Lightning Artist Toolkit: A Hand-Drawn Volumetric Animation Pipeline

Nicholas Allan Fox-Gieg

A Dissertation submitted to the Faculty of Graduate Studies
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

Computational Arts, York University, Toronto, Ontario

April 2025

Abstract

This research contributes a set of methods in the form of a toolkit for freely integrating live-action volumetric video with hand-drawn volumetric animation, implemented as one component of a larger collection of pipeline tools under the title *Latk* (Lightning Artist Toolkit). This is achieved by both intuitively drawing hand-made 3D animation in *XR* (extended reality), and using trained *ML* (machine learning) models to generate a collection of 3D brushstrokes from a point cloud that approximates what an artist might draw by hand in XR.

The Kinect, the first consumer depth camera, arrived in 2010; in 2016, the HTC Vive headset introduced the first mass-market *6DoF* (six degrees of freedom) controllers. Combined, these two advances unlocked a new approach to creating frame-by-frame animation with 6DoF drawing tools, which my research has developed as a complete pipeline for hand-drawn volumetric animation. At the time of writing it is the only open-source example of its kind. The goal of the project is to make creation in 3D as expressive and intuitive as creation in 2D, by retaining the human gesture from its origins in hand-drawn animation on paper.

Importing and manipulating scanned photographic images alongside handmade drawings has been a core feature of 2D image editing and animation tools for over fifty years. Initially, applying raster editing capabilities to real-world animation production was impractical—so the earliest hand-drawn computer-animated short films used 2D vector strokes. Today, operating naïvely on 3D voxels similarly requires excessive computational resources to be scaled up for even a few minutes of high-resolution footage, and working with 3D vector graphics representations offers a promising solution.

At this project's core is a collection of applied machine learning systems that transform live-action volumetric video into a sequence of volumetric brushstroke vectors. Integrated into a conventional animation workflow, this is suitable for the practical production of hand-drawn 3D animated short films in an XR drawing system. The contribution is less a computer vision challenge with an objective goal, as with for example point cloud segmentation, than it is an attempt to approximate human aesthetics—an imitation of a drawing process that records as markings the information from a scene that was subjectively important to an individual artist.

In addition to supporting animation production through this workflow, this project also contributes a large public dataset of 3D drawings that may be usable in new and unexpected ways.

Dedication

To Meagan and Beth, who is new here.

Acknowledgements

I would like to thank, in order of appearance—

Graham Wakefield, my supervisor, and supervising committee members Matt Kyan and Ian Garrett.

In York Computational Arts administration, Michael Longford, Mark-David Hosale, Joel Ong, Don Sinclair, Dawn Burns, Andrea DiFlorio Sgro, Helen Ogundeji, Aishah Rashid, and Yameng Zou.

Franci Duran made my introductions to York Cinema Media Arts; Michael Haslam set me on the path to getting all the GPU compute I needed from SHARCNET aka Compute Canada aka DRAC; Melanie Wilmink got me sorted with Scrivener and Zotero.

Andy Baker invited me to join the Open Brush development team, and suggested several extremely useful ML models for my early research.

Andrew Hogue at Ontario Tech maintains the Kinect capture array that provided many of the testing examples for this research; Chris Remde is the lead developer of the open-source multi-Kinect capture solution LiveScan3D; Sarah Vollmer, my Alice Lab colleague, helped us set up our own array at York; Aimée Mitchell shepherded us through the procurement process.

Rounding out our Common Volume volumetric capture community are Alexander Porter (whose 3D-printed Asus Xtion tripod mount makes an appearance in Fig. 7), and co-founders Cindy Poremba and Veronika Szkudlarek at OCAD.

Also at OCAD, Judith Doyle and Tyson Moll, who made the A-Frame starter templates that I use with volcap material.

I developed some of the ideas here at artist residencies facilitated by Golan Levin at Carnegie Mellon University's STUDIO for Creative Inquiry; Benton Bainbridge at the FEED Media Art Center; and Madi Piller at Pix Film Gallery.

Valuable access to XR equipment came via Michael "wmmsi" Irwin at York Cinema Media Arts; Rob Allison at York VISTA; Fangmin Lee, Namir Ahmed, and Jimmy Tran at the Toronto Metropolitan University Library Collaboratory; Tiffany Schofield and Rob Cruickshank at InterAccess; and Greg Woodbury at Charles Street Video.

John Dupuis and Lily Ren at York University Libraries, and Erin Clary at FRDR, guided me through my

publishing, dataset, and rights issues.

Chris Coleman and Laleh Mehran invited me to the Clinic for Open Source Arts at the University of Denver, where I was able to develop a broader community strategy for Latk.

I developed the concluding section at reading groups run by Scott Richmond, Cate Alexander, Mynt Marsellus, and Matt Nish-Lapidus at the University of Toronto's Centre for Culture and Technology.

Marcus Gordon, Grace Grothaus, and Michael Palumbo from my PhD cohort started the writing group that brought this dissertation over the finish line.

And finally, thanks to external examining committee members Alexis Morris and Hector Centeno.

This work was supported in part by a SSHRC Doctoral Fellowship (752-2023-2691), an Ontario Graduate Scholarship, a York University VISTA Doctoral Scholarship, and a York University Crocker-Hunkin Award in Fine Arts.

This research was also made possible with infrastructure funded via the Canada Foundation for Innovation (CFI) grant #40207 "Infrastructure for Embodied Multimodal Awareness in Human-Machine Creativity", PI Graham Wakefield.

