INTERACT-NET: An Interactive Interface For Multimedia Machine Learning

Alberto Kopiler *IMPA*
Tiago Novello *IMPA*
Guilherme Schardong *University of Coimbra*
Luiz Schirmer *Unisinos*
Daniel Perazzo *IMPA*
Luiz Velho *IMPA*

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

***Index Terms*—interactive, interface, multimedia, machine learning**

I. Introduction

This work is not intended to be a survey about interactivity, as it would be necessary to read and analyze hundreds of articles. 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 (Vision and Graphics Laboratory). In this lab, there are already projects that work intensely in analyzing and synthesizing 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 is an integral part. In these courses the algorithmic part was highlighted, mainly 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 adoption of image generation tools using generative artificial intelligence, such as DALL-E, Stable Diffusion, Imagen and Midjourney only 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 channels. We can even input 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 image or more and get 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 ML models such as auto-encoders, 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, such as Surf - Speeded Up Robust Features, Sift - scale-invariant feature transform, ORB - Oriented FAST and Rotated BRIEF, Harrys Corner, edge detection, face landmarks like DLIB – 68 unique features or MediaPipe – 468 3D features), depending on the application. Another fundamental area of computer graphics is interactive computer graphics [13]. 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 innovation trend, voice-activated personal assistants that synthesize speech have become more efficient with ML. In other words, pipelines that receive voice input, recognize it, and transform it into text, which in turn activate tasks related to generating images, and video, or executing some of them as if they 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 get new functionalities and even improve existing ones, making these interfaces very powerful, as they can be self-generative. Additionally, we need to consider 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 both, 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 ML 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. 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) revolutionized

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 multi-modal information, that is multimedia, using ML, computer graphics, and image processing interactively, 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 begins by specifying requirements and, based on case studies associated with the projects currently underway at Visgraf and "their pain points", analyzes the tools available to meet this demand incrementally. You should avoid "reinventing the wheel", but rather seek "on the shelf" frameworks and solutions, preferably open source.

II. Motivation

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

III. Requisites

The desirable requirements are the following: interactivity, web orientation (preferably "serverless"), suitability for 2D but also 3D, mobility, quickness, and ease of use. Further details of each of the requirements are given below.

A. 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, 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.

B. Web Orientation

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 "serverless". Of course, 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.

C. 2D/3D

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

D. Mobility

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

E. Quickness

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

F. User Friendship

When building the user interface, you should remember that simplicity and easy navigability are key concepts.

IV. Tools

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

A. Web Interfaces

Here we list some general web frameworks that can be used to interact with the images in an ML pipeline.

1) Gradio: [2] is the fastest way to demo and share your ML model with a friendly web interface so anyone can use it, anywhere! The interface is Gradio's main high-level class and allows you to create a web-based GUI demo around a ML 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.

2) Streamlit: [17] [3] 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.

3) Flutter: [61] 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.

4) Dash: [9] 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.

5) Django: [14] is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of the hassle of web development, so you can focus on writing your app without reinventing the wheel. It's free and open source.

6) Mesop: [27] is a Python-based UI framework that allows quickly building web apps, like demos and internal apps.

We can also mention Pyramid [38], Flask [12], and Taipy [49] as good web interface resources.

B. JavaScript

JavaScript is one of the most popular programming languages in the world. It powers millions of websites today and 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 [60]. Here we list some JavaScript libraries that can be used to build interactively two categories of user interfaces: 2D and 3D graphics and ML. These libraries can be used individually or merged with more JavaScript libraries or even with one of the mentioned web interfaces.

1) JavaScript for 2D and 3D Graphics:

a) React: [40] is a JavaScript library for building user interfaces that allow the construction of web and native user interfaces out of individual pieces called components. 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 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: [33] 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 a larger team, Next.js can help the construction of interactive, dynamic, and fast React applications.

c) Three.js: [51] is a 3D JavaScript library that tries to make it easy to get 3D content on a web page. Three.js is often confused with WebGL since often, but not always, three.js uses WebGL to draw 3D. WebGL is a 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 into play. It handles stuff like scenes, lights, shadows, materials, textures, 3d math, all things you would have to write yourself if you were to use WebGL directly.

d) D3.js: [8] 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: [35] is a JavaScript library for creative coding, making it 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 functionalities. 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: [24] is a JavaScript library developed by Luma.ai to interface to Three.js and render Gaussian Splatting.

Luma WebGL Library [25], React Three Fiber [39], Babylon.js [4], PlayCanvas [37], Vue [56], Svelte [47], AngularJS [59], and Node.js [60] are also JavaScript libraries that span from standard web user interfaces to 2D/3D ones.

