Laboratório VISGRAF

Instituto de Matemática Pura e Aplicada

INTERACT-NET: Interactive Interfaces for Multimedia Machine Learning

Alberto Kopiler, Luiz Velho (supervisor)

Technical Report TR-24-05 Relatório Técnico

May - 2024 - Maio

INTERACT-NET: INTERACTIVE INTERFACES FOR MULTIMEDIA MACHINE LEARNING

TECHNICAL REPORT

AUTHOR: ALBERTO ARKADER KOPILER

ADVISOR: LUIZ VELHO

MAY 2024

VISGRAF/IMPA

SUMMARY

1 INTRODUCTION 5
2 MOTIVATION 7
3 REQUISITES 7
3.1 INTERACTIVITY 7
3.2 WEB (SERVERLESS) 7
3.3 2D/3D 7
3.4 MOBILE 7
3.5 FAST 7
3.6 USER FRIENDLY 8
4 TOOLS 8
4.1 WEB Interfaces 8
4.2 Javascript 9
4.2.1 Javascript for 2D and 3D graphics 9
4.2.2 Javascript for Machine Learning and Computer Vision 11
4.3 Libraries 13
5 RELATED WORK 14
5.1 DragDiffusion 14
5.2 FreeDrag 15
5.3 Drag Your GAN 15
5.4 RITM 16
5.5 SAM 16
5.6 EditGan 17
5.7 UserCotrollablelLT 18
5.8 SERF 19
5.9 Observable Notebooks 19
5.9.1 Interactively Assessing Disentanglement in GANs 20
5.9.2 Machine Learning in The Browser 21
5.9.3 Drawings to Human 21
5.9.4 Visualization in Deep Learning 22
5.9.5 Interactive Visualization of Convolutional Neural Networks 23
5.9.6 Background Position Scrubber 23
5.10 Peering Inside the Black Box 24
6 USE CASES 24

6.1 FACE MORPHING 24
7 EXPERIMENTS 33
7.1 Face Morphing 33
7.2 Face LandMarks Detection and Visualization 33
7.3 Face LandMarks Correspondence and Editing (Interaction) 34
8 CONCLUSION AND FUTURE WORK 35
9 BIBLIOGRAPHY 35
10 APPENDIX 38
10.1 TOP USE CASES FOR FACIAL LANDMARK DETECTION 38
10.1.1 Classification of Algorithms 38
10.1.2 Common 7 Use Cases for Facial Landmark Detection 39
10.1.3 DLIB Library 39

Figure Index

Figure 1 - Face image landmarks with clmtrackr. 13
Figure 3 - Face image with MediaPipe Face Mesh drawn on top. 13
Figure 4 - DragDiffusion greatly improves the applicability of interactive point-based editing. Given an input image, the user clicks handle points (red), target points (blue), and draws a mask specifying the editable region (brighter area). 15
Figure 5 - The comparison between the feature-centric FreeDrag and point-based DragGAN. Given an image input, users can assign handle points (red points) and target points (blue points) to force the semantic positions of the handle points to reach corresponding target points, and optional mask can also be provided by users to assign editing region. 15
Figure 6 - Green and red dots denote positive and negative clicks, respectively. 16
Figure 7 - Face landmark manipulation. Compared to UserControllableLT, the DragGan method can manipulate the landmarks detected from the input image to match the landmarks detected from the target image with less matching error. 16
Figure 8 - Green and red dots denote positive and negative clicks, respectively. 16
Figure 9 - Each column shows 3 valid masks generated by SAM from a single ambiguous point prompt (green circle). 17
Figure 10 – Segmentation of a face. 18
Figure 11: Editing StyleGAN image layout using user-controllable latent transformer. As shown in (a) and (d), this method can interactively generate an image reflecting a user-specified movement direction (white arrows) via manipulation in a latent space. As shown in (b) and (c), the user can specify the locations where the user does not want to move with anchor points (blue circles). This method can also handle 3D motion with an additional key input (denoted as the “o” or “i” key), as shown in (e) and (f). 18
Figure 12 – 3D Editing with SERF. 19
Figure 13 – Examples of Deep Learning visualization and interaction in Observable platform. 20
Figure 14 – Visualization of Deep Learning factors to control content produced by a GAN. 21
Figure 15 – Choosing a template to generate a synthetic human with a text description to guide the wearing clothes. 22
Figure 16 – Visualization in Deep Learning. 22
Figure 17 – Interactive Visualization of Convolutional Neural Networks Layers. 23
Figure 18 – Visualization of Age Transition of Face Morphing Aligned with the Background. 23
Figure 19 – Neural Warping + linear blending. 25
Figure 20 – face-api.js Playground. 25
Figure 21 – real-time face mesh point cloud with Three.js and Tensorflow.js. 25
Figure 22 – Interactive Decals: original and with rotation. 26
Figure 23 – Click and Drag to Control animation (source, animation, and target). 26
Figure 24 – Interactive Ripple Mouse. 27
Figure 25 – Interactive Storm. 27
Figure 26 – Interactive Sphere. 27
Figure 27 – Real-time Face Mesh Extraction. 28
Figure 28 – Landmark detection for virtual makeover. 28
Figure 29 – Delaunay Triangulation for face morphing. 28
Figure 30 – Face morphing results. 29
Figure 31 – Media Pipe Face Mesh 438 3D face landmarks. 29
Figure 32 – 68 Face Landmark detection using Dlib. 30
Figure 33 – Screenshots of the virtual environment. (A) The user is about to place a landmark with the ray-gun. (B) The user has placed six landmarks on the 3D model. 31
Figure 34 – Screenshot of the landmark editor. 31
Figure 35 – Proposed Method for Face Recognition. 32
Figure 36 – Proposed Interface screen for simulating Face Morphing [77] 33
Figure 37 – Interface screen for viewing face Landmarks: DLIB on the left, and MEDIAPIPE on the right. 34
Figure 38 – Simulated Interface screen for editing landmarks. The coordinates of the origin and destination points can be changed interactively. 34

ABSTRACT

INTERACT-NET is a study of interactive man-machine interfaces for use in pipelines of multimedia using machine learning, image processing and computer graphics.

1 INTRODUCTION

This report is not intended to be a survey about interactivity, as it would be necessary to read and analyze hundreds of articles to do so. Therefore, we do not presume to exhaust the subject, our objective being more mundane: to resolve the “pains” of the projects currently underway at IMPA's Visgraf.

Visgraf, a computer graphics experimentation laboratory, was created more than 35 years ago and has always remained at the forefront of studies related to computer graphics, with numerous projects and articles published in the main national and international magazines and conferences, as well as master's theses and doctoral theses. In this lab, there are already projects that work intensely in the analysis and synthesis of both 2D and 3D images and, in their representation, both explicitly and implicitly. More recently, even before the boom in generative artificial intelligence, it was already offered courses in 3D computer graphics and image processing in which artificial intelligence was and is an integral part of the algorithmic part and, in its strong mathematical foundation, showing new ways of carrying out generative functionalities, faster and more efficiently. This fusion of computer graphics, image processing, and artificial intelligence has led to the democratization of the use of computer graphics, as can be seen by the increasing use of image generation tools using generative artificial intelligence, such as DALL-E, Stable Diffusion, Imagen and Midjourney, just to name the most famous to date.

If we approach from a multimodal point of view, that is, not restricting the study to just 2D and 3D images, we can add 1D signals such as sound, 3D/4D such as video and images with depth and alpha channel and even text, given the tendency to create pipelines in which, for example, you enter text describing an image and obtain an image as output and vice versa, or you enter one or more images and obtain a video as output (such as SORA tools, Runway Gen-2). This “phenomenon” is called the “text-to-anything” transformation or, more generally, the “everything-to-anything” transformation). Behind this phenomenon are large language models (LLMs), which apply attention techniques to text (transformers and embeddings), which can be extended to images (vision transformers), as well as deep machine learning models such as autoencoders, GANS (generative adversarial networks) and diffusion. Segmentation models such as SAM (META's Segment Anything Model) are also useful, as well as obtaining image features (key point detection) (Surf - Speeded Up Robust Features, Sift - scale-invariant feature transform, ORB - Oriented FAST and Rotated BRIEF, Harrys Corner, edge detection), and/or face landmarks (DLIB – 68 unique features, MEDIAPIPE – 468 3D features), depending on the application.

Another fundamental area of computer graphics is interactive computer graphics [1]. Graphic input and output devices have evolved, as have operating systems (Windows and Linux with a graphical interface, as well as specific ones for mobile devices, such as IOS and Android). Following this trend of innovation, voice-activated personal assistants that synthesize speech have become more efficient with machine learning. In other words, pipelines that receive voice input, recognize it and transform it into text, which in turn activate tasks related to generating images, video, or executing some tasks as if it were an agent. The ability of a large language model to understand what the text (prompt) presupposes in each context allows the creation of "menuless" interfaces, that is, without explicit options. This fact makes it possible to generate dynamic interfaces, and the code generation capacity of these models makes it possible to generate new functionalities and even improve existing ones, making these interfaces very powerful, as they can be self-generative.

Additionally, we need to take into consideration the platform on which our applications will run. Mobile devices have converged on merging the keyboard with the screen using touchscreens. Additionally, the use of voice recording instead of the keyboard is growing. The number of mobile devices is in the billions, while the use of graphics stations is in the millions. For example, some questions can be asked: On which platform do we want our application to run? Mobile or Desktop? (A possible answer could be either, correct?) Do you want an application installed on your cell phone or an interface that runs on any browser?

Current mobile devices are versatile: they have a camera (with very high resolution), many have more than one lens (some three), many have sensors (accelerometers, gyroscope, and barometer) and LiDAR (Light Detection and Ranging), and last generation graphics processing units. Then it's expected that local machine learning execution capabilities will emerge, as computer graphics have been a reality for some time. Therefore, interactivity such as shaking the cell phone, clapping your hands to activate a function, or recognizing a gesture from the image or video must be considered in interactivity.

If we broaden the scope, a major catalyst for interactivity in computer graphics is the area of games. The idea of using a wireless controller (Wii) ended up revolutionizing the gaming industry, as well as the multi-user game industry. Other related areas are Virtual Reality and Augmented Reality.

So, this work assumes that we are dealing with multimodal information, that is, multimedia, using machine learning, computer graphics, and image processing interactively, that is, using interactive computer graphics resources.

The progressive visualization of learning stages, as well as the graphical visualization of the layer outputs (in the case of artificial neural networks), as well as the possibility of interactively editing parameters or the image itself (or features of that image), are the objectives of this study, but in an applied way to support the projects.

The methodology adopted is to start by specifying requirements and, based on case studies associated with the projects currently underway at Visgraf and "their pain points", analyze the tools available to meet this demand incrementally.

You should avoid the idea of "reinventing the wheel", but rather seek "on the shelf" frameworks and solutions, preferably open source.

2 MOTIVATION

The motivation for this work is to provide interactive human-machine interfaces that allow the integration of Visgraf multimedia projects. That is, designing interaction for multimedia analysis and synthesis pipeline (e.g. 2D and 3D images) using machine learning, image processing, and computer graphics.

3 REQUISITES

The desirable requirements are the following: interactivity, web-oriented (preferably “serverless”), suitable for 2D but also 3D, mobile (mobility), fast and easy to use. Further details of each of the requirements are given below.

3.1 INTERACTIVITY

One of the key requirements is interactivity. The user should interact at the beginning or during the learning process to achieve a better result by adjusting, for instance one or more face landmarks, or choosing a desired target point or limiting an area of interest. The type or types of interactions will depend on the specificity of the application.

3.2 WEB (SERVERLESS)

The build application should run on the web browser. Preferably it should do most of the processing on the client, so that the user will have a better experience. In this way, we can also call this requirement as “serverless”. Of course, that deep learning will need most of the time a server to process the heavy load, but the interactive part should desirably be processed on the client side.

3.3 2D/3D

We will begin with experiments in two dimensions, but we have also to consider three dimensions as there are already 3D projects at Visgraf.

3.4 MOBILE

If your application goes mobile it will have more views and users than if it were stuck in a desktop.

3.5 FAST

If your application takes too long to respond, probably it will not get the user attention or addiction. Of course, that deep learning will need time to process, but you should build your application interface in a way that the user is always aware and in control. The faster the interaction the better.

3.6 USER FRIENDLY

When conceiving the user interface, you should have in mind that simplicity and easy navigability are key concepts.

4 TOOLS

To verify the compliance with requirements and use cases, some tools were sought for evaluation. They can be divided into four groups: web interfaces, JavaScript language, and libraries.