Table of Contents

List of Figures

Preface

This dissertation follows the York Faculty of Graduate Studies' archival requirements for accompanying "complex digital" material—defined as work containing integral digital media components. Full details are available in Appendix A. In addition to observing current best practices regarding the use of commercial code-sharing sites like GitHub, the primary code repositories and datasets are distributed for long-term preservation as follows:

Federated Research Data Repository (Canada)

TiltSet dataset	https://doi.org/10.20383/103.0917

Borealis (Canada)

ABC-Draco dataset	https://doi.org/10.5683/SP3/QGGXYJ

Zenodo (Switzerland)

Blender addon	https://doi.org/10.5281/zenodo.14927542
Unity app for iOS and Android	https://doi.org/10.5281/zenodo.14931984
Unity app for Quest/Oculus and Vive	https://doi.org/10.5281/zenodo.14931980
JavaScript library	https://doi.org/10.5281/zenodo.14933485
openFrameworks addon	https://doi.org/10.5281/zenodo.14933475
Processing library	https://doi.org/10.5281/zenodo.14933451
Python module	https://doi.org/10.5281/zenodo.14933479
Unity package	https://doi.org/10.5281/zenodo.14933477

1. Introduction

> "Since ancient times, artists have longed to create with moving lights a music for the eye comparable to the effects of sound for the ear. If they were less successful than composers of auditory music, the sole reason rests in the fact that light is harder to manipulate than air."
> —William Moritz (1986)[1]

Fig. 1. A volumetric drawing done in Open Brush, rendered in Blender.

New media art history periodically undergoes rapid transition periods—when inflection points in technological development suddenly place previously inaccessible processes and equipment in the hands of a large cohort of studio artists, and enable the creation of small-scale, personal work that until recently was feasible only on an industrial level. The latest such period has involved the technologies that we now collectively refer to as *XR* ("extended reality", or a catchall acronym for "virtual reality + augmented reality + mixed reality"),[2] the primary novelty of which I will argue came from the advent of mass-market depth cameras and *6DoF* ("six degrees of freedom", which means 3D position and orientation) controllers, rather than the more commonly-discussed VR headsets. In particular I will consider two events—the arrival of Microsoft's Kinect depth camera in 2010, and the HTC Vive

roomscale VR kit in 2016.[3] Combined, these two advances unlocked a new approach to creating frame-by-frame animation with 3D scanning and 6DoF drawing tools.

Creating hand-drawn XR animation with these devices has turned out to be an approach that fits perfectly with my own filmmaking methods. To support my long-term efforts in this field, I developed ***Latk* (the Lightning Artist Toolkit)**—a complete pipeline for frame-by-frame volumetric animation that, as far as I know, is currently the only open-source example of its kind. My goal with this project has been to make creation in 3D as expressive and intuitive as drawing in 2D, retaining the human gesture from its origins in hand-drawn animation on paper. At the project's core is a collection of applied machine learning systems that transform live-action *volcap*, or volumetric video—"a computational fusion of digital video and depth sensor data, resulting in a spatialized, and potentially navigable, 3D captured moving image"[4]—into a sequence of volumetric brushstrokes. Integrated into a conventional animation workflow, this output is now suitable for the practical production of hand-drawn 3D animated short films in an XR drawing system. My hope with Latk is that open-source, non-commercial research—done, as we will see later, in collaboration with artists and educators and responding to our community's present needs—has succeeded where earlier efforts have failed.

In particular, Latk has sought a way to freely integrate live-action volumetric video with hand-drawn volumetric animation. Importing and manipulating scanned photographic images alongside drawings has been a core feature of raster 2D image editing and animation tools for over fifty years.[5] But initially, applying these editing capabilities to real-world animation production was impractical. Charles Csuri's *Hummingbird* (1967), the first widely recognized hand-drawn computer-animated short film, used 2D vector strokes in an era when storage for an equivalent raster representation would have been beyond the capabilities of all but the largest institutions.[6] Today, operating naïvely on 3D voxels similarly still requires too much compute power to scale up to large quantities of high-resolution footage; post-processing live-action volcap on consumer hardware for even a few minutes of material has only recently become feasible. However, 3D vector graphics representations of hand-drawn animation, for example those in Wesley Allsbrook's short film *Dear Angelica* (2017), offer a flexible and powerful near-term solution.[7]

When the ability to capture brushstrokes with a digital stylus arrived with Ivan Sutherland's watershed 1963 Sketchpad project, it tapped into an older history of hand-drawn lines at the vanguard of experimental animation.[8] In the pre-photographic era, Charles Wheatstone made the first stereograms using pen and paper.[9] And at the beginning of film animation, early animators like Winsor McCay first created their work for a live audience in vaudeville halls, and were billed as *lightning artists*—the

namesake for this project.[10] In the second half of the 20th century, a cluster of essential developments in vector graphics specifically for hand-drawn computer animation were the product of public investment in Canada, most significantly Peter Foldès' *Hunger* (1974), produced at the National Film Board and using a vector animation system developed at the National Research Council.[11]

Moving from general historical milestones to personal inspiration, in 1983, the donation of a Telidon vector graphics workstation to the Toronto arts organization InterAccess[12] became an early example of the potential of vector animation tools in the hands of independent artists outside of major institutions, prefiguring better-known industrial efforts like Disney's 1989 CAPS system.[13] In 1997, the IMAX SANDDE project created the first complete pipeline for 6DoF hand-drawn animation production.[14] And more recently, Google Creative Lab's game *Quick, Draw!* (2016) involved the procedural recording, parsing, and manipulation of a vast quantity of hand-drawn vector artwork using applied *ML* (Machine Learning) techniques.[15]

My work on the original *Quick, Draw!* development team inspired me to explore my own ML applications for hand-drawn 3D animation. A key discovery emerged for me there, that ML systems could connect different representations of data in ways that would be neither obvious nor practical using procedural methods. With *Quick, Draw!* it turned out that a system designed for semantic photo recognition could also reliably recognize objects in line drawings, although only for a small set of cases—ideal for the preselected challenges in a Pictionary-style drawing game. Similarly, my Latk project relies on a process that transforms raster-based volumetric representations (live-action volumetric video) into vector-based representations (sequences of volumetric brushstrokes). The contribution is less a computer vision challenge with an objective goal, as with for example point cloud segmentation, than it is an attempt to approximate the aesthetics of human vision—to generate a collection of brushstrokes from a point cloud that resembles what an artist might draw from scratch in XR, in imitation of a drawing process that records as markings the information from a scene that was subjectively important to an individual artist.