2) JavaScript for ML and Computer Vision:

a) TensorFlow.js: [50] is a library for ML in JavaScript. Develop JavaScript models and use ML directly in the browser or Node.js. TensorFlow.js is a general-purpose, WebGL-accelerated numeric platform for JavaScript. It brings highly performing ML building blocks to your fingertips, allowing you to train neural networks in a browser or run pre-trained models in inference mode.

b) Transformers.js: [54] is the state-of-the-art ML 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 you can run the same pre-trained 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, summarising, 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.

c) Face-api.js: [31] is a JavaScript API library for face detection and face recognition in the browser implemented on top of the tensorflow.js core API.

Keras.js [16], ml5.js [29], OpenCV.js [34], Synaptic.js [48], ConvNet.js [7], Neuro.js [32], Brain.js [5], Tracking.js [53], and clmtrackr [6] are additional JavaScript libraries for ML and computer vision.

Fig. 1. Face image landmarks with clmtrackr [6].

C. Libraries

1) DLIB: [18] contains a wide range of ML algorithms. All are designed to be highly modular, quick to execute, and

simple to use via a clean and modern C++ API. It is used in many applications including robotics, embedded devices, mobile phones, and large high-performance computing environments. Many examples use API wrappers for Python: Face detector, alignment, landmark detection, and recognition, among others.

2) MediaPipe: A wide range of potential ML applications today rely on several fundamental baseline ML tasks. For example, both gestural navigation and sign language detectors rely on the ability of a program to identify and track human hands. Once building something like a hand-tracking model is time-consuming and resource-intensive, a developmental bottleneck exists in creating all applications that rely on hand-tracking. To address this problem, MediaPipe [23] provided cornerstone ML models for common tasks like hand tracking, therefore removing the same developmental bottleneck occurring for a host of ML applications. These models, along with their excessively easy-to-use APIs, streamline the development process and reduce project lifetime for many applications that rely on Computer Vision. Another useful task provided by MediaPipe is face landmark detection and visualization.

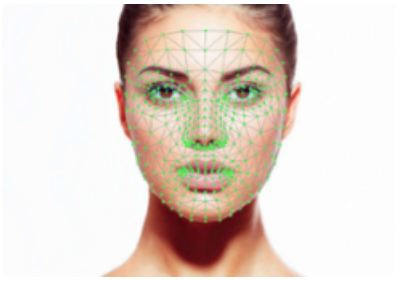

Fig. 2. Face image with MediaPipe Face Mesh drawn on top [23].

It is worth mentioning in this section the WebGPU [57] initiative that enables web developers to use the underlying system's GPU (Graphics Processing Unit) to carry out high-performance computations and draw complex images that can be rendered in the browser. There is also the WebXR [58] initiative for virtual, and augmented reality in web browsers.

V. Related Work

In this section, we will show related work with interactive techniques to mark images (e.g., clicks, scribble, or area of interest selection, for 2D and 3D) either to guide generative ML or to correct stages of this learning. Drag Diffusion [45], Free Drag [22], Drag Your Gan [36], EDIT Gan [21], and UserControllablelLT [10] show simple interactions like marking two points (two clicks), an origin and a target so that to generate a new image composed by the drag interaction and adding anchors to mark positions not to be modified or mark an area to limit the image transformation only inside it. The task of RITM [46] (Reviving Iterative Training with Mask Guidance for Interactive Segmentation), SAM [19] (Segment Anything Model), and SERF [63] (Fine-Grained Interactive 3D Segmentation and Editing with Radiance Fields) is to segment images using click-based interactive segmentation. You can select part of the image by clicking one or more times or deselecting only a small part also by clicking.

Fig. 3. Segment Anything Model (SAM) [19].

It is worthwhile mentioning the Observable Notebooks [3] with several examples of visualization and interactions for ML, image processing, and computer graphics: Interactively Assessing Disentanglement in GANs, ML in The Browser, Drawings to Human, Visualization in Deep Learning, Background Position Scrubber and Peering Inside the Black Box.

VI. Use Cases

This section lists use cases related to face morphing, one of Visgraf's projects selected to add interaction. The first one is the Neural Implicit Morphing of Face Images [43], Visgraf's paper which uses DLIB for landmarks, and ML in the learning optimization phase. It has an interactive editor to adjust the auto-detected landmarks in the warp process.

Fig. 4. Face Morphing [43].