4.1 WEB Interfaces

Here we list some important general web frameworks that can be used to interact with the images in a machine learning pipeline.

a) **Gradio** [3] is the fastest way to demo and share your machine learning model with a friendly web interface so that anyone can use it, anywhere! Interface is Gradio's main high-level class and allows you to create a web-based GUI demo around a machine learning model (or any Python function) in a few lines of code. You must specify three parameters: (1) the function to create a GUI for (2) the desired input components and (3) the desired output components. Additional parameters can be used to control the appearance and behavior of the demo. You can use the HuggingFace Community to deploy, manage, and share your app.

b) **Flutter** [4] is an open-source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase. Flutter transforms the development process. Build, test, and deploy beautiful mobile, web, desktop, and embedded experiences from a single codebase.

c) **Dash** [5] is an open-source framework for building data visualization interfaces. Released in 2017 as a Python library, it's grown to include implementations for R, Julia, and F#. Dash helps data scientists build analytical web applications without requiring advanced web development knowledge.

d) **Streamlit** [6] lets you transform Python scripts into interactive web apps in minutes, instead of weeks. Build dashboards, generate reports, or create chat apps. Once you've created an app, you can use the Streamlit Community Cloud platform to deploy, manage, and share your app.

e) **Django** [7] is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

a) **Taipy** [48] Build Python Data & AI web applications from simple pilots to production-ready web applications in no time. No more compromise on performance, customization, and scalability. Taipy enhances performance with caching control of graphical events, optimizing rendering by selectively updating graphical components only upon interaction.

- Python-Based UI Framework: Taipy is designed for Python users, particularly those working in AI and data science. It allows them to create full stack applications without needing to learn additional skills like HTML, CSS, or JavaScript.
- Pre-Built Components for Data Pipelines: Taipy includes pre-built components that allow users to interact with data pipelines, including visualization and management tools.
- Scenario and Data Management Features: Taipy offers features for managing different business scenarios and data, which can be useful for applications like demand forecasting or production planning.
- Version Management and Pipeline Orchestration: It includes tools for managing application versions, pipeline versions, and data versions, which are beneficial for multi-user environments.

4.2 Javascript

JavaScript is one of the most popular programming languages in the world. It powers millions of websites today, and it has attracted droves of developers and designers to build features for the web. If you’re new to programming, JavaScript is easily one of the best programming languages to get under your belt [32].

Here we list some JavaScript libraries that can be used to build interactively two categories of user interfaces: 2D and 3D graphics and machine learning. These libraries can be used individually or merged with other JavaScript libraries or even with the one of the above-mentioned web interfaces.

4.2.1 Javascript for 2D and 3D graphics

a) **React** [8] is a JavaScript library for building user interfaces. You can build web and native user interfaces out of individual pieces called components written in JavaScript. React has been designed from the start for gradual adoption, and you can use as little or as much React as you need. Whether you want to get a taste of React, add some interactivity to a simple HTML page, or start a complex React-powered app. React makes it painless to create interactive UIs. Design simple views for each state in your application and React will efficiently update and render just the right components when your data changes.

b) **Next.js** [9] is a React framework for building full-stack web applications. You use React Components to build user interfaces, and Next.js for additional features and optimizations. Used by some of the world's largest companies, Next.js enables you to create high-quality web applications with the power of React components. Under the hood, Next.js also abstracts and automatically configures tooling needed for React, like bundling, compiling, and more. This allows you to focus on building your application instead of spending time with configuration. Whether you're an individual developer or part of a larger team, Next.js can help you build interactive, dynamic, and fast React applications.

c) **Three.js** [10] is a 3D JavaScript library that tries to make it as easy as possible to get 3D content on a webpage. Three.js is often confused with WebGL since often, but not always, three.js uses WebGL to draw 3D. WebGL is a very low-level system that only draws points, lines, and triangles. To do anything useful with WebGL generally requires quite a bit of code and that is where three.js comes in. It handles stuff like scenes, lights, shadows, materials, textures, 3d math, all things that you'd have to write yourself if you were to use WebGL directly.

d) **D3** (or **D3.js**) [11] is a free, open-source JavaScript library for visualizing data. Its low-level approach built on web standards offers unparalleled flexibility in authoring dynamic, data-driven graphics.

e) **P5.js [12]** is a JavaScript library for creative coding, with attention to making code accessible and inclusive for artists, designers, educators, beginners, and anyone else. It is a free and open-source library, that is, it can be accessible to everyone. Using the metaphor of a sketch, p5.js features a full set of drawing functionality. However, you're not limited to your drawing canvas. You'll consider your whole browser page as your sketch, including HTML5 objects for text, input, video, webcam, and sound.

f) **Luma AI's Three.js and R3F Gaussian Splatting Library** [13] is a JavaScript library developed by Luma.ai to interface to Three.js and render Gaussian Splatting.

g) **Luma WebGL Library** [17] luma-web is a npm package for rendering photoreal interactive scenes captured by the Luma app. It includes LumaSplatsWebGL, which is a WebGL-only gaussian splatting implementation designed to be integrated with 3D frameworks, and LumaSplatsThree, which is a Three.js implementation that uses LumaSplatsWebGL under the hood.

h) **React Three Fiber** [18] is an interface of React to use Three.js.

i) **Vue** [19] is the progressive JavaScript framework. It's an approachable, performant, and versatile framework for building web user interfaces. Vue (pronounced /vjuː/, like **view**) is a JavaScript framework for building user interfaces. It's built on top of standard HTML, CSS, and JavaScript and provides a declarative, component-based programming model that helps you efficiently develop user interfaces of any complexity. Vue is a framework and ecosystem that covers most of the common features needed in frontend development. But the web is extremely diverse - the things we build on the web may vary drastically in form and scale. With that in mind, Vue is designed to be flexible and incrementally adoptable.

j) **Svelte** [20] is a tool for building web applications. Like other user interface frameworks, it allows you to build your app declaratively out of components that combine markup, styles and behaviors. These components are compiled into small, efficient JavaScript modules that eliminate overhead traditionally associated with UI frameworks.

k) **AngularJS** [28] is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. AngularJS's data binding and dependency injection eliminate much of the code you would otherwise have to write. And it all happens within the browser, making it an ideal partner with any server technology. AngularJS is what HTML would have been, had it been designed for applications. HTML is a great declarative language for static documents. It does not contain much in the way of creating applications, and as a result building web application is an exercise in *what do I have to do to trick the browser into doing what I want?*

l) **Node.js** [32] For its first 20 years, JavaScript was used mainly for client-side scripting. Since JavaScript could be used only within the <script> tag, developers had to work in multiple languages and frameworks between the front-end and back-end components.

Later came Node.js, which is a run-time environment that includes everything required to execute a program written in JavaScript.
Node.js is a single-threaded, open-source, cross-platform runtime environment for building fast and scalable server-side and networking applications. It runs on the V8 JavaScript runtime engine, and it uses event-driven, non-blocking I/O architecture, which makes it efficient and suitable for real-time applications.

4.2.2 Javascript for Machine Learning and Computer Vision

a) **TensorFlow.js** [14] is a library for machine learning in JavaScript. Develop ML models in JavaScript and use ML directly in the browser or in Node.js. **TensorFlow.js** is a general purpose, WebGL-accelerated numeric platform for JavaScript. It brings highly performant machine learning building blocks to your fingertips, allowing you to train neural networks in a browser or run pre-trained models in inference mode. You can see an example of use here [30].
b) **Transformers.js** [37] is the state-of-the-art Machine Learning for the web. Run Transformers directly in your browser, with no need for a server! Transformers.js is designed to be functionally equivalent to Hugging Face's transformers python library, meaning that you can run the same pretrained models using a very similar API. These models support common tasks in different modalities, such as:
   - Natural Language Processing: text classification, named entity recognition, question answering, language modeling, summarization, translation, multiple choice, and text generation.
   - Computer Vision: image classification, object detection, and segmentation.
   - Audio: automatic speech recognition and audio classification.
   - Multimodal: zero-shot image classification.

   Transformers.js uses ONNX Runtime to run models in the browser. The best part about it, is that you can easily convert your pretrained PyTorch, TensorFlow, or JAX models to ONNX using Optimum.

c) **ml5.js** [44] aims to make machine learning approachable for a broad audience of artists, creative coders, and students. The library provides access to machine learning algorithms and models in the browser, building on top of TensorFlow.js with no other external dependencies.
d) **Keras.js** [38] Run Keras models in the browser, with GPU support provided by WebGL 2. Models can be run in Node.js as well, but only in CPU mode. Because Keras abstracts a great number of frameworks as backends, the models can be trained in any backend, including TensorFlow, CNTK, etc.
e) **OpenCV.js** [39], Open Source Computer Vision (OpenCV), is an open-source library initially written in C++ but later came to support many other languages such as Python, Java, and JavaScript. It's used for image processing, feature extraction & detection, object detection, face detection, camera calibration, machine learning algorithms, and more. It's a computer vision and image processing library and has an outstanding reputation among developers. It's widely adopted in healthcare, robotics, surveillance, entertainment, research, automotive, etc. The most exciting part of it for web developers of it is that it's available in JavaScript. Even though some features from the original implementation are missing in OpenCV.js it's still very capable and powerful.

f) **Synaptic.js** [41] is the JavaScript architecture-free neural network library for node.js and the browser. A significant feature of this library is its ability to build and train any first- or second-order neural network architecture due to its architecture-less algorithm and pre-made structure. Synaptic.js can also import or export networks to JSON as a standalone function so you can connect to other networks.

g) **ConvNet.js** [43] is a JavaScript library for training Deep Learning models (Neural Networks) entirely in your browser. Open a tab and you're training. No software requirements, no compilers, no installations, no GPUs, no sweat. It's designed specifically for training deep learning models and working with neural networks. The most important feature of this library is that it is completely dependent on browsers, so any other special software like GPU, compilers are not required. ConvNetJS also supports Node.js. ConvNetJS consists of common neural network modules that have fully connected layers and non-linearities. This library can formulate and solve neural networks using simple JavaScript, offering support for some common network modules.

h) **Neuro.js** [42] is a renowned machine learning JavaScript library for training and developing ML models and can be easily deployed in the web browser or Node.js. Furthermore, it supports online learning, multi-label classification, as well as website development real-time classification and can be used to create artificial intelligence-based chatbots and assistants. Everyone should have access to simple machine learning. Practical machine learning should be simple.

i) **Brain.js** [40] is an open-source JavaScript library used to run and process neural networks. It is particularly useful for developers venturing into machine learning and would be the best option for those who are already familiar with the intricacies of JavaScript. Can be used in the browser or with Node.js. With Brain.JS, different types of networks are available for different tasks. Brain.js is a fast-processing library due to the use of GPU for calculations. Even if the GPU is not available, it falls back to pure JS and continues processing. Brain.js provides multiple implementations of neural networks and encourages creating training and running these neural networks on the server side alongside Node.js. Another advantage of this library is that you don't need to be completely familiar with neural networks to work with it. To integrate your website with these network templates, simply implement them as a function or use the JSON format.

j) **Face-api.js** [70][71] JavaScript API library for face detection and face recognition in the browser implemented on top of the tensorflow.js core API.

k) **Tracking.js** [73] JavaScript library brings different computer vision algorithms and techniques into the browser environment. By using modern HTML5 specifications, it enables you to do real-time color tracking, face detection and much more — all that with a lightweight core (~7 KB) and intuitive interface.

l) **clmtrackr** [74] is a JavaScript library for fitting facial models to faces in videos or images. It currently is an implementation of *constrained local models* fitted by *regularized landmark mean-shift*, as described in Jason M. Saragih's paper. **clmtrackr** tracks a face and outputs the coordinate positions of the face model as an array, following the numbering of the model below:

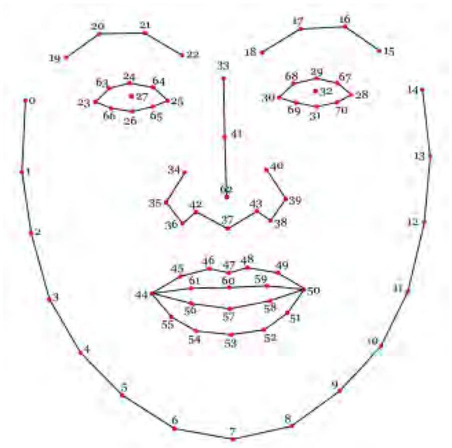


Figure 1 - Face image landmarks with clmtrackr.

4.3 Libraries

a) **DLIB** [47] contains a wide range of machine learning algorithms. All designed to be highly modular, quick to execute, and simple to use via a clean and modern C++ API. It is used in a wide range of applications including robotics, embedded devices, mobile phones, and large high-performance computing environments. There are many examples using API wrapper for Python: Face detector, alignment, landmark detection, recognition, among others.

b) **Mediapipe** [2] - A wide range of potential Machine Learning applications today rely on several fundamental baseline Machine Learning tasks. For example, both gestural navigation and sign language detectors rely on the ability of a program to identify and track human hands. Given that building something like a hand tracking model is time-consuming and resource-intensive, a developmental bottleneck exists in the creation of *all* applications that rely on hand tracking. To address this problem, Google invented **MediaPipe**. MediaPipe provides cornerstone Machine Learning models for common tasks like hand tracking, therefore removing the *same* developmental bottleneck that exists for a host of Machine Learning applications. These models, along with their excessively easy-to-use APIs, in turn streamline the development process and reduce project lifetime for many applications that rely on Computer Vision. Another useful task provided by MediaPipe is face landmarks detection and visualization.

