gonçalves, M. A., and Paes, A. C. (2014). Um estudo exploratório sobre a indústria brasileira de animação para a TV. *Revista Eletrônica de Administração*, 20(2):461–495. DOI: 10.1590/1413-2311057201238250.

Google (2025). Meet Flow: AI-powered filmmaking with Veo 3. Google Blog. Available at: `https://blog.google/innovation-and-ai/products/google-flow-veo-ai-filmmaking-tool/` Acesso em: 03 jan. 2026.

Hackos, J. T. and Redish, J. C. (1998). *User and Task Analysis for Interface Design*. Wiley, New York, NY. Available at: `https://scispace.com/pdf/user-and-task-analysis-for-interface-design-370fbm6hl.pdf`.

Hausman, A. (2019). Keep the classic cartoons current. The Prowler News. Available at: `https://www.theprowlernews.org/uncategorized/2019/01/23/keep-the-classic-cartoons-current/` Acesso em: 2 nov. 2020.

Hernandez, M. *et al*. (2012). Laser scan quality 3-D face modeling using a low-cost depth camera. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, pages 1995–1999, Bucharest, Romania. Available at: `https://www.eurasip.org/Proceedings/Eusipco/Eusipco2012/Conference/papers/1569587493.pdf`.

Karras, T. *et al*. (2017). Audio-driven facial animation by joint end-to-end learning of pose and emotion. *ACM Transactions on Graphics*, 36(4). DOI: 10.1145/3072959.3073658.

Kerlow, I. V. (2009). *The art of 3D computer animation and effects*. Wiley, Hoboken, NJ, 4 edition. Available at: `https://archive.org/details/artof3dcomputera0000kerl_h6h8`.

Kling AI (2025). Motion control. Available at: `https://kling.ai/quickstart/motion-control-user-guide` Acesso em: 03 jan. 2026.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521:436–444. DOI: 10.1038/nature14539.

Lewis, J. *et al*. (2013). UMUX-LITE: when there's no time for the SUS. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 2099–2102, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2470654.2481287.

Lewis, J. P. (1991). Automated lip-sync: background and techniques. *Journal of Visualization and Computer Animation*, 2(4):118–122. DOI: 10.1002/VIS.4340020404.

Li, T. *et al*. (2017). Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics*, 36(6):1–17. DOI: 10.1145/3130800.3130813.

Ma, S. *et al*. (2024). 3D Gaussian blendshapes for head avatar animation. In *Proceedings of the ACM SIGGRAPH Conference*, pages 1–10, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3641519.3657462.

Ma, Y. *et al*. (2025). Controllable video generation: a survey. *arXiv preprint arXiv:2507.16869*. Acesso em: 03 jan. 2026. DOI: 10.48550/arXiv.2507.16869.

Mao, J. Y. *et al*. (2001). User-centered design methods in practice: a survey of the state of the art. In *Proceedings of the 2001 conference of the Centre for Advanced Studies on Collaborative research (CASCON '01)*, page 12. IBM Press. Available at: `https://scispace.com/pdf/user-centered-design-methods-in-practice-a-survey-of-the-019pox9x67.pdf`.

Marcotte, E. (2011). *Responsive web design*. Editions Eyrolles. Available at: `https://archive.org/details/responsivewebdesign`.

Mitchell, B. (2017). *Independent animation: developing, producing and distributing your animated films*. CRC Press, Boca Raton, FL, 1 edition. DOI: 10.1201/9781315363974.

Nielsen, J. (1993). *Usability Engineering*. Morgan Kaufmann, San Diego. DOI: 10.1016/C2009-0-21512-1.

Norman, D. A. (2004). *Emotional Design: Why We Love (or Hate) Everyday Things*. Basic Books, New York, NY. Book.

Norman, D. A. (2013). *The Design of Everyday Things: Revised and Expanded*. Basic Books, New York, NY. Book.

Orvalho, V. *et al*. (2012). A facial rigging survey. In *Eurographics 2012 – State of the Art Reports*, pages 183–204, Cagliari, Italy. The Eurographics Association. DOI: 10.2312/conf/EG2012/stars/183-204.

Parke, F. I. (1972). Computer generated animation of faces. In *Proceedings of the ACM annual conference - Volume 1 (ACM '72)*, pages 451–457, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/800193.569955.

Parke, F. I. (1974). *A Parametric Model for Human Faces*. PhD thesis, The University of Utah. Available at: `https://apps.dtic.mil/sti/tr/pdf/ADA038695.pdf` Acesso em: 6 ago. 2021.

Parke, F. I. (1982). Parameterized models for facial animation. *IEEE Computer Graphics and Applications*, 2(9):61–68. DOI: 10.1109/MCG.1982.1674492.

Parke, F. I. and Waters, K. (2008). *Computer Facial Animation*. A K Peters/CRC Press, Boca Raton, FL, 2 edition. DOI: 10.1201/b10705.

Pham, H. X., Cheung, S., and Pavlovic, V. (2017). Speech-driven 3d facial animation with implicit emotional awareness: A deep learning approach. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2328–2336. DOI: 10.1109/CVPRW.2017.287.

Pighin, F. and Lewis, J. P. (2006). Facial motion retargeting. In *ACM SIGGRAPH 2006 Courses (SIGGRAPH '06)*, pages 2–es, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/1185657.1185842.