This dissertation uses a "complex digital" approach according to the procedures laid out by the York Faculty of Graduate Studies—meaning that it exists as a support document for a larger body of research-creation with a "high reliance on media (e.g., images, audio, videos, computer code and/or data set) in which the digital material is an integral part of the work as a whole."[16] It is challenging to adequately represent the significant contributions of this research in the static, 2D print-based medium of a dissertation. Accordingly, the dissertation links to many of its research contributions in data, code, software and video formats. Highlights of this body of work include an archival dataset I call the TiltSet,

containing over 56,000 licensed 3D drawings in Tilt Brush format; demonstration apps archived in open-source repositories and distributed on multiple commercial app stores like the Apple App Store and Steam; libraries distributed via major package managers like npm and PyPI; short films making use of these tools distributed on video platforms like Vimeo and my personal website; and a SIGGRAPH Art Paper surveying my ML methods.[17]. All these are fully detailed in Appendix A and also linked on my project website (https://lightningartist.org).

In order to build my functional prototypes, I overcame four major challenges that placed this project well outside the paths of least resistance in artist-driven ML experimentation; I will address each in turn. First, in Section 2, I explain why Latk needs to work with vector data instead of raster data. This requires a broader discussion of vector and raster graphics—the two fundamentally different image representations underlying computer animation throughout its history—and the creative possibilities that emerge from the tension between them. Next, Latk is an attempt to develop and sustain hand-drawn animation tools in 3D, so in Section 3, I review the evolution of the hardware drawing interfaces that have so far allowed us to explore the frame-by-frame manipulation of 3D vector brushstrokes. Then, in Section 4, I detail the ML methods that convert my volumetric captures from raster into vector representations, including the ethical sourcing of a large hand-drawn 3D training dataset. Finally, in Section 5, I discuss the practical artistic use of the toolkit, by storing its output in a file format that exposes the resulting frame-by-frame animation to any production pipeline.

2. First Word Vector, Last Word Raster

> "First word art is groundbreaking and exploratory. It's playing outside any rule structures. It side-steps competition. People often don't know how to react to it. Last word art is virtuosity after the rules have been fixed. It accepts the established form, and is judged by comparison."
> —Michael Naimark (2001)[18]

The two fundamental methods of image representation in computer graphics are *vector graphics*, which represent an image as sets of coordinates in space called vectors, and *raster graphics*, which represent an image as rows of dots called pixels. In the context of moving images, to use Naimark's distinction, a vector representation is a *first word* technology, and a raster representation is a *last word* technology. The terms do not refer to their order in a timeline, but rather to how they are used.

Chronologically, the "first" and the "last" are reversed in the case of computer graphics history. In the beginning, there was the *pixel*—a portmanteau of "picture element", coined by Fred Billingsley in 1965.[19] The basic concept of forming an image by manipulating component elements predates the name; physical implementations such as mosaic tiles are thousands of years old. But a more useful starting line for animation purposes—when individual picture elements were first applied to moving images—is in 1933, when Claire Parker built the *Écran d'épingles*, or Parker Pinscreen.[20]

Fig. 2. Three lighting setups for an identical simulated Pinscreen frame modeled in Blender.

This mechanical invention was the first device capable of creating an animated sequence using picture elements. Together with her husband, Alexandre Alexeieff, Parker created several short films and title sequences over the next few decades, manipulating pins in small groups with hand tools in an excruciatingly slow process similar to modern pixel art. Critical to the look of the final result was the fact that Pinscreen images were not directly formed from the structure of the pins, but by shadows cast from a nearby light source—in other words, Pinscreen animation established a concept of sculpting images from light instead of solid objects. While there is no evidence the Pinscreen had any direct influence on the earliest digital image technologies, there are uncanny echoes in its specifications. The 200,000-pin resolution of the most common version of the Pinscreen hardware works out to

approximately the same pixel dimensions as the first commercial 512 x 384 *framebuffer* (a block of computer working memory dedicated to holding a grid of pixels) forty years later. Both devices, in their early forms, displayed a sequence of non-realtime images to be photographed on film frame by frame.[21]

The first documented *digital* raster image, not yet moving, came a little over a decade after the start of Parker's Pinscreen experiments.[22] It was drawn in 1947 by Tom Kilburn at the University of Manchester, UK, and consisted only of the words "CRT STORE" on a raster *CRT* (Cathode Ray Tube) monitor in block capitals. Ironically, Kilburn's goal was not to generate an image at all, but merely to test the functionality of the CRT store, an early type of computer working memory. In fact, he was only able to document this pivotal moment with a photograph because the Manchester Baby, the mainframe he was working on—itself the first digital computer—was partially disassembled. CRTs used in this way, to save and recall data, were not even meant to be visible to the end user once installed. Raster graphics arriving first might seem surprising, given the subsequent dominance of vector graphics as both a technology and an aesthetic in the major recognized works of early computer fine art. But the history of vector graphics begins a further decade later, at the end of the 1950s.

While Kilburn went on to a celebrated career in computer science, the author of the first documented vector drawing in computer graphics (and also probably the earliest digital drawing of a human being), remains anonymous.[23] Sometime between 1956 and 1958, an unknown American Air Force radar operator recreated the December page of a 1956 illustrated pinup calendar on IBM AN/FSQ-7 mainframe in Kingston, NY. The anonymous artist did this by writing a program, named *girley1*, that rendered a series of polygonal lines from coordinates encoded on 97 IBM punch cards. While the Baby mainframe had been an academic project hand-built by a team of three people, this mainframe was one of 42 mass-produced units, each the size of a midrise apartment building and costing approximately USD $2.2 billion in 2024 terms. These computers were distributed around the U.S. to create *SAGE* (Semi-Automatic Ground Environment), the first realtime monitoring system for national airspace. The *girley1* card set became part of a standard SAGE diagnostic kit, duplicated and shared by operators for over twenty years afterward, although there is no record of it being seen after 1983. Once again, the only surviving evidence of this milestone turns out to be an informal photograph—in this case a Polaroid taken of a SAGE monitor by airman Lawrence Tipton in 1959.