Another use case is face and landmark detection using face-api.js [30]. Next, we have a Three.js use case to draw decals

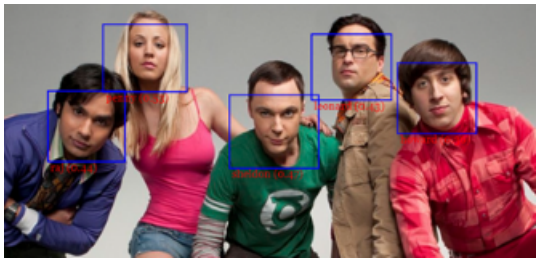

Fig. 5. Face Recognition using face-api.js [30].

interactively. [52]. Finally, we list more use cases related to INTERACT-NET: Real-Time AI Face Landmark Detection in 20 Minutes with Tensorflow.JS [41]; Real-time 3D face mesh point cloud with Three.JS, Tensorflow.js and Typescript [42]; 56 Three JS Examples - Collection of three.js [1]; MediaPipe video tutorial - Extracting Face Mesh Facial Landmark Detection using OpenCV [26]; Facial Landmark Detection Simplified with OpenCV and MediaPipe [44]; The Top 7 Use

Fig. 6. Interactive decals using Three.js [52].

Cases for Facial Landmark Detection [62]; Virtual Reality for anatomical landmark annotation in geometric morphometrics [28]; Landmark Editor Program [11]; Simulated interactive Neural Implicit Morphing of Face Images using Gradio and hosted by HuggingFace [20]; and 68 landmarks are efficient for 3D face alignment: what about more? [15].

VII. Experiments

A. Face Morphing

In this experiment [64], the faces of two generative artificial intelligence personalities, Yann LeCun and Geoffrey Hinton, were used as input. The video generated by the Face Morphing project algorithm [43] 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 [2], and hosted on Hugging Face. The simulated morphing routine can be easily switched to the real one.

B. Face Landmarks Detection and Visualization

In this experiment two landmark detection models were used: DLIB [18] and MediaPipe [23]. 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 7 we present the results of a video frame capture of LeCun and Hinton faces morphing. Python, DLIB, MediaPipe, and OpenCV libraries were used among others.

C. Face Landmarks Detection and Visualization

The idea of this experiment is to combine the 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 (dragged), interactively with the help of the mouse, in their location coordinates on only one image, or both the source and target images. Optionally, points can be added or subtracted. The points' visualization (and the lines connecting the corresponding points) can be filtered by section (mouth, lips, nose, eyes, eyebrows, and jaw), by a single point, or by a range of points. It will be done so that it is easy to start working with 2D and then migrate to 3D. The first version will be made using Python and later evolve into Gradio and JavaScript (Three.js for example). Another feature will be to draw the Delaunay triangles over the faces and allow the interface to stretch the lines like a rubber band. This is useful to interactively align the landmarks to the faces, as the algorithms often outcomes good automatic landmarks, but not perfect ones, as needed in the case of face morphing.

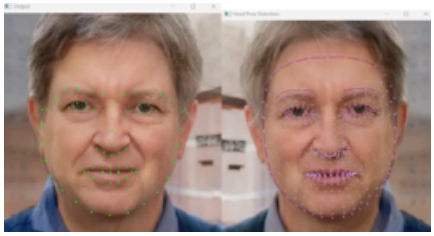

Fig. 7. Interface screen for viewing face Landmarks: DLIB on the left, and MediaPipe on the right. source: authors.

VIII. Conclusions

We showed here our objective of studying interactivity applied to Visgraf's projects [55]. We established requisites and presented some tools, related work, use cases, and experiments. Besides Face Morphing we intend to apply INTERACT-NET in generative AI, 2D, and 3D reconstruction using Gaussian Splatting, and also in VR, and XR projects.

References

[1] *56 Three JS Examples - Collection of three.js (Javascript 3D library) code examples.* https://freefrontend.com/three-js-examples. (accessed Jul. 03, 2024).

[2] Abubakar Abid et al. "Gradio: Hassle-free sharing and testing of ml models in the wild". In: *arXiv preprint arXiv:1906.02569* (2019).

[3] *Artificial Intelligence / Observable — Observable (observablehq.com).* https://observablehq.com/collection/@observablehq/a-i-artificial-intelligence. (accessed Jul. 03, 2024).

[4] *babylon.js.* https://www.babylonjs.com. (accessed Jul. 03, 2024).

[5] *Brain.js: GPU accelerated Neural networks in JavaScript for Browsers and Node.js.* https://brain.js.org. (accessed Jul. 03, 2024).