Ping, Y. *et al*. (2013). Computer facial animation: A review. *International Journal of Computer Theory and Engineering*, 5(6):658–662. DOI: 10.7763/IJCTE.2013.V5.770.

Richard, A., Zollhöfer, M., Wen, Y., de la Torre, F., and Sheikh, Y. (2021). Meshtalk: 3d face animation from speech using cross-modality disentanglement. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1153–1162. DOI: 10.1109/ICCV48922.2021.00121.

Rogers, Y., Sharp, H., and Preece, J. (2019). *Interaction Design: Beyond Human–Computer Interaction*. John Wiley & Sons, Chichester, 5 edition. Book.

Sauro, J. (2015). SUPR-Q: a comprehensive measure of the quality of the website user experience. *Journal of Usability Studies*, 10(2):68–86. Available at: `https://dl.acm.org/doi/10.5555/2817315.2817317`.

Schrepp, M., Hinderks, A., and Thomaschewski, J. (2017). Construction of a benchmark for the User Experience Questionnaire (UEQ). *International Journal of Interactive Multimedia and Artificial Intelligence*, 4(4):40–44. DOI: 10.9781/ijimai.2017.445.

Schrepp, M., Sandkühler, H., and Thomaschewski, J. (2021). How to create short forms of UEQ+ based questionnaires? In *Mensch und Computer 2021-Workshopband*. Gesellschaft für Informatik e.V.. DOI: 10.18420/muc2021-mci-ws01-230.

Schrepp, M. and Thomaschewski, J. (2019). Design and validation of a framework for the creation of user experience questionnaires. *International Journal of Interactive Multimedia and Artificial Intelligence*, 5(3):88–95. DOI: 10.9781/ijimai.2019.06.006.

Stanton, N. A. (2006). Hierarchical task analysis: Developments, applications, and extensions. *Applied Ergonomics*, 37(1):55–79. DOI: 10.1016/j.apergo.2005.06.003.

Sturman, D. J. (1994). A brief history of motion capture for computer character animation. In *Proceedings of the International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, New York, NY, USA. Association for Computing Machinery. Available at: `https://www.scribd.com/doc/248757741/a-brief-history-of-motion-capture-for-computer-chara`
Acesso em: 7 dez. 2020.

Sze, V. *et al*. (2017). Efficient processing of deep neural networks: A tutorial and survey. *arXiv preprint arXiv:1703.09039*. DOI: 10.48550/arXiv.1703.09039.

Taylor, S., Kim, T., Yue, Y., Mahler, M., Krahe, J., Rodriguez, A. G., Hodgins, J., and Matthews, I. (2017). A deep learning approach for generalized speech animation. *ACM Trans. Graph.*, 36(4). DOI: 10.1145/3072959.3073699.

Terzopoulos, D. and Waters, K. (1990). Physically-based facial modelling, analysis, and animation. *Journal of Visualization and Computer Animation*, 1(2):73–80. DOI: 10.1002/vis.4340010208.

Tolstikhin, I. O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., *et al*. (2021). Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272. Available at: `https://proceedings.neurips.cc/paper_files/paper/2021/file/cba0a4ee5ccd02fda0fe3f9a3e7b89fe-Paper.pdf`.

Tschang, T. F. and Goldstein, A. (2004). Production and political economy in the animation industry: why insourcing and outsourcing occur. In *Proceedings of the DRUID Summer Conference*, Copenhagen, Denmark. Copenhagen Business School. Available at: `https://ink.library.smu.edu.sg/lkcsb_research/2853/` Acesso em: 7 dez. 2025.

Villar, O. (2014). *Learning Blender: A Hands-On Guide to Creating 3D Animated Characters*. Addison-Wesley Professional, Boston, MA, 1 edition. Book.

Wang, L. *et al*. (2022). FaceVerse: a fine-grained and detail-controllable 3D face morphable model from a hybrid dataset. *arXiv preprint arXiv:2203.14057*. DOI: 10.48550/arXiv.2203.14057.

Xing, J., Xia, M., Zhang, Y., Cun, X., Wang, J., and Wong, T.-T. (2023). Codetalker: Speech-driven 3d facial animation with discrete motion prior. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12780–12790. DOI: 10.1109/CVPR52729.2023.01229.

Zhang, H. *et al*. (2025). GSmoothFace: Generalized smooth talking face generation via fine grained 3D face guidance. *IEEE Transactions on Visualization and Computer Graphics*, 31(10):8231–8242. DOI: 10.1109/TVCG.2025.3566382.

Zheng, M. *et al*. (2025). ImFace++: A sophisticated nonlinear 3D morphable face model with implicit neural representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(2):994–1012. DOI: 10.1109/TPAMI.2024.3480151.

Zhou, Y., Xu, Z., Landreth, C., Kalogerakis, E., Maji, S., and Singh, K. (2018). Visemenet: audio-driven animator-centric speech animation. *ACM Trans. Graph.*, 37(4). DOI: 10.1145/3197517.3201292.

Zhu, C. and Joslin, C. (2024a). A facial motion retargeting pipeline for appearance agnostic 3D characters. *Computer Animation and Virtual Worlds*, 35(6):e70001. DOI: 10.1002/cav.70001.

Zhu, C. and Joslin, C. (2024b). A review of motion retargeting techniques for 3D character facial animation. *Computers & Graphics*, 123. DOI: 10.1016/j.cag.2024.104037.