Already, comparing just these two firsts in digital imaging, we see a sequence that will establish itself in a pattern: raster graphics are more intuitive for humans to manipulate technically, to the degree that the first achievement in this domain happened on a device not even intended for image display. But our imagination outruns the capabilities of raster hardware—while we are waiting for compute power to

catch up, we turn to vector graphics to move forward aesthetically. Why do raster graphics require so much more compute power than vector graphics? Because vector representations only need to store a set of coordinates for a shape, instead of defining every pixel in the shape individually. For a simple example, imagine a very small display area of ten pixels wide and ten pixels high.

Using vector graphics, you could represent a filled rectangle covering half that area as:

```
(0, 0), (0.5, 0), (0.5, 1), (0, 1)
```

That same shape in raster graphics would need to be represented as:

```
1, 1, 1, 1, 1, 0, 0, 0, 0, 0
1, 1, 1, 1, 1, 0, 0, 0, 0, 0
1, 1, 1, 1, 1, 0, 0, 0, 0, 0
1, 1, 1, 1, 1, 0, 0, 0, 0, 0
1, 1, 1, 1, 1, 0, 0, 0, 0, 0
1, 1, 1, 1, 1, 0, 0, 0, 0, 0
1, 1, 1, 1, 1, 0, 0, 0, 0, 0
1, 1, 1, 1, 1, 0, 0, 0, 0, 0
1, 1, 1, 1, 1, 0, 0, 0, 0, 0
1, 1, 1, 1, 1, 0, 0, 0, 0, 0
```

(Here we assume 1-bit pixels, capable of representing only two colour choices, such as black and white.) In a raster representation this information needs to be repeated over and over for each pixel in the grid, making our example above nine times larger than pictured. Yet another increase in size comes when we increase the resolution of the display area—the vector representation does not change, while the raster representation always needs to add more pixels. As a result, the development of raster graphics was first bottlenecked by memory limitations.

The first word/last word pattern repeats itself at the next historical milestone, the first documented hand-drawn computer animation. As for the first computer animation of any kind, there is anecdotal evidence of a *girley2* program for the SAGE that could display an animated sequence of vector images, but no recording of it exists. So the honour instead goes to the Swedish Institute of Technology's *Rendering of a Planned Highway* in 1961, which inaugurated the genre of the first-person architectural visualization.[24] To draw vector images by hand, however, first required the advances demonstrated in Ivan Sutherland's watershed Sketchpad demo in 1963—introducing the light pen, a video camera mounted in a stylus synced to a CRT's electron beam to compose vector graphics in realtime.[8] It became possible to record, in vector coordinates, the movement of an artist's hand as they drew, a radical development that raster graphics was not quite ready to accommodate.

So when Charles Csuri and James Shaffer created their short film *Hummingbird* in 1967, they used vector graphics—brushstrokes drawn with a light pen, then printed on paper using a pen plotter, and finally

photographed on film. The following year, their film was added to MOMA's collection and billed as the first representational hand-drawn computer animation.[6] Meanwhile, storing multiple raster frames of animation data was only possible at very small sizes—for example, the 252 x 184 resolution of the framebuffer that Ken Knowlton used for his contemporary *BEFLIX* (Bell Flicks) programming language experiments between 1963 and 1969.[25]

It is important to note at this stage, in the early 1970s, that vector and raster graphics not only required different logical representations in the computer's memory, but also each required their own specialized display hardware. The CRT monitors attached to these systems were either a raster type, where the electron beam moves in fixed horizontal lines, or a *calligraphic* (vector) type, where the beam could be positioned arbitrarily. In other words, the display limitations of this era essentially required the exclusive use of one technology or the other in a given system.[26] Despite this constraint, the contemporary set of technologies needed to create vector animation—a minicomputer like the popular DEC PDP-11, a drawing stylus, a calligraphic monitor, and a film or video camera to record the output—began to spread beyond a limited number of elite institutional settings. This was a critical moment in the medium's history when, to use Ursula Franklin's terms, raster graphics tools existed primarily in the industrial world of *prescriptive technologies*, where processes are broken down into steps to be carried out by multiple specialists, and no individual specialist can go through every step of the process themselves. Unanticipated aesthetic advances came from individuals and small groups using vector graphics tools because these had begun to enter the domain of *holistic technologies*, processes that a single generalist can master from beginning to end.[27]

Naimark points out that three enormously influential artist-made video synthesizers, using a similar set of equipment and processes, were all completed at this time. The Paik-Abe Video Synthesizer was built in 1972, by Nam June Paik and Shuya Abe; the Rutt-Etra Video Synthesizer in 1972, by Steve Rutt and Bill Etra; and the Sandin Image Processor in 1974, by Dan Sandin and Phil Morton.[28] These video processing systems are arguably not directly connected to the history of computer graphics—they were used to manipulate analog video signal paths, a method of image representation outside our scope here. But, like the Parker Pinscreen forty years prior, they have striking parallels to contemporary digital graphics projects. In particular, the Sandin system used an innovative hybrid of vector graphics and analog video processing. The vector stage of its pipeline was controlled by programs written in the *GRASS* (GRAphics Symbiosis System) language, invented in 1974 by Tom DiFanti—a student of Csuri's at Ohio State University, and who had also started the *ACM SIGGRAPH* (Association for Computing Machinery Special Interest Group on GRAPHics) Electronic Theatre screening series the year prior.[29] In a recurrence of our pattern, the GRASS systems adopted the overall approach of the earlier raster

BEFLIX systems, but by using vector graphics, they became capable of working at far higher resolutions for output to film and tape, and reached a much larger user community of artists.