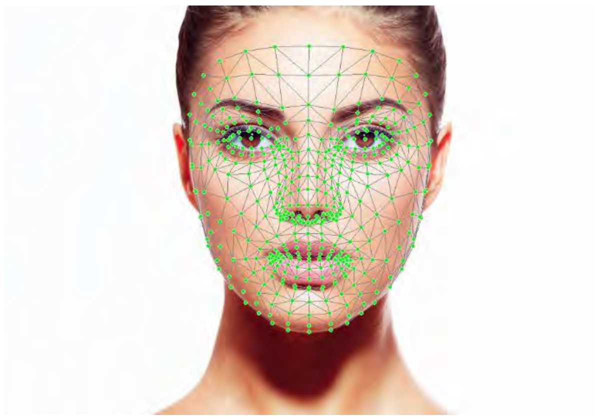


Figure 2 - Face image with MediaPipe Face Mesh drawn on top.

c) **Tkinter** [49] is the de facto way in Python to create Graphical User interfaces (GUIs) and is included in all standard Python Distributions. In fact, it's the only framework built into the Python standard library. This Python framework provides an interface to the Tk toolkit and works as a thin object-oriented layer on top of Tk. The Tk toolkit is a cross-platform collection of 'graphical control elements', aka widgets, for building application interfaces.

d) **PyQt** [50] is a Python binding of the cross-platform GUI toolkit Qt, implemented as a Python plug-in. PyQt is free software developed by the British firm Riverbank Computing. It is available under similar terms to Qt versions older than 4.5; this means a variety of licenses including GNU General Public License (GPL) and commercial license, but not the GNU Lesser General Public License (LGPL).[3] PyQt supports Microsoft Windows as well as various kinds of UNIX, including Linux and MacOS (or Darwin).

5 RELATED WORK

Here we present some work with interactive techniques that involve modifying images either to guide generative machine learning or to correct stages of this learning. Some works mention editing images interactively oriented to points, with the inclusion or exclusion of points. Others also mention selection by clicks, scribble, or area of interest, for 2D and 3D.

In related works, we do not address using text to generate images. But a possible interactivity could be to describe what you want (by text or voice) to modify the entire image or just part of it by marking an area of interest. This fact was due to the need to establish a scope to limit the possibilities, at least in this initial approach, to enable the generation of objectively practical products.

5.1 DragDiffusion

"DragDiffusion: Harnessing Diffusion Models for Interactive Point-based Image Editing" [25]. Accurate and controllable image editing is a challenging task that has attracted significant attention recently. Notably, DragGAN [23] is an interactive point-based image editing framework that achieves impressive editing results with pixel-level precision. However, due to its reliance on generative adversarial networks (GANs), its generality is limited by the capacity of pretrained GAN models. In this work, we extend this editing framework to diffusion models and propose a novel approach DragDiffusion. By harnessing large-scale pretrained diffusion models, we greatly enhance the applicability of interactive point-based editing on both real and diffusion-generated images. Our approach involves optimizing the diffusion latents to achieve precise spatial control. The supervision signal of this optimization process is from the diffusion model's UNet features, which are known to contain rich semantic and geometric information. Moreover, we introduce two additional techniques, namely LoRA fine-tuning and latent-MasaCtrl, to further preserve the identity of the original image. Experiments across a wide range of challenging cases (e.g., images with multiple objects, diverse object categories, various styles, etc.) demonstrate the versatility and generality of DragDiffusion.

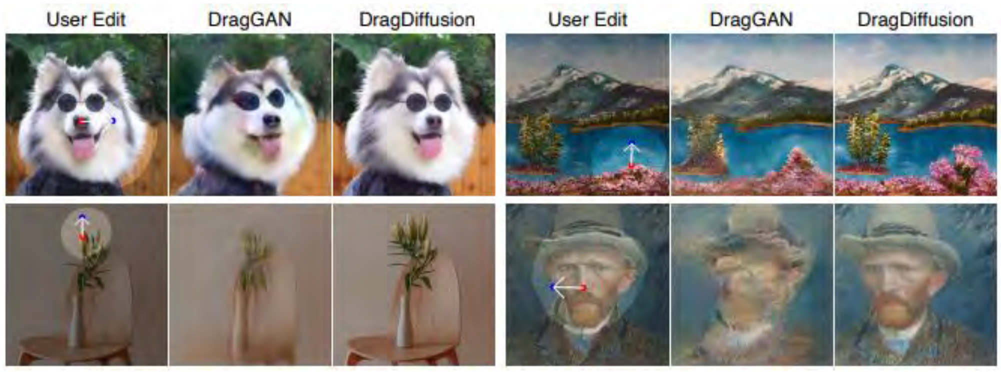


Figure 3 - DragDiffusion greatly improves the applicability of interactive point-based editing. Given an input image, the user clicks handle points (red), target points (blue), and draws a mask specifying the editable region (brighter area).

5.2 FreeDrag

FreeDrag [24] is a Feature Dragging for Reliable Point-based Image Editing Point. That is, tracking is not what you need for Interactive point-based image editing.

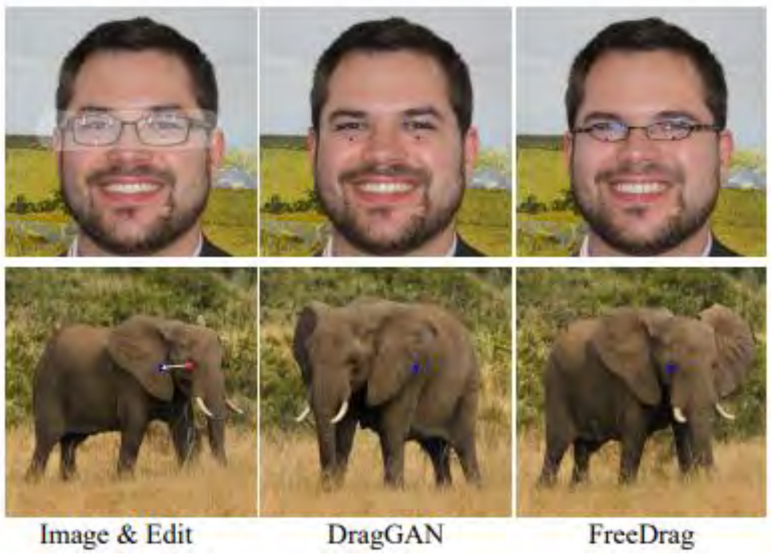


Figure 4 - The comparison between the feature-centric FreeDrag and point-based DragGAN. Given an image input, users can assign handle points (red points) and target points (blue points) to force the semantic positions of the handle points to reach corresponding target points, and optional mask can also be provided by users to assign editing region.

5.3 Drag Your GAN

Drag Your GAN [23] is an Interactive Point-based Manipulation on the Generative Image Manifold. Is a way of controlling GANs, that is, to "drag" any points of the image to precisely reach target points in a user-interactive manner, as shown in Figure 5. To achieve this, DragGAN, has two main components: 1) a feature-based motion supervision that drives the handle point to move towards the target position, and 2) a new point tracking approach that leverages the discriminative generator features to keep localizing the position of the handle points. Through DragGAN, anyone can deform an image with precise control over where pixels go, thus manipulating the pose, shape, expression, and layout of diverse categories such as animals, cars, humans, landscapes, etc.

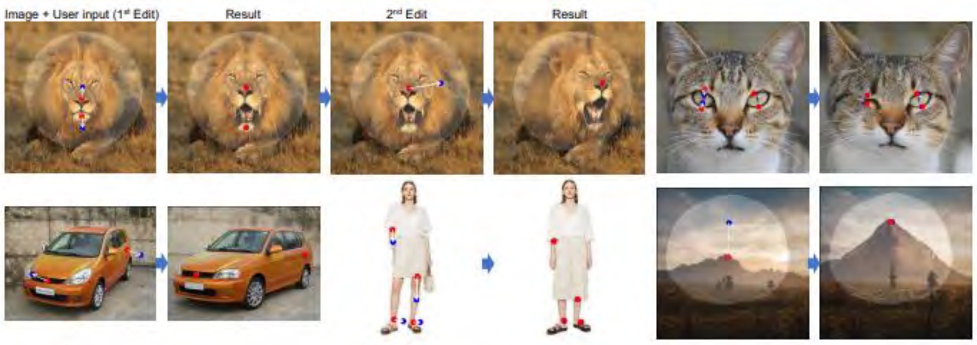


Figure 5 - Green and red dots denote positive and negative clicks, respectively.

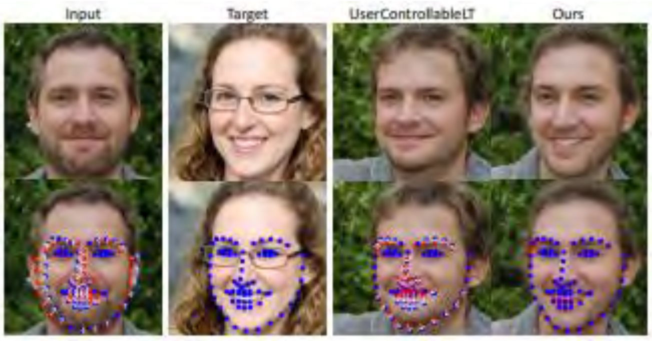


Figure 6 - Face landmark manipulation. Compared to UserControllableLT [26], the DragGan method can manipulate the landmarks detected from the input image to match the landmarks detected from the target image with less matching error.

5.4 RITM

Reviving Iterative Training with Mask Guidance for Interactive Segmentation (RITM) is a state-of-the art click-based interactive segmentation integrated into Supervisely Image Annotator [21].

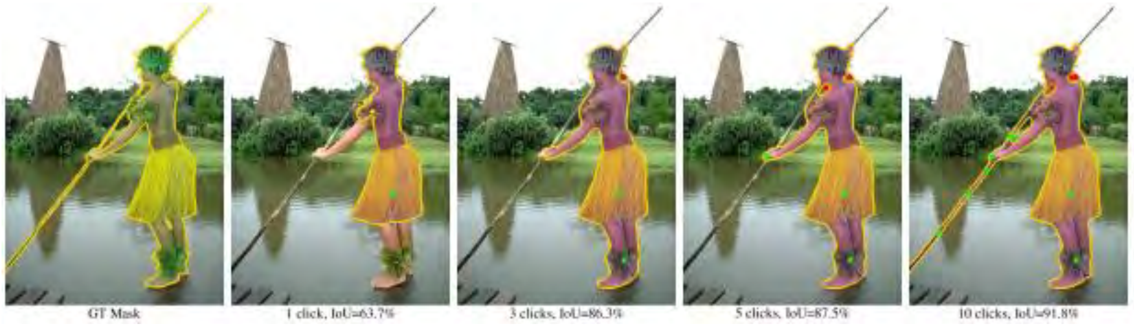


Figure 7 - Green and red dots denote positive and negative clicks, respectively.

5.5 SAM

Segment Anything Model (SAM) [22]: a new AI model from Meta AI that can "cut out" any object, in any image, with a single click. SAM is a promptable segmentation system with zero-shot generalization to unfamiliar objects and images, without the need for additional training.

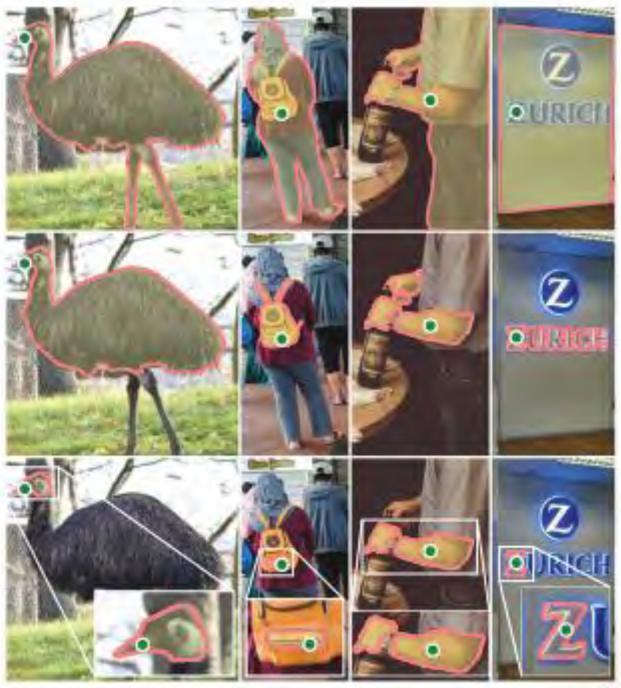


Figure 8 - Each column shows 3 valid masks generated by SAM from a single ambiguous point prompt (green circle).