[6] *clmtrackr.* https://www.npmjs.com/package/clmtrackr. (accessed Jul. 03, 2024).

[7] *ConvNetJS: Deep Learning in your browser.* https://cs.stanford.edu/people/karpathy/convnetjs. (accessed Jul. 03, 2024).

[8] *D3js: The JavaScript library for bespoke data visualization.* https://d3js.org. (accessed Jul. 03, 2024).

[9] Elias Dabbas. *Interactive Dashboards and Data Apps with Plotly and Dash: Harness the power of a fully fledged frontend web framework in Python–no JavaScript required.* Packt Publishing Ltd, 2021.

[10] Yuki Endo. *User-Controllable Latent Transformer for StyleGAN Image Layout Editing*. 2022. arXiv: 2208.12408 [cs.CV]. URL: https://arxiv.org/abs/2208.12408.

[11] *Facial Landmark Editing Program.* https://github.com/yunss97/Facial-Landmark-Editing-Program?tab=readme-ov-file. (accessed Jul. 03, 2024).

[12] *Flask: A simple framework for building complex web applications.* https://pypi.org/project/Flask. (accessed Jul. 03, 2024).

[13] James D Foley and Andries Van Dam. *Fundamentals of interactive computer graphics*. Addison-Wesley Longman Publishing Co., Inc., 1982.

[14] Jeff Forcier, Paul Bissex, and Wesley J Chun. *Python web development with Django*. Addison-Wesley Professional, 2008.

[15] Marwa Jabberi et al. "68 landmarks are efficient for 3D face alignment: what about more? 3D face alignment method applied to face recognition". In: *Multimedia Tools and Applications* 82.27 (2023), pp. 41435–41469.

[16] *Kera.js*. https://transcranial.github.io/keras-js. (accessed Jul. 03, 2024).

[17] Mohammad Khorasani, Mohamed Abdou, and Javier Hernández Fernández. "Streamlit use cases". In: *Web Application Development with Streamlit: Develop and Deploy Secure and Scalable Web Applications to the Cloud Using a Pure Python Framework*. Springer, 2022, pp. 309–361.

[18] Davis E King. "Dlib-ml: A machine learning toolkit". In: *The Journal of Machine Learning Research* 10 (2009), pp. 1755–1758.

[19] Alexander Kirillov et al. "Segment anything". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 4015–4026.

[20] Alberto Arkader Kopiler. *Face Morphing Experiment.* https://akopiler-face-morphing.hf.space. (accessed Jul. 03, 2024).

[21] Huan Ling et al. "EditGAN: High-Precision Semantic Image Editing". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[22] Pengyang Ling et al. "FreeDrag: Feature Dragging for Reliable Point-based Image Editing". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 6860–6870.

[23] Camillo Lugaresi et al. "Mediapipe: A framework for building perception pipelines". In: *arXiv preprint arXiv:1906.08172* (2019).

[24] *Luma AI's Three.js and R3F Gaussian Splatting Library.* https://discourse.threejs.org/t/luma-ais-three-js-and-r3f-gaussian-splatting-library/58960. (accessed Jul. 03, 2024).

[25] *Luma WebGL Library.* https://lumalabs.ai/luma-web-library. (accessed Jul. 03, 2024).

[26] *MediaPipe vídeo tutorial - Extracting FaceMesh.* https://www.youtube.com/watch?v=9O6VkIL3rZE. (accessed Jul. 03, 2024).

[27] *Mesop: Quickly build web UIs in Python.* https://google.github.io/mesop. (accessed Jul. 03, 2024).

[28] Dolores Messer et al. "Using virtual reality for anatomical landmark annotation in geometric morphometrics". In: *PeerJ* 10 (2022), e12869.

[29] *ml5.js: Friendly Machine Learning for the Web*. https://ml5js.org. (accessed Jul. 03, 2024).

[30] Vincent Muhler. "face-api.js-JavaScript API for Face Recognition in the Browser with tensorflow.js". In: *Medium* (2018).

[31] V. Mühler. *Face and Landmark Detection using face-api.js*. https://justadudewhohacks.github.io/face-api.js/face_and_landmark_detection. (accessed Jul. 03, 2024).

[32] *Neuro.js: machine learning framework for building AI assistants and chat-bots.* https://neuro.js.org. (accessed Jul. 03, 2024).

[33] *Next.js: The React Framework for the Web.* https://nextjs.org. (accessed Jul. 03, 2024).