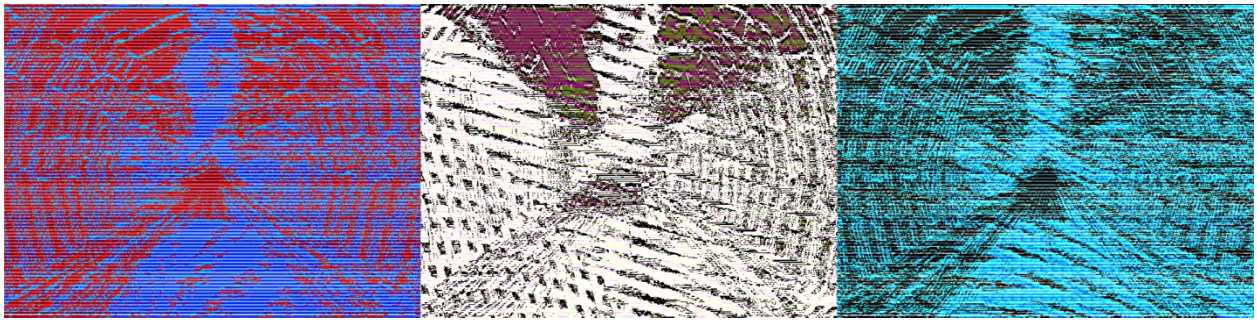

Fig. 3. Example output of a Sandin Image Processor emulation, using a Max patch by Amanda Long.[30]

A further pair of milestones demonstrated a leap in both technical and aesthetic sophistication for hand-drawn computer animation during this period. The first, in 1974, is *Hunger,* a well-known short film produced at the National Film Board and directed by Peter Foldès. It was drawn using a pioneering vector animation system developed at the National Research Council by Nestor Burtnyk and Marcelli Wein (who later received a Technical Achievement Academy Award in 1997 for the project).[11] The second milestone, in 1982, is Rebecca Allen's work in *The Catherine Wheel*, less established in animation history in part because it was not a standalone short film, but a set of title sequences for a live-action documentary of a Twyla Tharp dance piece. Working at *NYIT* (the New York Institute of Technology), Allen keyframed a base 3D model over a video recording of a dancer, and then drew on the surface of the model to create what is probably the first 3D hand-drawn animation—that is, where individual brushstrokes contain Z coordinates, as well as an X and Y. A modern viewer encountering the work might presume it used motion capture—but the first use of motion capture in 3D animation was being developed more or less simultaneously, by Adam Powers, debuting at SIGGRAPH in 1981.[31]

By the late 1970s, the hardware-based separation of vector and raster graphics had largely disappeared, thanks to the wider availability of framebuffers for *rasterization* (converting a set of vector coordinates into a grid of pixels in order to display them on a raster monitor). In 1976, the image processing techniques of analog video could be replicated in realtime raster graphics, albeit at a very low resolution, in the 128 x 128 framebuffer of the Digital Image Processor (aka Vasulka Imaging System), created by Steina and Woody Vasulka, Jeff Schier, and Don McArthur.[32] In 1979, a Norpak Telidon graphics workstation, still based on a PDP-11 variant but now equipped with a 256 x 212 framebuffer, could rasterize a vector image as it was being drawn by an artist.[33] From this point on, cheaper raster display hardware, drawing on economies of scale driven by broadcast television, would increasingly replace specialized vector display hardware. By the 2000s, CRTs were in turn supplanted by *LCD* (Liquid

Crystal Diode) panels, which have no calligraphic display equivalent at all. To pick two memorable examples approximately 25 years apart, Larry Cuba's *Death Star Trench Run* (a GRASS animation running on a PDP-11 in 1977) and Jonti Picking's *Badger Badger Badger* (a Flash animation running on a Windows XP desktop in 2003) both use logical representations of vector graphics, meaning they contain sets of coordinates grouped into shapes. But *Death Star Trench Run* was drawn on a calligraphic CRT monitor, pointing an electron beam at each of its coordinates in turn.[34] *Badger Badger Badger* was converted to pixels and drawn row by row on a raster CRT monitor or a raster LCD panel.

In its first two decades, raster graphics evolved glacially slowly relative to vector graphics, until the greater availability of framebuffers and the feasibility of realtime rasterization allowed the pace of development to dramatically speed up.[35] The first raster image scanned from a photograph was achieved in 1957, and the first multicolour raster image was rendered in 1962. In 1973, Richard Shoup's *SuperPaint* established the enduring user interface paradigm of the raster paint program. But if we were to pinpoint the moment where raster graphics finally developed into a practical animation technology, it might be when Christine Barton and Alvy Ray Smith (one of Shoup's collaborators, and a future Pixar principal) created the first 24-bit paint program, *Paint3*, at NYIT in 1977.[22] (Barton, who designed the hardware, in fact built a system capable of 32-bit colour—24 bits for colour and an 8-bit alpha channel, representing transparency.)[36] The extraordinary utility of being able to manipulate colour and transparency together in raster painting would not be widely understood outside NYIT until 1981, when Marc Levoy (today best known for creating the Pixel phones' computational camera software at Google) developed an alpha-channel-based raster video compositing system for Hanna Barbera.[37] The Quantel Paintbox, launched that same year, definitively announced the arrival of the era of raster animation. Still, even at standard-definition video resolution, this required a phenomenal engineering effort—not software running on a general-purpose computer, but dedicated hardware costing USD $875,000 in 2024 terms. Its 335MB hard drive could hold a little over ten seconds of uncompressed *SD* (Standard Definition) video.[22]

The key factor still limiting innovation in raster images thus far was storage. Even with the resources available to contemporary high-budget feature film productions, there was still no solution large, fast, and cheap enough to store a significant number of high-resolution images. Instead, an animation sequence would be composed as a compact vector representation. Each frame of the sequence would then be loaded, rendered, and rasterized, before being either printed on paper or photographed directly off a monitor—and finally cleared from the framebuffer's memory, with no raster representation saved back to disk. Until the mid-1980s, the photography process mostly relied on mechanical film recorders, often the part of an animation system most prone to failure. To refine this process, David DiFrancesco at the Lucasfilm Graphics Group—forerunner of Pixar, a separate entity from the better-known effects

division *ILM* (Industrial Light and Magic)—developed a laser film printer that could write images directly to celluloid. The group released its test-case short film *Adventures of Andre and Wally B.* in 1984, likely the first high-resolution computer animation produced without a physical camera.[26] A related approach could also work at lower resolutions for video instead of film, outputting an analog signal to tape.