5.6 EditGan

Generative adversarial networks (GANs) have recently found applications in image editing. However, most GAN-based image editing methods often require large-scale datasets with semantic segmentation annotations for training, only provide high level control, or merely interpolate between different images. There, they propose EditGAN [27], High-Precision Semantic Image Editing, a novel method for high-quality, high-precision semantic image editing, allowing users to edit images by modifying their highly detailed part segmentation masks, e.g., drawing a new mask for the headlight of a car. EditGAN builds on a GAN framework that jointly models images and their semantic segmentations, requiring only a handful of labeled examples – making it a scalable tool for editing. Specifically, it's embedded an image into the GAN's latent space and perform conditional latent code optimization according to the segmentation edit, which effectively also modifies the image. To amortize optimization, they find "editing vectors" in latent space that realize the edits. The framework allows them to learn an arbitrary number of editing vectors, which can then be directly applied on other images at interactive rates. They experimentally show that EditGAN can manipulate images with great level of detail and freedom, while preserving full image quality. They can also easily combine multiple edits and perform plausible edits beyond EditGAN's training data.

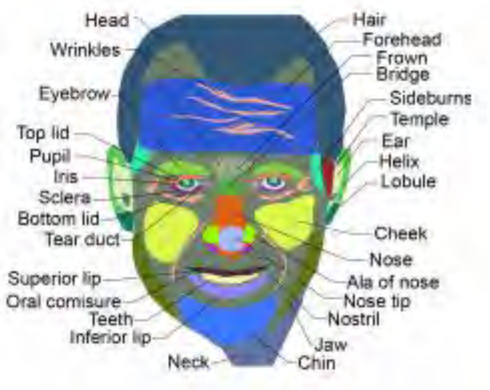


Figure 9 – Segmentation of a face.

5.7 UserCotrollablelLT

Latent space exploration is a technique that discovers interpretable latent directions and manipulates latent codes to edit various attributes in images generated by generative adversarial networks (GANs). However, in previous work, spatial control is limited to simple transformations (e.g., translation and rotation), and it is laborious to identify appropriate latent directions and adjust their parameters. In this paper, we tackle the problem of editing the StyleGAN image layout by annotating the image directly. To do so, is proposed an interactive framework for manipulating latent codes in accordance with the user inputs named "User-Controllable Latent Transformer for StyleGAN Image Layout Editing", or simply UserControllablelLT [26]. In this framework, the user annotates a StyleGAN image with locations they want to move or not and specifies a movement direction by mouse dragging. From these user inputs and initial latent codes, this latent transformer based on a transformer encoder decoder architecture estimates the output latent codes, which are fed to the StyleGAN generator to obtain a result image. To train this latent transformer, it's used synthetic data and pseudo-user inputs generated by off-the-shelf StyleGAN and optical flow models, without manual supervision.

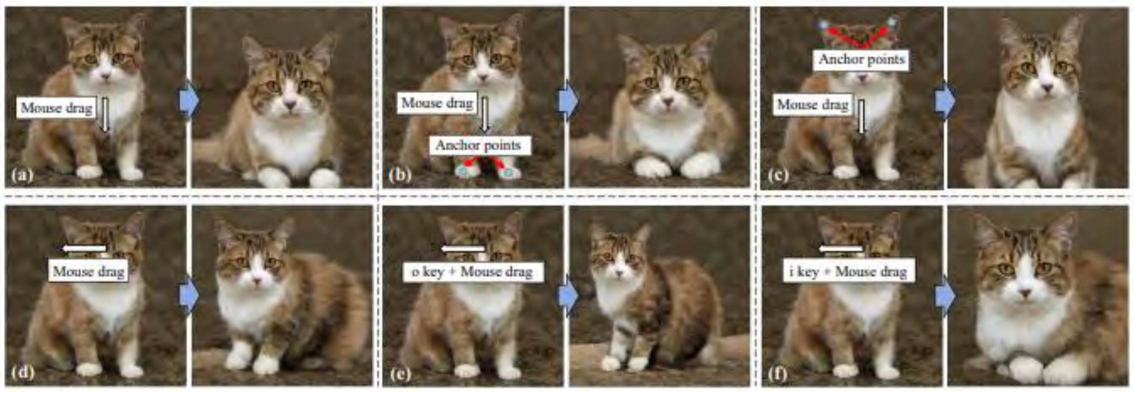


Figure 10: Editing StyleGAN image layout using user-controllable latent transformer. As shown in (a) and (d), this method can interactively generate an image reflecting a user-specified movement direction (white arrows) via manipulation in a latent space. As shown in (b) and (c), the user can specify the locations where the user does not want to move with anchor points (blue circles). This method can also handle 3D motion with an additional key input (denoted as the "o" or "i" key), as shown in (e) and (f).

5.8 SERF

Although significant progress has been made in the field of 2D-based interactive editing, fine-grained 3D-based interactive editing remains relatively unexplored. This limitation can be attributed to two main challenges: the lack of an efficient 3D representation robust to different modifications and the absence of an effective 3D interactive segmentation method. In this paper it's introduced a novel fine-grained interactive 3D segmentation and editing algorithm with radiance fields, which we refer to as SERF: Fine-Grained Interactive 3D Segmentation and Editing with Radiance Fields [46]. This method entails creating a neural mesh representation by integrating multi-view algorithms with pre-trained 2D models. Building upon this representation, it's introduced a novel surface rendering technique that preserves local information and is robust to deformation. Moreover, this representation forms the basis for achieving accurate and interactive 3D segmentation without requiring 3D supervision. Harnessing this representation facilitates a range of interactive 3D editing operations, encompassing tasks such as interactive geometry editing and texture painting. According to the authors, extensive experiments, and visualization examples of editing on both real and synthetic data demonstrate the superiority of this method on representation quality and editing ability.

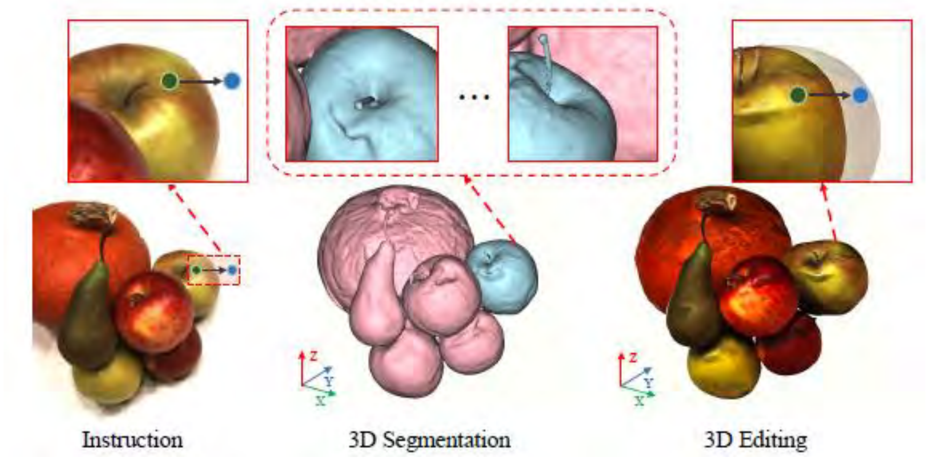
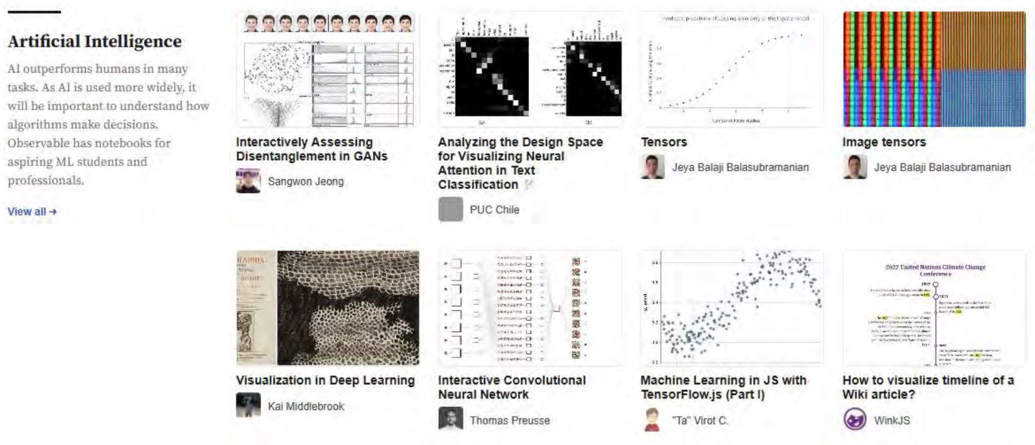


Figure 11 – 3D Editing with SERF.

5.9 Observable Notebooks

Artificial Intelligence outperforms humans in many tasks. As AI is used more widely, it will be important to understand how algorithms make decisions. Observable[1] has notebooks for aspiring ML students and professionals. [35]

[1] Observable mantains D3.js


Artificial Intelligence

AI outperforms humans in many tasks. As AI is used more widely, it will be important to understand how algorithms make decisions. Observable has notebooks for aspiring ML students and professionals.

View all →

Interactively Assessing Disentanglement in GANs — Sangwon Jeong

Analyzing the Design Space for Visualizing Neural Attention in Text Classification — PUC Chile

Tensors — Jeya Balaji Balasubramanian

Image tensors — Jeya Balaji Balasubramanian

Visualization in Deep Learning — Kai Middlebrook

Interactive Convolutional Neural Network — Thomas Preusse

Machine Learning in JS with TensorFlow.js (Part I) — "Ta" Virot C.

How to visualize timeline of a Wiki article? — WinkJS

Figure 12 – Examples of Deep Learning visualization and interaction in Observable platform.

Here is a list of these examples content:

1) Interactively Assessing Disentanglement in GANs
2) Analyzing the Design Space for Visualizing Neural Attention in Text Classification
3) Tensors
4) Image tensors
5) Visualization in Deep Learning
6) Interactive Convolutional Neural Network
7) Machine Learning in JS with TensorFlow.js (Part I)
8) How to visualize timeline of a Wiki article?
9) Simulation Results
10) Machine learning in the browser
11) Recommender systems
12) Python in Observable through Pyodide
13) Artificial intelligence systems Timeline
14) background-position Scrubber
15) Drawings to Human
16) sahajBERT - Bubbles chart
17) Pretrained models (image classification)
18) Advanced pretrained models (object detection)
19) Robotics Perturbation: Broken Joint (Rotation)
20) Multi-Agent Evaluation: Object Counting

In the next subsections, we are going to develop six of the above examples that are more related to the interactive subject.

5.9.1 Interactively Assessing Disentanglement in GANs

Generative adversarial networks (GAN) have witnessed tremendous growth in recent years, demonstrating wide applicability in many domains. However, GANs remain notoriously difficult for people to interpret, particularly for modern GANs capable of generating photo-realistic imagery. In this work we contribute a visual analytics approach for GAN interpretability, where we focus on the analysis and visualization of GAN disentanglement. Disentanglement is concerned with the ability to control content produced by a GAN along a small number of distinct, yet semantic, factors of variation. The goal of our approach is to shed insight on GAN disentanglement, above and beyond coarse summaries, instead permitting a deeper analysis of the data distribution modeled by a GAN. Our visualization allows one to assess a single factor of variation in terms of groupings and trends in the data distribution, where our analysis seeks to relate the learned representation space of GANs with attribute-based semantic scoring of images produced by GANs. Through use-cases, we show that our visualization is effective in assessing disentanglement, allowing one to quickly recognize a factor of variation and its overall quality. In addition, we show how our approach can highlight potential dataset biases learned by GANs [36].

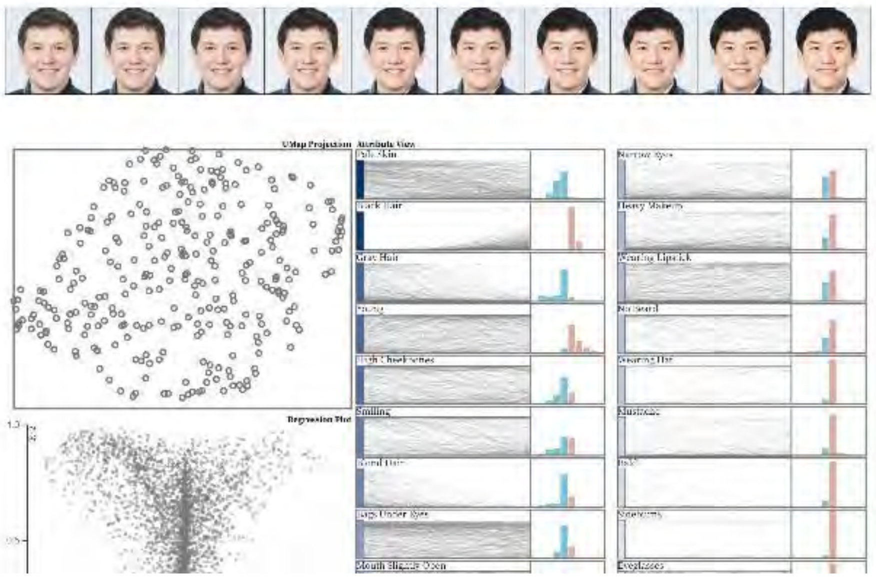


Figure 13 – Visualization of Deep Learning factors to control content produced by a GAN.