[34] *OpenCV.js: OpenCV for the JavaScript programmer.* https://docs.opencv.org/4.x/df/d0a/tutorial_js_intro.html. (accessed Jul. 03, 2024).

[35] *p5*js*. https://p5js.org. (accessed Jul. 03, 2024).

[36] Xingang Pan et al. "Drag your gan: Interactive point-based manipulation on the generative image manifold". In: *ACM SIGGRAPH 2023 Conference Proceedings*. 2023, pp. 1–11.

[37] *playcanvas: Web Graphics Creation Platform.* https://playcanvas.com. (accessed Jul. 03, 2024).

[38] *Pyramid: The Start Small, Finish Big, Stay Finished Framework.* https://trypyramid.com. (accessed Jul. 03, 2024).

[39] *React Three Fiber.* https://docs.pmnd.rs/react-three-fiber/getting-started/introduction. (accessed Jul. 03, 2024).

[40] *React: The library for web and native user interfaces.* https://react.dev. (accessed Jul. 03, 2024).

[41] *Real Time AI Face Landmark Detection in 20 Minutes with Tensorflow.JS and React.* https://www.youtube.com/watch?v=7lXYGDVHUNw. (accessed Jul. 03, 2024).

[42] *Real-time face mesh point cloud with Three.JS, Tensorflow.js and Typescript.* https://techtee.medium.com/real-time-face-mesh-point-cloud-with-three-js-tensorflow-js-and-typescript-1f37ae844e1f. (accessed Jul. 03, 2024).

[43] Guilherme Schardong et al. "Neural implicit morphing of face images". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 7321–7330.

[44] Syed Abdul Gaffar Shakhadri. *Facial Landmark Detection Simplified With Opencv.* https://www.analyticsvidhya.com/blog/2021/07/facial-landmark-detection-simplified-with-opencv/. (accessed Jul. 03, 2024).

[45] Yujun Shi et al. *DragDiffusion: Harnessing Diffusion Models for Interactive Point-based Image Editing*. 2024. arXiv: 2306.14435 [cs.CV]. URL: https://arxiv.org/abs/2306.14435.

[46] Konstantin Sofiiuk, Ilya A Petrov, and Anton Konushin. "Reviving iterative training with mask guidance for interactive segmentation". In: *2022 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2022, pp. 3141–3145.

[47] *Svelte: cybernetically enhanced web apps.* https://svelte.dev. (accessed Jul. 03, 2024).

[48] *Synaptic.js: The javascript architecture-free neural network library for node.js and the browser.* https://caza.la/synaptic. (accessed Jul. 03, 2024).

[49] *Taipy: Go beyond existing libraries. Build Python Data AI web applications.* https://taipy.io. (accessed Jul. 03, 2024).

[50] *TensorFlow.js — Machine Learning for JavaScript Developers*. https://www.tensorflow.org/js. (accessed Jul. 03, 2024).

[51] *Three.js*. https://threejs.org. (accessed Jul. 03, 2024).

[52] *Three.js Decal Splatter.* https://threejs.org/examples/\#webgl_decals. (accessed Jul. 03, 2024).

[53] *tracking.js: A modern approach for Computer Vision on the web.* https://trackingjs.com. (accessed Jul. 03, 2024).

[54] *Transformers.js: State-of-the-art Machine Learning for the web.* https://huggingface.co/docs/transformers.js/index. (accessed Jul. 03, 2024).

[55] *Visgraf Projects.* https://visgraflab.impa.br/neural/2023/12/11/links. (accessed Jul. 03, 2024).

[56] *Vue: The Progressive JavaScript Framework.* https://vuejs.org. (accessed Jul. 03, 2024).

[57] *WebGPU.* https://www.w3.org/TR/webgpu. (accessed Jul. 03, 2024).

[58] *WebXR.* https://www.w3.org/TR/webxr. (accessed Jul. 03, 2024).

[59] *What Is AngularJS?* https://docs.angularjs.org/guide/introduction. (accessed Jul. 03, 2024).

[60] *What's Node.js ?* https://kinsta.com/knowledgebase/what-is-node-js. (accessed Jul. 03, 2024).

[61] Eric Windmill. *Flutter in action*. Simon and Schuster, 2020.

[62] Halime Yilmaz. *The Top 7 Use Cases for Facial Landmark Detection.* https://www.plugger.ai/blog/the-top-7-use-cases-for-facial-landmark-detection. (accessed Jul. 03, 2024).

[63] Kaichen Zhou et al. "Serf: Fine-grained interactive 3d segmentation and editing with radiance fields". In: *arXiv preprint arXiv:2312.15856* (2023).