Thanks to these workarounds, the pace of raster graphics development accelerated further, reaching an inflection point in the 1980s. In 1986, Ian Pearson and Gavin Blair (who would later found Mainframe Studios) created Dire Straits' *Money For Nothing* music video on a Bosch FGS-4000 minicomputer, one of the last graphics workstations built from specialized hardware.[38] Three years later, in 1989, Rick Morris and David Silverman created a parody of this same video for Weird Al Yankovic, *Beverly Hillbillies*, using an Amiga 1000 desktop computer—rendering a substantially identical animation on a general-purpose system just as capable and approximately two orders of magnitude cheaper.[39] In 1989, Disney began using Pixar's *CAPS* (Computer Animation Production System) in feature production—although it was initially kept secret, and only acknowledged in 1994.[12] This mid-1990s incarnation of CAPS had 5TB of storage, which meant that manipulating a feature film's worth of high-resolution raster images was now finally feasible, albeit only at the scale of a capital project at a major studio. In contemporary lower-end production, systems like the Newtek Video Toaster, a third-party addon for the Amiga 2000 desktop, still needed to output each frame to tape, without the ability to store significant quantities of footage locally.

Another powerful motivating force in contemporary graphics development was the demoscene, a loose collective of graphics experts that coalesced around the software piracy scene of the 1980s and 1990s.[40] Demoscene art was originally developed for splash screens crediting the distributors of cracked software, in particular for the Commodore 64, but rapidly evolved into a distinct culture of competitive high-performance graphics coding, often under deliberately extreme resource constraints;[41] a classic demoscene exercise is to require the entire artwork to fit in a single executable file not more than 64KB. Demoscene art was distributed globally via BBS networks, although the competition circuits were most active in Europe. Separately from its technical contributions, for example advancing the application of signed distance fields, demoscene work is also useful to consider as a counterexample in the context of hand-drawn computer animation—because of its broad and emphatic rejection of frame-by-frame solutions in favour of procedural ones, initially for practical distribution purposes, but in time for purely aesthetic reasons. (In 2020, the Finnish Heritage Agency became the first national arts granting body to recognize demoscene as a distinct artistic discipline,[42] followed by counterparts in Germany, Poland, the Netherlands, and most recently Sweden.)[43]

Intentional creative constraints aside, by the mid-1990s, the storage situation had improved enough for the *DV* (Digital Video) format, at 1GB per five minutes of footage, to be ingested and edited on consumer hard drives. Just a few years later, however, SD digital video started to be replaced by *HD* (High Definition) video formats. This increased storage requirements by a factor of six, completely impractical to distribute online using the 0.05mbps modems of the time. Once again, commodity hardware was not up to the graphics tasks required of it. As a result, animation in the late 1990s and early 2000s returned to vector graphics in order to output to HD and stream moving images on the web. The most famous example of this is the *Flash* vector animation software, created by Jonathan Gay, Charlie Jackson, and Michelle Welsh in 1995, perfectly suited to hand-drawing compact vector frame sequences for viewing over a very low-bandwidth connection.[44] A related contemporary approach in feature filmmaking used *rotoscoping* (the process of matching animation frame by frame to live-action footage).[45] In 1997, Bob Sabiston created *Rotoshop*, a tool that allowed an artist to ingest relatively low-resolution live-action footage shot on DV, draw and keyframe vector animation on top of it, and output a high-resolution rasterized video to HD and film. The Rotoshop software itself was never commercially released, but was prominently used in the feature films *Waking Life*, in 2001, and *A Scanner Darkly*, in 2006.[46]

By the end of the 2000s, *SSD* (Solid State Drive) storage began to replace mechanical hard drives, decisively altering the arms race between raster image size and storage capacity. With far greater reliability and speed, working with large numbers of high-resolution, high-bit-depth image files on commodity storage hardware was finally feasible. The increase in storage bandwidth also altered the way images were acquired—in terms of the sheer amount of data they could write to disk per second, even early versions of solid-state recording media vastly outstripped HDCAM, the last major videotape format. Increasingly, live-action digital imaging was no longer restricted to an intermediate medium in between film or tape input and output (memorably called a "Digital Sandwich" by editor Walter Murch).[47] By the 2010s, feature films were routinely commercially distributed entirely digitally, using *DCP* (Digital Cinema Package) files. As virtual cinematographer Christopher Prevoe describes it, "In 2008, Canon's 5D Mk II camera broke the agreement among camera manufacturers that cinema-quality digital video cameras would cost at least [CAD] $300,000 [$450,000 in 2024]. Ten years later, you have Red, you have Blackmagic, the price of a cinema camera is $30,000 [$38,000 in 2024], and Arri has had to come down to match. About five years ago, the transition happened—practically all film and television was now being shot on video, and a system a hundred years old was suddenly over."[48]

Around the same time, the focus of hardware improvements in computing performance abruptly moved from the *CPU* (Central Processing Unit) to the *GPU* (Graphics Processing Unit). While specialized

approaches to graphics processing are at least as old as integrated circuits themselves, a market for third-party add-in graphics cards did not arise until the microcomputer era. (The term itself was coined almost accidentally in 1994, used in Sony's advertising copy for the Playstation 1.)[49] The breakthrough product that ended up blurring the lines between GPUs and general-purpose computing was Nvidia's Geforce 3 card in 2001, which introduced user-programmable shaders. Even here, in its earliest mass-market implementation, hardware shader programming already embraced both vector graphics (vertex shaders) and raster graphics (fragment shaders)—with the first GPU model offering a recognizably modern implementation of both being the ATI (now AMD) Radeon 9700 in 2002.[50] [51] Other types of programmable shaders followed, for example geometry shaders, which generate new geometry from scratch instead of manipulating the vertices of existing geometry. But the most important later variety for our purposes is the compute shader—which has no graphics output at all, but allows arbitrary floating-point operations—arriving in 2006. This was again an Nvidia development, named CUDA (Compute Unified Device Architecture), and likely the determining factor in allowing ML development to move onto GPUs in the 2010s.