5.9.2 Machine Learning in The Browser

Using JavaScript and frameworks like Tensorflow.js is a great way to get started and learn more about machine learning [34]. Machine learning often feels like it belongs to the realm of data scientists and Python developers. However, over the past couple of years, open-source frameworks have been created to make it more accessible in different programming languages, including JavaScript.

5.9.3 Drawings to Human

This is an unofficial drawing tool to explore the generative human generator **Text2Human**. This is the original prototype that was later converted to Svelte and published here drawings-to-human.

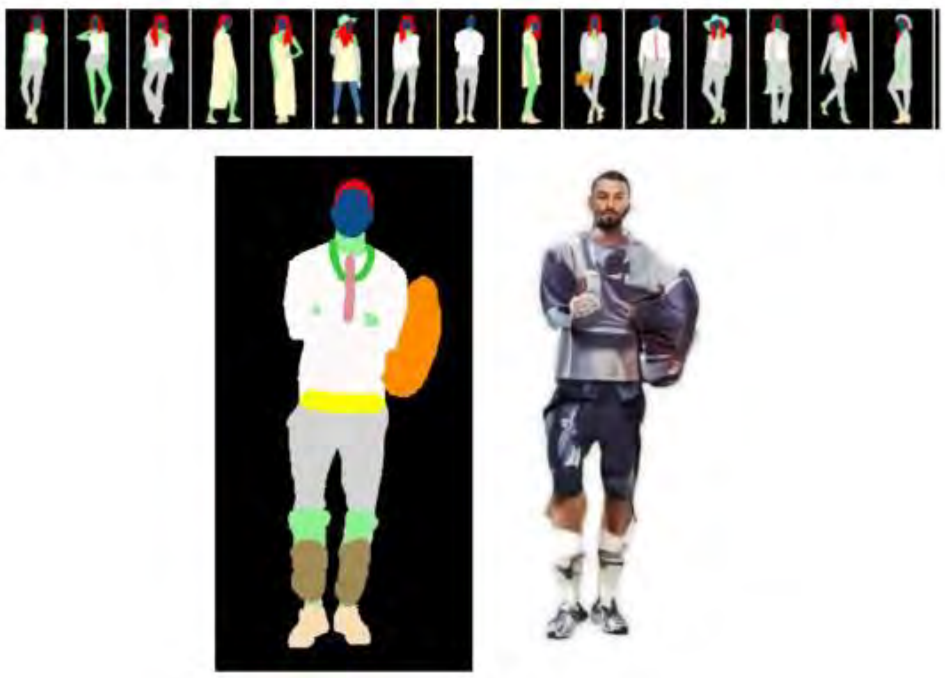
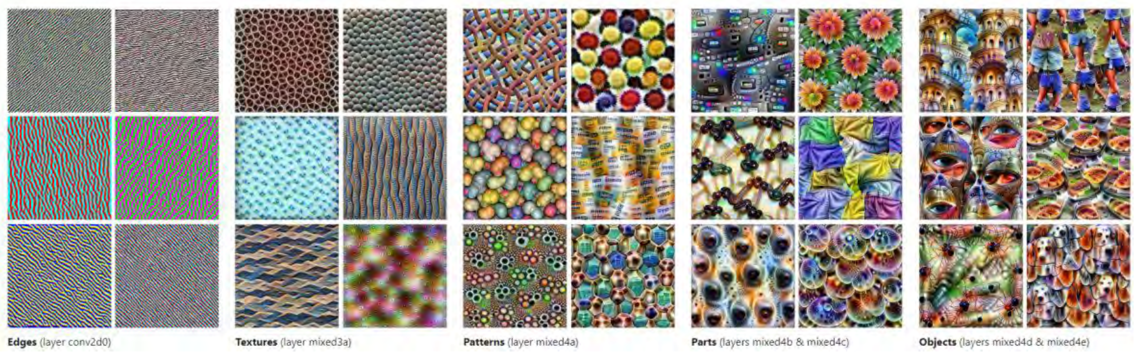


Figure 14 – Choosing a template to generate a synthetic human with a text description to guide the wearing clothes.

5.9.4 Visualization in Deep Learning

Deep learning models achieve great success on image recognition, speech processing, and language translation tasks. Yet relatively little is known about why these networks work. Perhaps more importantly, there is no easy method to identify factors that cause models to produce unfavorable results. To advance deep learning safely and ethically, we must develop tools to better understand how our models work. Visualization techniques will play a crucial role in this effort.


Edges (layer conv2d0)
Textures (layer mixed3a)
Patterns (layer mixed4a)
Parts (layers mixed4b & mixed4c)
Objects (layers mixed4d & mixed4e)
Figure 15 – Visualization in Deep Learning. Feature Visualization of GoogLeNet trained on ImageNet [75].

5.9.5 Interactive Visualization of Convolutional Neural Networks

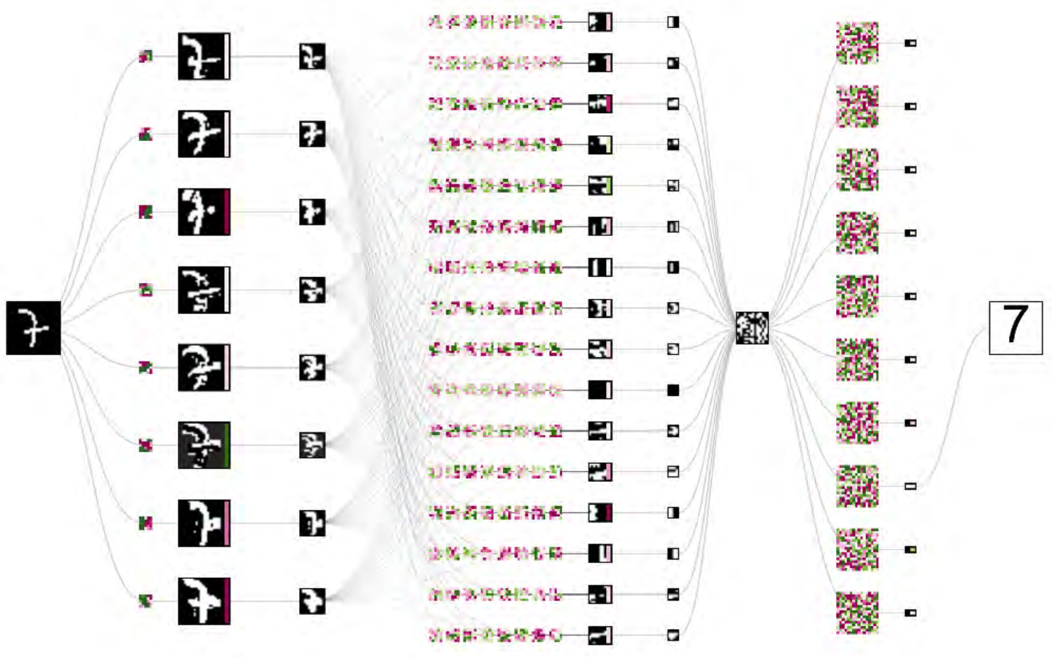


Figure 16 – Interactive Visualization of Convolutional Neural Networks Layers.

5.9.6 Background Position Scrubber

Single large image controlled by the background-position property.

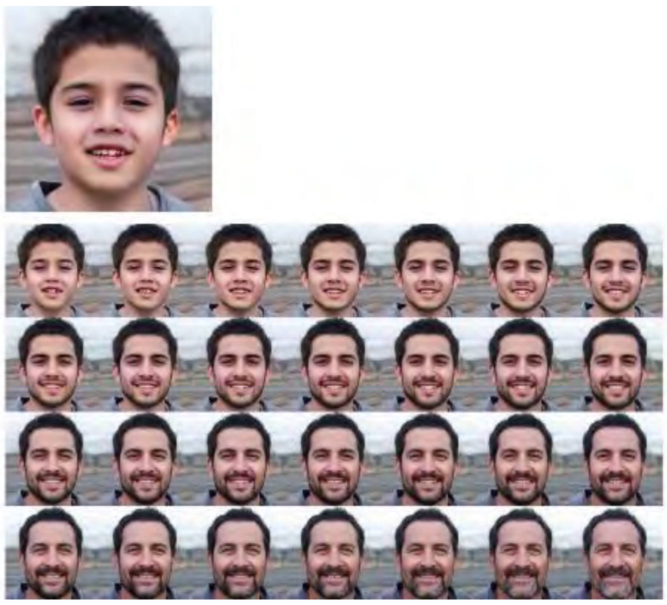


Figure 17 – Visualization of Age Transition of Face Morphing Aligned with the Background.

5.10 Peering Inside the Black Box

Artificial intelligence is advancing in leaps and bounds, but it still suffers from some major weaknesses. On the second leg of our journey through the digital world, we examine these vulnerabilities and ask: Why do they exist? And can they be overcome? [33]

6 USE CASES

Here are some case studies presented. The first one is directly related to the most recent work published by the Neural Media group: face morphing, where it will be interesting to insert interactivity in the learning stages. Then the second use case is interactivity in generative artificial intelligence[2].

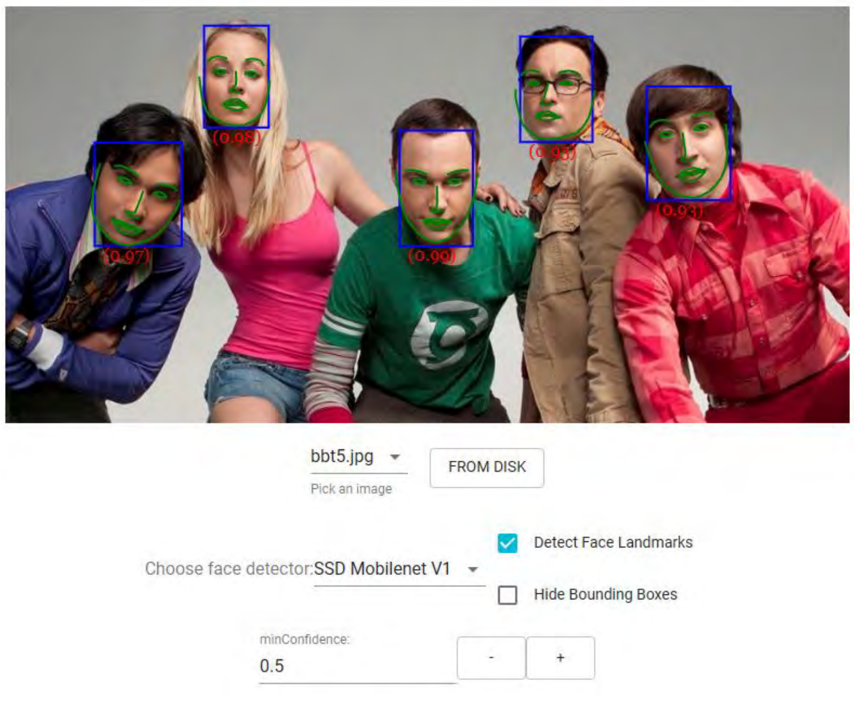
6.1 FACE MORPHING

a) "Neural Implicit Morphing of Face Images" [51] - Face morphing is a problem in computer graphics with numerous artistic and forensic applications. It is challenging due to variations in pose, lighting, gender, and ethnicity. This task consists of a warping for feature alignment and a blending for a seamless transition between the warped images. This work leverage coordinated-based neural networks to represent such warping and blending of face images. During training, it exploits the smoothness and flexibility of such networks by combining energy functionals employed in classical approaches without discretization. Additionally, this method is time-dependent, allowing a continuous warping/blending of the images. During morphing inference, we need both direct and inverse transformations of the time-dependent warping. The first (second) is responsible for warping the target (source) image into the source (target) image. This neural warping stores those maps in a single network dismissing the need for inverting them. The results of their experiments indicate that their method is competitive with both classical and generative models under the lens of image quality and face-morphing detectors. Aesthetically, the resulting images present a seamless blending of diverse faces not yet usual in the literature.

[2] Future work

Figure 18 – Neural Warping + linear blending.

b) “Face and Landmark Detection using face-api.js” [52]



bbt5.jpg

Pick an image

FROM DISK

Choose face detector: SSD Mobilenet V1

[✓ Detect Face Landmarks]

Hide Bounding Boxes

minConfidence: 0.5

Figure 19 – face-api.js Playground.

c) Real Time AI Face Landmark Detection in 20 Minutes with Tensorflow.JS and React [53]contains a video showing how to use Tensorflow.JS and React (client side) to detect face landmarks in real time.

d) Real-time 3D face mesh point cloud with Three.JS, Tensorflow.js and Typescript [54]. This article focuses on the steps needed to implement a real-time face mesh point cloud with Three.js and Tensorflow.js.