In other words, having thus far emphasized the historical differences between vector and raster representations, we now find it difficult to maintain a meaningful distinction between them past the 20th century. Modern graphics applications freely combine both vector and raster representations as the need arises—in nearly all 3D animation applications, like Maya or Blender, a scene is composed in vector graphics, then rendered from a camera viewpoint as a series of raster images. Many 2D animation applications, like North American industry standard Toon Boom Harmony, follow this basic approach as well. And even most raster paint programs, like Photoshop or TV Paint, internally represent brushstrokes in progress just as Sketchpad did—vector shapes—before rasterizing them one stroke at a time. So instead of continuing to parse out vector and raster distinctions, we should more precisely isolate and describe the elements of each that remain in tension.

In modern computer graphics, the conflict between the vector and raster approaches has evolved into one between *non-photorealism and photorealism.* Photorealism in graphics is usually defined narrowly, as a single rendering method—modeling the behaviour of rays of light in a scene, sampling the end points of the rays, interpolating between the samples, and rasterizing the final colour values as a grid of pixels.[52] An equivalent definition for *NPR* (Non-Photorealistic Rendering) might simply be, "everything else". Julie Turnock argues that a consideration of photorealism in computer graphics should interrogate the deliberate stylistic choices made at the time of the initial 1980s raster breakthrough—in other words, the predominant live-action film aesthetics of the late 1970s. "Specifically, that means a muted color palette, lens flares, handheld cameras, and available light, which marked the more naturalistic 'New

Hollywood' filmmaking styles of, for example, Robert Altman, Hal Ashby, Terrence Malick, and Monte Hellman."[53]

Alternatively, as seen from the viewpoint of one of those New Hollwood-era live-action filmmakers, the opposing binary might not be vector and raster, or non-photorealism and photorealism, but a distinction that predates computer graphics entirely—*optical shots* (effects created with mechanical optical printers in post-production) and *process shots* (in-camera effects filmed simultaneously with live action). Visual effects in the silent film era were almost exclusively process shots. Live-action filmmakers have always gravitated to in-camera solutions when they were available—both Méliès and Kubrick used front and rear projection shots more frequently than optical composites. Linwood Dunn and Cecil Love developed the first retail optical printer, the Acme-Dunn, in 1942, but what was then called *optical animation* still required further refinement over the next thirty years—for example replacing the expensive 70mm film format used by optical printers with the much cheaper VistaVision format, and finally using computerized motion control to allow optical compositing with a moving camera.[53] Another significant development supporting both optical and process effects came in 1968, when Jim Songer created the first modern video assist—using a beam splitter to show the exact framing of a final film image on a video monitor. The specific challenge Songer solved was how to divert sufficient light from the lens to the video tube without forcing a change to the lighting setup for the main film camera. (He used a special high-resolution Saticon tube which had 1,600 lines of analog resolution, instead of the standard NTSC 525 or PAL 625, and a fibre-optic connection to expose a smaller, brighter standard-definition image using only the centre portion of the tube.) The ability to accurately preview the recorded image in realtime rapidly enabled more complex optical compositing approaches, as well as new creative uses of forced perspective in-camera.[54]

Turnock points out that, once optical technology finally began to replace process technology in the 1980s, the three primary creative roles of auteur cinema colluded to exclude the new role of *VFX* (Visual Effects) supervisor. "The so-called 'creative triangle' (director, cinematographer, and art director) did not see much advantage in becoming a square…To demonstrate Lucas's and other producers' success in repressing the rise of the star effects artist, it is striking that if one can name a special effects artist today [2015], it is likely the same names *Future* [magazine] lists in 1978: Trumbull, Dykstra, or Harryhausen."[53] George Lucas in particular insisted the VFX supervisor had no creative role, and frequently clashed with first Douglas Trumbull and then Trumbull's protege John Dykstra, who also faced resistance when they later tried to break into directing themselves. In the end, VFX artists did not tend to enjoy the status and opportunities for peer recognition of other unionized film craftspeople. Adam Beckett, who did hand-drawn effects animation for the original 1977 *Star Wars*, recalled being criticized for producing work that

looked “too individualistic”—contemporary ILM reserved optical animation jobs for outside contractors, who often go unmentioned in its official histories.

It took several more decades of development, but in the late 2010s the VFX industry finally found its way back to traditional process shots. *Virtual production* (using realtime raster graphics to create in-camera final visual effects) in a sense returns VFX to its roots. ILM debuted the StageCraft system in 2019, an airplane-hangar-sized 270-degree LED wall fed by a game engine backend, rendering each new frame of the environment from the live-action main camera’s realtime perspective. The end result is high-resolution imagery seamlessly integrated into a live performance, complete with perfect real-world lighting and reflections courtesy of the video playing on the enormous screen.[55] StageCraft, though undoubtedly impressive, also bakes in a couple of important assumptions that serve as constraints—that the realtime graphics will function as a background set for the live actors and props in the foreground, and also that the goal of the whole system is a photorealistic rendering of a practical set design. The 2019 TV series *The Mandalorian* was the first production to use the StageCraft system, and in its first season 70% of all its VFX shots were done in-camera. According to Prevoe, “You don’t have to worry about clothing, or glasses, or bottles of water. You can have a main character who’s a big mirrored disco ball. And the hierarchical, coordinated stages of the VFX process happen before the shoot, preparing the assets. The final creative decisions are being made during the shoot by the director and DoP.”[48]

Beginning in the late 2010s, ML-based rendering techniques introduced a second major change in experimental graphics technology, in particular *diffusion rendering*, which can generate structured photorealistic image output from random noise conditioned by very large training corpora of billions of source images.[56] The first of these systems to gain prominence was *DALL-E* (a portmanteau of “WALL-E” and “Salvador Dalí”), made by the OpenAI research consortium. In 2018, this group created the first of the *GPT* (Generative Pre-trained Transformer) systems, each major release of which has boasted best-in-class advances in text processing. Next, they released the *CLIP* (Contrastive Language–Image Pre-training) system in 2021, which matches text queries with images. The same year, those two components were combined to make the first of the DALL-E systems, which synthesize a novel image from a text query by first generating a vast array of image collages based on the material in a training corpus, then performing a secondary search within that output for the best solution.[57] 2022 then saw the initial release of StabilityAI’s open-source Stable Diffusion model series, a generally promising avenue for independent research-creation projects and as of 2024 on its third major version.