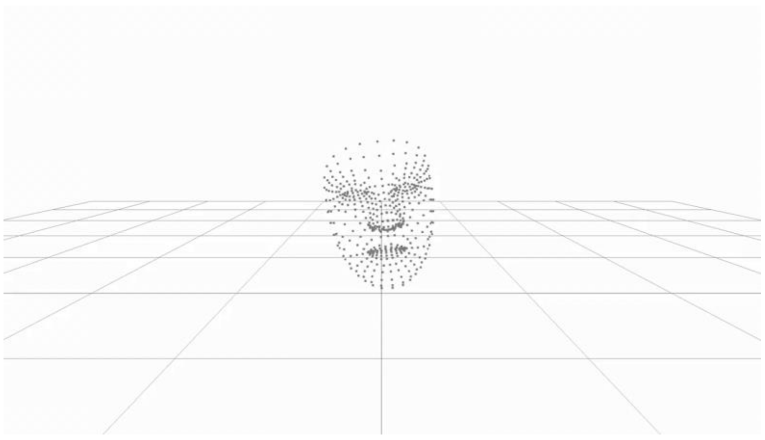


Figure 20 – real-time face mesh point cloud with Three.js and Tensorflow.js.

e) Interactive decals using three.js [55]

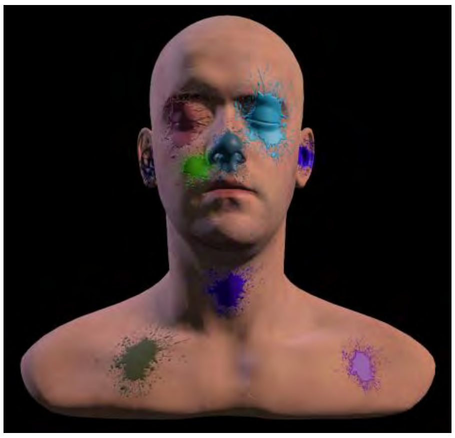
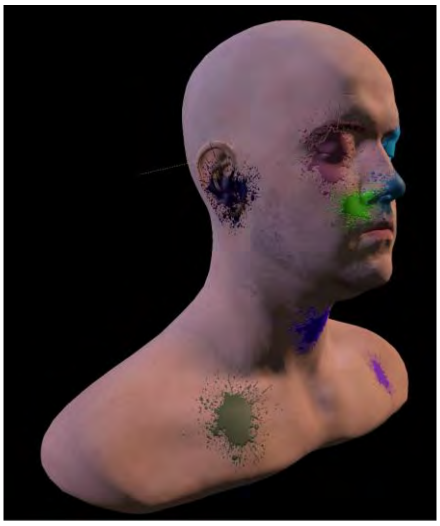



Figure 21 – Interactive Decals: original and with rotation.

f) Click and drag to control animation [56]

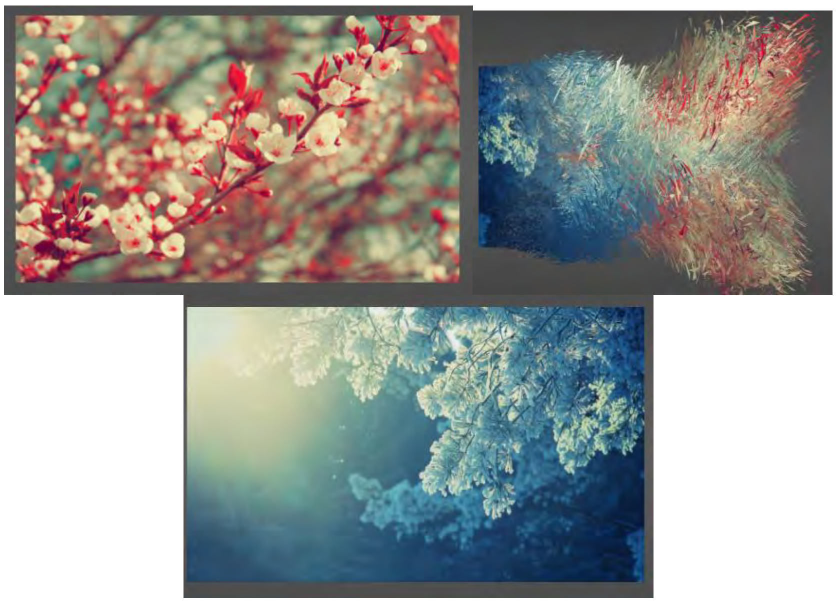


Figure 22 – Click and Drag to Control animation (source, animation, and target).

g) "56 Three JS Examples - Collection of three.js (Javascript 3D library) code examples." [56]

Figure 23 – Interactive Ripple Mouse.

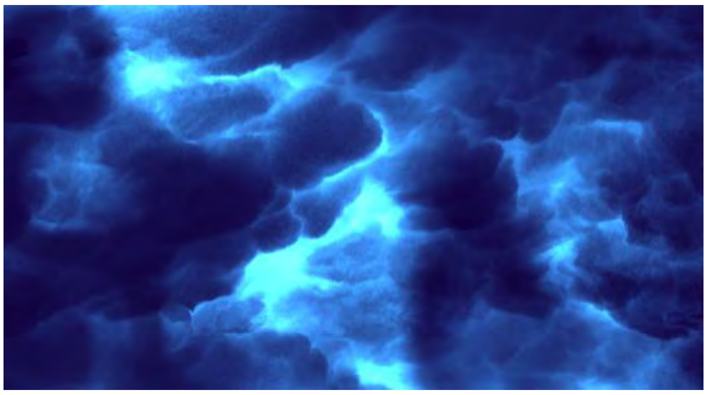


Figure 24 – Interactive Storm.

h) AI Assistant | Three.js interactive sphere [58]

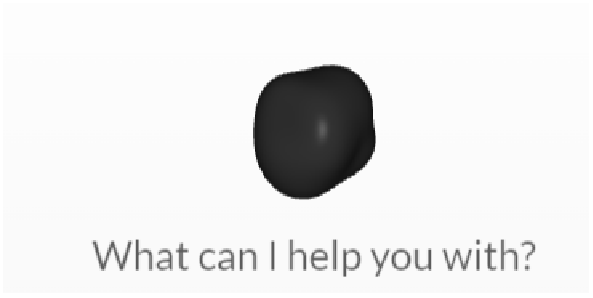


Figure 25 – Interactive Sphere.

i) MediaPipe vídeo tutorial - Extracting Face Mesh [59]

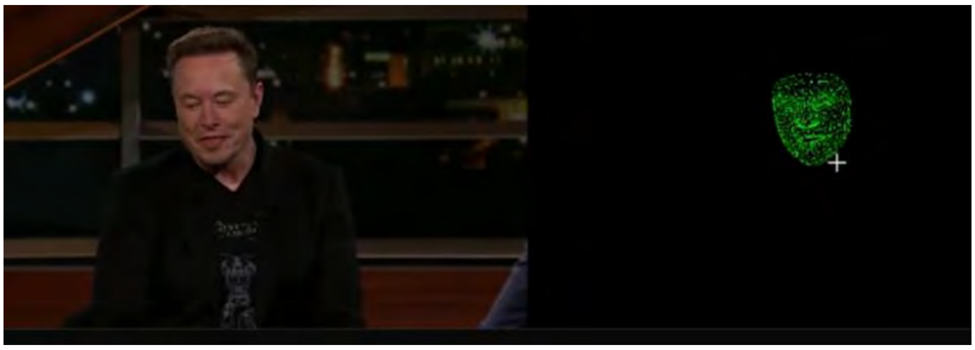


Figure 26 – Real-time Face Mesh Extraction.

j) Facial Landmark Detection using OpenCV [60] shows applications of facial key point detections: head pose estimation, face morphing, virtual makeover, and face replacement.

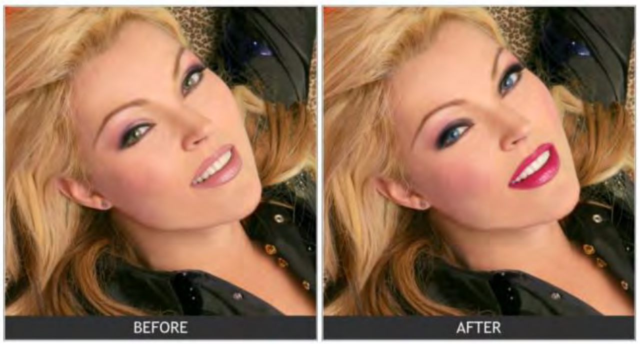


Figure 27 – Landmark detection for virtual makeover.

k) Face Morphing using OpenCV [61] shows the steps to face morphing: 1) Find Point Correspondences using Facial Feature Detection, 2) Delaunay Triangulation, 3) Warping images and alpha blending, 4) Results.

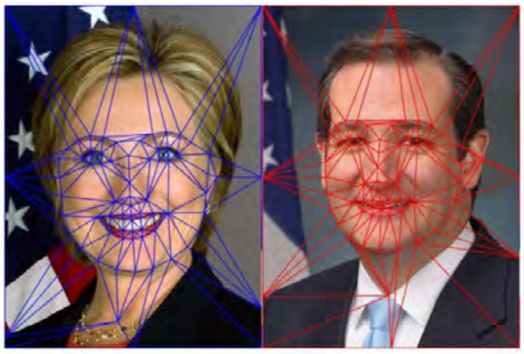


Figure 28 – Delaunay Triangulation for face morphing.

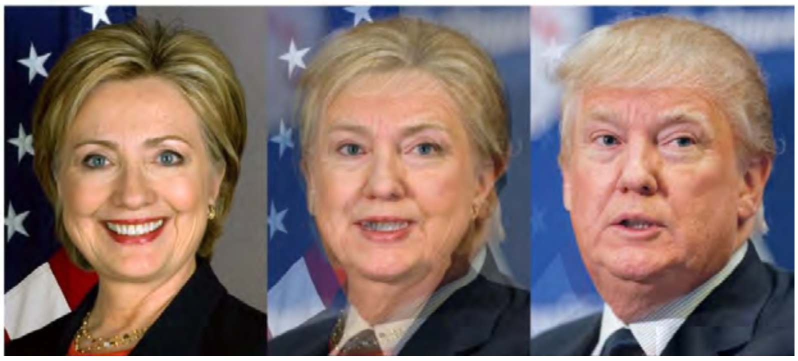


Figure 29 – Face morphing results.

l) Facial Landmark Detection Simplified with OpenCV [62] uses OpenCV and MediaPipe to detect 468 facial landmarks in an image. OpenCV is the cross-platform open-source library for computer vision, machine learning, and image processing using which we can develop real-time computer vision applications. It is mainly used for image or video processing, and analysis including object detection, face detection, etc. Facial landmarks are used to localize and represent important regions of the face. It states that MediaPipe Face Mesh estimates 468 3D face landmarks in real-time even on mobile devices. It requires only a single camera input by applying machine learning (ML) to infer the 3D surface geometry, without the need for a dedicated depth sensor.

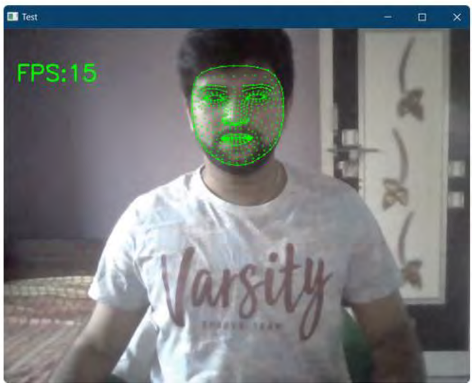


Figure 30 – Media Pipe Face Mesh 438 3D face landmarks.

m) The Top 7 Use Cases for Facial Landmark Detection [63] is an article that covers the definition of facial landmark detection, types of landmark detection algorithms, and 7 common use cases for facial landmark detection. It uses the DLib library. See the Appendix for more information.

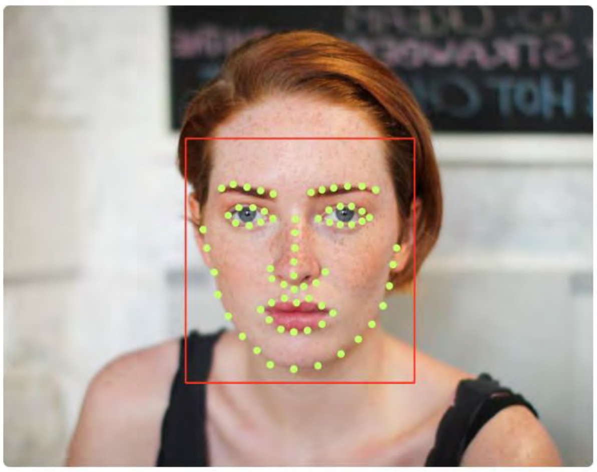


Figure 31 – 68 Face Landmark detection using Dlib.

n) This work [64] uses virtual reality for anatomical landmark annotation in geometric morphometrics. To study the shape of objects using geometric morphometrics, landmarks are oftentimes collected digitally from a 3D scanned model. The expert may annotate landmarks using software that visualizes the 3D model on a flat screen, and interaction is achieved with a mouse and a keyboard. However, landmark annotation of a 3D model on a 2D display is a tedious process and potentially introduces error due to the perception and interaction limitations of the flat interface. In addition, digital landmark placement can be more time-consuming than direct annotation on the physical object using a tactile digitizer arm. Since virtual reality (VR) is designed to resemble the real world more closely, we present a VR prototype for annotating landmarks on 3D models. We use an experimental setup, where four operators placed six landmarks on six grey seal (*Halichoerus grypus*) skulls in six trials for both systems. Our analysis shows that annotation in VR is a promising alternative to desktop annotation.

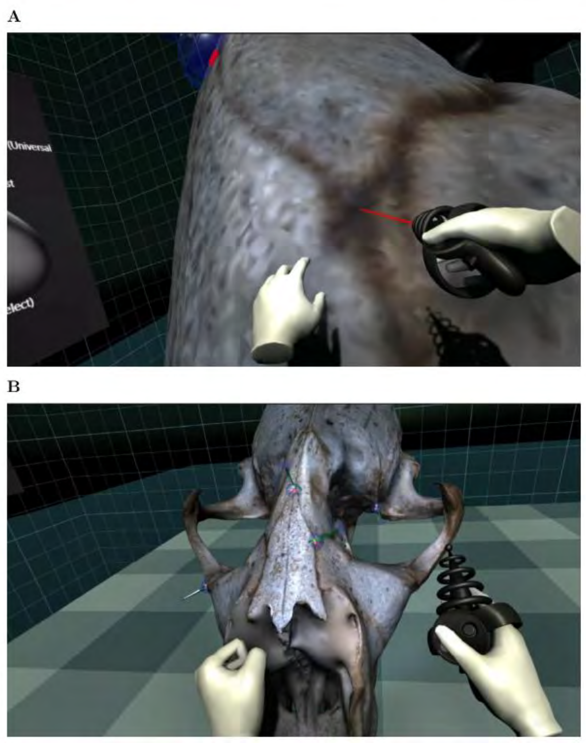


Figure 32 – Screenshots of the virtual environment. (A) The user is about to place a landmark with the ray-gun. (B) The user has placed six landmarks on the 3D model.

o) Landmark Editor Program [65] where you can load different face images, select the landmark type, edit, and save the modified landmarks.

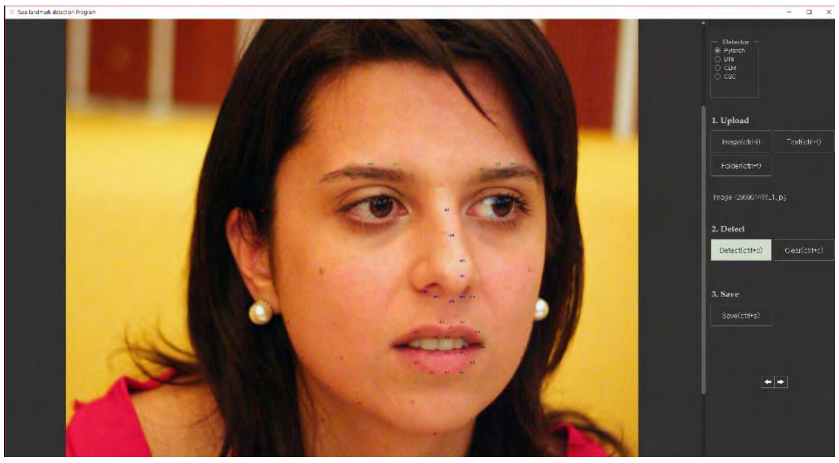


Image :29696148_1.jpg

Figure 33 – Screenshot of the landmark editor.

p) Interactive Data Editor [65][66] is a software to interactively edit data in a graphical manner.

q) Interactive Computer Graphics – A top-down approach with shader-based opengl [67] is a computer graphics reference book.

r) Example of a simulated interactive face morphing [68] based on the "Neural Implicit Morphing of Face Images" [51] using Gradio [3] and hosted by HuggingFace.

s) "68 landmarks are efficient for 3D face alignment: what about more?" [69] Discusses if 68 landmarks are sufficient or not for 3D alignment and propose a new face recognition method using 3D face reconstruction, and alignment.

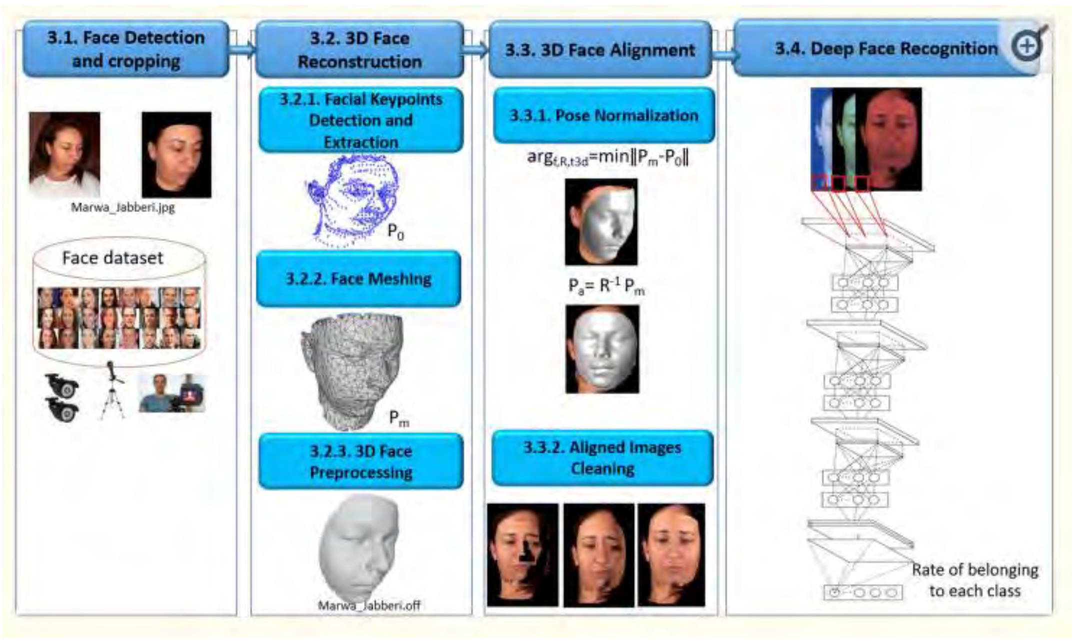
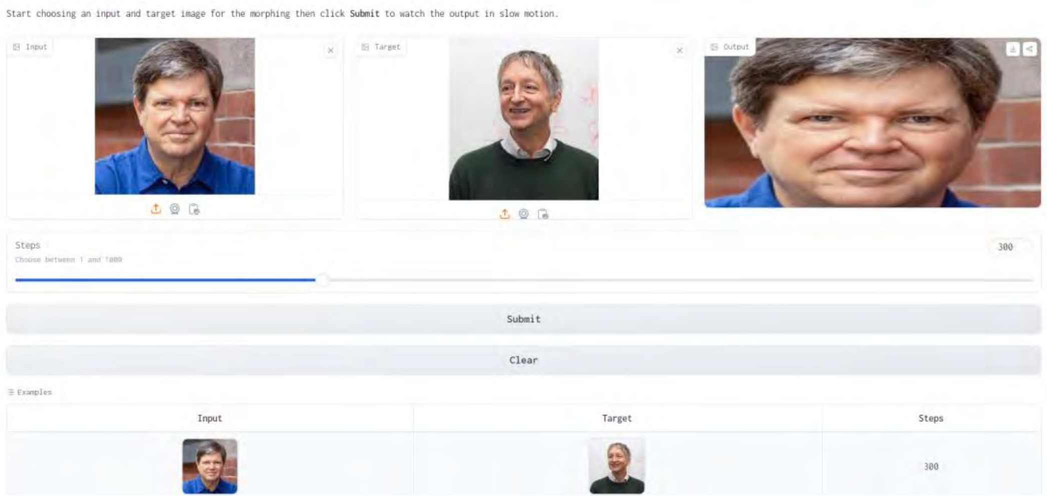

3.1. Face Detection and cropping
Marwa_Jabberi.jpg
Face dataset
3.2. 3D Face Reconstruction
3.2.1. Facial Keypoints Detection and Extraction
P_0
3.2.2. Face Meshing
P_m
3.2.3. 3D Face Preprocessing
Marwa_Jabberi.off
3.3. 3D Face Alignment
3.3.1. Pose Normalization
$\arg_{f,R,t3d}=\min\|P_m-P_0\|$
$P_a= R^{-1} P_m$
3.3.2. Aligned Images Cleaning
3.4. Deep Face Recognition
Rate of belonging to each class

Figure 34 – Proposed Method for Face Recognition.

7 EXPERIMENTS

7.1 Face Morphing

In this experiment, the faces of two generative artificial intelligence personalities, Yann LeCun and Geoffrey Hinton, were used as input and the video generated by the Face Morphing project algorithm [31] [51] was used as output, showing the transformation. The idea of the slide bar is to inform the number of steps to show (simulate) the progression of each morphing step. It was developed in Python, using the Gradio interface, and hosted on Hugging Face. The simulated morphing routine can be easily replaced by the real one.



Start choosing an input and target image for the morphing then click **Submit** to watch the output in slow motion.

Steps
Choose between 1 and 1000
300

Input	Target	Steps
		300

Figure 35 – Proposed Interface screen for simulating Face Morphing [77]

7.2 Face LandMarks Detection and Visualization

In this experiment two landmark detection models were used: Dlib and MEDIAPIPE. The tests were carried out by taking as input the video camera of the computer or cell phone or using a pre-recorded video. In Figure 2 we present the results of the detection and video of the morphing of LeCun and Hinton's faces. Python and the Dlib, MEDIAPIPE, and OpenCV libraries were used, among others.

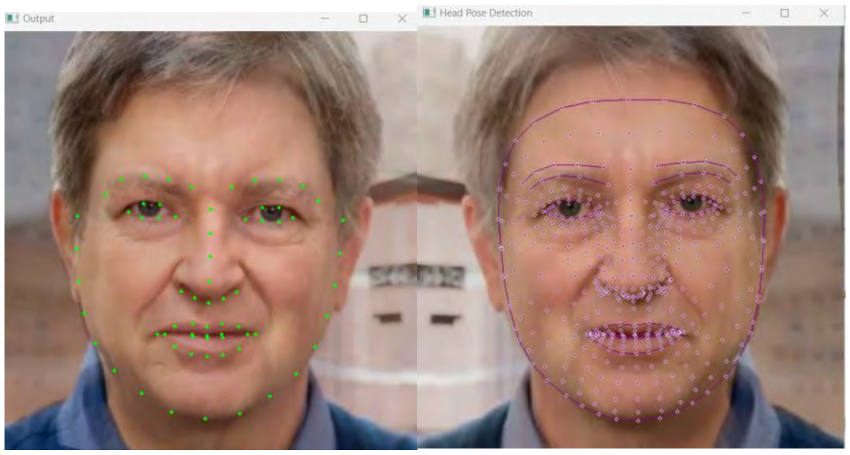


Figure 36 – Interface screen for viewing face Landmarks: DLIB on the left, and MEDIAPIPE on the right.

7.3 Face LandMarks Correspondence and Editing (Interaction)

The idea of this experiment is to combine the two previous ones, placing landmark type options (Dlib or MEDIAPIPE), whether you want the landmark to appear or be invisible. In addition, correspondence lines will be drawn between the landmarks of the source and target faces. These points can be edited, interactively with the help of the mouse, in their location coordinates of both the source and target images. Optionally, points can be added or subtracted. The visualization of the points (and the lines connecting the corresponding points) can be filtered by section (mouth, lips, nose, right eye, left eye, right eyebrow, left eyebrow, and jaw), by a single point, or by a range of points. It will be done in such a way that it is easy to start working with 2D and then migrate to 3D. The first version will be made using Python and will later evolve into Gradio and JavaScript (Three.js for example).

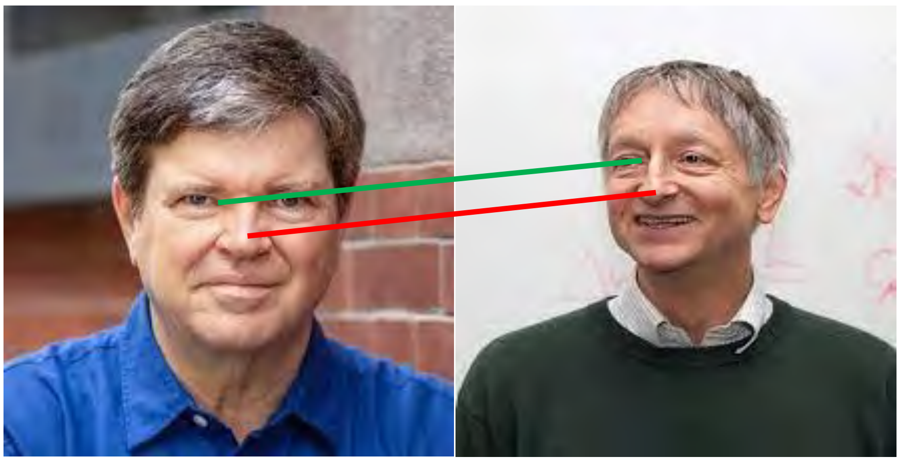


Figure 37 – Simulated Interface screen for editing landmarks. The coordinates of the origin and destination points can be changed interactively.

8 CONCLUSION AND FUTURE WORK

We showed here our objective of studying interactivity applied to Visgraf projects [45]. We established some requisites, presented some tools, related work, use cases and experiments.

Besides Face Morphing we intend to apply INTERACT-NET in generative artificial intelligence and 2D and 3D reconstruction using Gaussian Splatting.

9 BIBLIOGRAPHY

[1] "Fundamentals of Interactive Computer Graphics", J.D. Foley and A. Van Dam.

[2] "MediaPipe: A Framework for Building Perception Pipelines", *https://arxiv.org/pdf/1906.08172.pdf, 2019 https://developers.google.com/mediapipe https://www.assemblyai.com/blog/mediapipe-for-dummies/*

[3] GRADIO https://www.gradio.app/

[4] Flutter https://flutter.dev/

[5] Dash https://plotly.com/dash/

[6] Streamlit https://streamlit.io/

[7] Django https://www.djangoproject.com/

[8] React https://react.dev/

[9] Next.js https://nextjs.org/

[10] Three.js https://threejs.org/

[11] D3 https://d3js.org/

[12] p5.js https://www.geeksforgeeks.org/p5-js/ https://p5js.org/download/

[13] Luma AI's Three.js and R3F Gaussian Splatting Library

[14] Tensorflow.js TensorFlow.js | Machine Learning for JavaScript Developers https://observablehq.com/@tvirot/machine-learning-in-javascript-with-tensorflow-js?collection=@observablehq/a-i-artificial-intelligence

[15] https://observablehq.com/@swj0418/slow-walker

[16] https://observablehq.com/collection/@observablehq/a-i-artificial-intelligence

[17] Luma WebGL Library https://lumalabs.ai/luma-web-library

[18] React Three Fiber React Three Fiber Documentation (pmnd.rs)

[19] Vue https://vuejs.org/

[20] Svelte https://svelte.dev/

[21] "RITM - Reviving Iterative Training with Mask Guidance for Interactive Segmentation", https://ecosystem.supervisely.com/apps/ritm-interactive-segmentation/supervisely https://arxiv.org/pdf/2102.06583.pdf, 2021

[22] "Segment Anything Model (SAM)", Segment Anything | Meta AI (segment-anything.com) [2305.03678] Towards Segment Anything Model (SAM) for Medical Image Segmentation: A Survey (arxiv.org), 2023

[23] "Drag Your GAN: Interactive Point-based Manipulation on the Generative Image Manifold", https://www.youtube.com/watch?v=DxzsgV8rTOw

[24] "FreeDrag: Feature Dragging for Reliable Point-based Image Editing" https://lin-chen.site/projects/freedrag/ [2307.04684] FreeDrag: Feature Dragging for Reliable Point-based Image Editing (arxiv.org), 2023

[25] "DragDiffusion: Harnessing Diffusion Models for Interactive Point-based Image Editing" https://yujun-shi.github.io/projects/dragdiffusion.html https://arxiv.org/abs/2306.14435, 2023.
[26] "User-Controllable Latent Transformer for StyleGAN Image Layout Editing", https://arxiv.org/pdf/2208.12408.pdf, 2022.
[27] "EditGAN: High-Precision Semantic Image Editing" https://arxiv.org/pdf/2111.03186.pdf , 2021
[28] Angular https://docs.angularjs.org/guide/introduction
[29] "Interactively Assessing Disentanglement in GANs" Interactively Assessing Disentanglement in GANs (wiley.com), 2022 https://observablehq.com/@swj0418/slow-walker, 2021
[30] TensorFlow.js Example https://observablehq.com/@tvirot/machine-learning-in-javascript-with-tensorflow-js?collection=@observablehq/a-i-artificial-intelligence tfjs-models/gpt2 at master · tensorflow/tfjs-models · GitHub

[31] https://www.visgraf.impa.br/morph/Warping_and_Morphing_of_Images_using_Neural_Networks_LR.pdf , 2024.

[32] "What's Node.js ?" https://kinsta.com/knowledgebase/what-is-node-js/
[33] "Peering Inside the Black Box" https://www.republik.ch/2018/06/26/peering-inside-the-black-box
[34] "Machine Learning in the Browser" Machine learning in the browser / Ryan Seddon | Observable (observablehq.com)

Machine Learning For Front-End Developers With Tensorflow.js — Smashing Magazine

[35]Artificial Intelligence / Observable | Observable (observablehq.com)
[36] "Interactively Assessing Disentanglement in GANs" Interactively Assessing Disentanglement in GANs / Sangwon Jeong | Observable (observablehq.com)
[37] Transformers.js https://huggingface.co/docs/transformers.js/index
[38] Keras.js https://transcranial.github.io/keras-js/#/
[39] OpenCV.js https://docs.opencv.org/4.x/df/d0a/tutorial_js_intro.html
[40] Brain.js https://brain.js.org/#/
[41] Synaptic.js https://caza.la/synaptic/#/
[42] Neuro.js https://neuro.js.org/
[43] Convnet.js https://cs.stanford.edu/people/karpathy/convnetjs/
[44] ml5.js https://ml5js.org/
[45] Visgraf Projects https://visgraflab.impa.br/neural/2023/12/11/links/
[46] SERF: Fine-Grained Interactive 3D Segmentation and Editing with Radiance Fields https://arxiv.org/abs/2312.15856

[47] DLIB http://dlib.net/ml.html Davis E. King. Dlib-ml: A Machine Learning Toolkit. Journal of Machine Learning Research, 2009

[48] Taipy https://taipy.io/
[49] Tkinter https://docs.python.org/3/library/tkinter.html
[50] PyQt https://en.wikipedia.org/wiki/PyQt
[51] "Neural Implicit Morphing of Face Images" https://schardong.github.io/ifmorph/index.html [2308.13888] Neural Implicit Morphing of Face Images (arxiv.org)

[52] “Face and Landmark Detection using face-api.js”
https://justadudewhohacks.github.io/face-api.js/face_and_landmark_detection/
[53] ”Real Time AI Face Landmark Detection in 20 Minutes with Tensorflow.JS and React.”
https://www.youtube.com/watch?v=7lXYGDVHUNw
[54] https://techtee.medium.com/real-time-face-mesh-point-cloud-with-three-js-tensorflow-js-and-typescript-1f37ae844e1f
[55] https://threejs.org/examples/#webgl_decals
[56] https://codepen.io/zadvorsky/pen/PNXbGo
[57] “56 Three JS Examples - Collection of three.js (Javascript 3D library) code examples.”
https://freefrontend.com/three-js-examples/#google_vignette
[58] “AI Assistant | Three.js interactive sphere” AI Assistant | Three.js interactive sphere
https://codepen.io/jjsalgado/pen/yLBjvem
[59] “MediaPipe vídeo tutorial - Extracting FaceMesh”
https://www.youtube.com/watch?v=9O6VkIL3rZE
[60] Facial Landmark Detection | LearnOpenCV #
[61] Face Morph Using OpenCV — C++ / Python | LearnOpenCV #
[62] Facial Landmark Detection Simplified With OpenCV - Analytics Vidhya
[63] “The Top 7 Use Cases for Facial Landmark
Detection” https://www.plugger.ai/blog/the-top-7-use-cases-for-facial-landmark-detection
[64] ” Using virtual reality for anatomical landmark annotation in geometric morphometrics”
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8830334/
[65] https://github.com/yun-ss97/Facial-Landmark-Editing-Program?tab=readme-ov-file
[66] Interactive Data Editor https://github.com/Koushikphy/Interactive_Data_Editor
[67] “Interactive Computer Graphics – A top-down approach with shader-based opengl”
https://theswissbay.ch/pdf/Books/Computer%20science/Interactive%20computer%20graphics_a%20top-down%20approach%20with%20shader-based%20OpenGL%20%286th%20edition%29%20-%20Edward%20Angel%2C%20Dave%20Shreiner.pdf
[68] https://akopiler-face-morphing.hf.space
[69] “68 landmarks are efficient for 3D face alignment: what about more?”, 2023
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10066970/

https://link.springer.com/article/10.1007/s11042-023-14770-x

[70] face-api.js: Uma maneira de construir um sistema de reconhecimento facial no navegador. (ichi.pro)
[71]face-api.js — JavaScript API for Face Recognition in the Browser with tensorflow.js | by Vincent Mühler | ITNEXT
[72]Realtime JavaScript Face Tracking and Face Recognition using face-api.js’ MTCNN Face Detector | by Vincent Mühler | ITNEXT
[73] tracking.js (trackingjs.com)
[74] clmtrackr https://www.npmjs.com/package/clmtrackr?activeTab=readme
[75] Feature Visualization of GoogLeNet trained on ImageNet.
https://distill.pub/2017/feature-visualization/#d-footnote-1

10 APPENDIX

10.1 TOP USE CASES FOR FACIAL LANDMARK DETECTION [3]

Recognizing human facial features such as the eyes, nose, and lips, among others, is a challenging computer vision task known as facial landmark detection.

When combined with other computer vision tasks, such as head posture estimation, determining the direction of a person's gaze, recognizing facial movements, and even face swapping, facial landmark detection become an essential tool.

There are two steps to accomplish the task. Step one is to identify the face in the image. The process of face detection is used to pinpoint a human face in an image and return a value that is the location of the face's bounding rectangle. Once the face has been found, we need to search the points within the enclosing rectangle for landmarks.

Several different approaches exist for detecting facial landmarks, but they all have the same goal, that is to locate and categorize the following anatomical landmarks:

- Right eyebrow;
- Left eyebrow;
- Nose;
- Mouth;
- Jaw;
- Right eye;
- Left eye.

10.1.1 Classification of Algorithms

The three distinct types of facial landmark identification algorithms can be broken down into subcategories: those focusing on appearance and those focusing on form patterns.

Here, the facial appearance is the unique intensity distribution of pixels around facial landmarks or the whole face area. The face shape patterns are the distributions of landmark positions and their spatial connections.

There are essentially three types of landmark detection algorithms:

- Applying Holistic Strategies
- Methods Using a Constrained Local Model (CLM)
- Techniques based on regression

[3] https://www.plugger.ai/blog/the-top-7-use-cases-for-facial-landmark-detection

10.1.2 Common 7 Use Cases for Facial Landmark Detection

Facial landmark detection reveals valuable information that can be used in many contexts, including but not limited to human-computer interaction, entertainment, security monitoring, and medical applications.

a) Animation and Reenactments - For example, the Pix2PixHD neural network can help lipsyncing, and the DeFA algorithm can build a 3D face mesh from scratch. Using boundaries and facial landmarks, the Dlib library can be used to make faces from scratch.
b) Facial Recognition - Face verification, face recognition, and algorithms group in this use case. Face preprocessing and face alignment is two ways the best algorithms improve face recognition. These algorithms often use multi-task pipelined convolutional networks (MTCNN) to find faces and landmarks.
c) Facial Emotion Recognition - The movements of the lips, eyes, and eyebrows show how someone feels. Using facial landmarks can help identify facial emotions;
d) Driver Tracking – Monitoring closing eyes and head posture to alert that possibly the driver is asleep;
e) Replacement of the Face – To successfully clone one face onto another, we need an estimate of the locations of landmark features on both faces to align them.
f) Estimating Head Posture - Once a few critical facial features have been identified, the head's position can be roughly estimated.
g) Face Morphing – Transforming a source image in a destination (target) one in intermediate steps. We need an estimate of the locations of landmark features on both faces to align them

10.1.3 DLIB Library

Dlib is an average facial dataset, which includes 68 x and y coordinates annotated to represent landmarks on a person's face. Dlib is a popular open-source library that can detect specific features in images. In addition to landmark detection, the Dlib library has face detection capabilities. For DLib, histogram-oriented methods (HOG) are used for face identification, whereas Kazemi's model provides the foundation for landmark detection. It analyzes a face and returns 68 unique features:

- Jaw Points: 0-16
- Right Eyebrow Points: 17-21
- Left Eyebrow Points: 22-26
- Nose Points: 27-35
- Right Eye Points: 36-41
- Left Eye Points: 42-47
- Mouth Points: 48-60
- Lip Points: 61-67