ML image processing approaches like this have enormous implications for all types of rendering, photoreal and non-photoreal alike. On the photoreal side, higher-resolution images can be generated

more easily by taking fewer initial samples and interpolating between them with greater accuracy and speed. The clusters of high-frequency detail we find to be pleasing evidence of quality in photorealistic imaging can also be implemented as a post-processing effect, given an ML system trained on high-resolution source material that can make an approximate match to the test image. On the non-photoreal side, things get even more interesting—the implicit rules governing systems like Caroline Chan's *informative-drawings*, also based on CLIP, mechanistically resemble processes of human perception, and can be used to generate unprecedentedly human-like hand-drawn illustrations.[58] These systems are also curiously difficult to simply categorize as raster or vector, typically following keyword matches representing a vector path in multidimensional space, but also using the matching keywords to retrieve raster pixel data stored in their models.[59]

Perhaps at this point we can take a step back from the binaries of raster/vector, photoreal/non-photoreal, and process/optical and propose a different, three-part frame. Dziga Vertov coined the term *Kino-Eye* in 1929 to describe the mechanical capture of indexical reality by the camera.[60] To complement this Lev Manovich in 1995 proposed an opposite, the *Kino-Brush*, to describe the digital manipulation of individual frames.[61] Manovich, at least at the time, conceived of this alteration of an original indexical image as a primarily, if not exclusively, manual process. Considering only these two contexts recalls a profound aesthetic debate, older than cinema itself—as Walter Benjamin might frame the question, a conflict between machine-made and hand-made images.[62] But splitting the difference between the Kino-Eye and the Kino-Brush modes is a third context that I would like to introduce as the *Kino-Stomach* mode, after Steina Vasulka's 1985 characterization of digital video processing as closer to digestion than vision.[63] This is the automated processing of images, where the artist determines the rules in advance and does not manually intervene frame by frame. Such systems arguably have a greater conceptual connection to the analog and early digital video processing hardware of the 1960s and 1970s than they do to either camera-original or handmade images.

For most of computer graphics history, the raster-based Kino-Eye mode, generating pixel-based images by procedurally modeling the behaviour of light, has dominated popular interest and infrastructure investment. The vector-based Kino-Brush mode, constrained by the Baumol Effect of artisanal production costs, only infrequently achieves a similar industrial scale through ingenious pipeline design, as opposed to through simple labour exploitation. But the Eye appears to be in the process of yielding its pole position in graphics to an ML-powered Stomach. As we will see, Latk is an attempt to develop and sustain work using Eye, Brush, and Stomach approaches in combination—but as a hand-drawn animation tool, it still fundamentally prioritizes the Brush. Next, we will examine the evolution of the hardware tools that have so far allowed us to explore the Kino-Brush route in 3D.

3. A Taxonomy of Tools for Drawing in 3D Space

> "Manual construction and animation of images gave birth to cinema and slipped into the margins…only to re-appear as the foundation of digital cinema. The history of the moving image thus makes a full circle. Born from animation, cinema pushed animation to its boundary, only to become one particular case of animation in the end."
> —Lev Manovich (1995)[61]

The mid-20th century saw the convergence of two lines of technological development, both of which grew from seeds planted in the 19th century. The first line originated with the stereoscope, in 1838. The second originated with the autopen, in 1888. Combined, they allow a user with stereo vision to control a 3D pointing device—a concept first described in science fiction in 1935, realized experimentally in 1968, and finally available in a mass-market consumer product in 2016.[64] As a result, tracing the early history of these devices requires first pursuing two separate taxonomies, before following the results of their cross-pollination.

3.1. Stereoscopic Viewing Devices

> "If you get into an area much larger than [a two-hundred-seat theatre]…you confront the problem of what is and what isn't 3-D. You don't see much 3-D beyond twenty or thirty feet, so the effect would be lost if you had to sit very far away from the image. Either you'll have a projected image that's like a person on a stage where about a hundred people can observe him, or you'll have a personalized box like a TV set, or a hood over your head."
> —Gene Youngblood (1970)[65]

The foundation of all our experiments with 3D pointing devices, both real and virtual, is *stereopsis*, or the presentation of different, yet spatially coherent, images to the right and left eyes, which our brain can "fuse" to generate a model in three dimensions. As an optical illusion, artificial stereopsis involves presenting the eyes with synthetic images that maintain a similar spatial coherence.[9] Stereo image pairs are by far the technologically simplest way to achieve this result, compared to the other three established methods of measuring depth. Structured light projection and time-of-flight measurement require lasers, while realtime lightfield synthesis was computationally impractical before modern GPUs. As a result, the use of stereo pairs for this purpose predates the other options by more than 120 years.[66]

3.1.1. Stereoscope (1838)

Somewhat counterintuitively, a hand-drawn line, rather than a photograph, marked the first known

exploration of artificial stereopsis. The *stereoscope*, first demonstrated by Charles Wheatstone in 1838, was a handheld stereo viewing device using a pair of lenses and mirrors, into which two separate images were inserted—already quite similar to the design of a Google Cardboard nearly 180 years later.[9] The first mass-produced portable camera would not be available until 1839, which is why Wheatstone drew his stereo image pairs by hand, estimating the correct offset to make the figures appear separated in space. (Photography technically begins with Joseph Niépce in 1826, but his *heliograph* device was designed to duplicate hand-drawn art for lithography, so it was still impractical to make a stereo image pair with a camera before Daguerre.)

Stereo image viewers continued to co-evolve with photography—by 1856, they had arrived at the same optical principles as a modern VR headset. By 1933 there was enough demand for stereo photography to support a mass-produced stereo viewer, the Tru-View, which took standard 35mm slide film. And probably the most popular stereo viewing device ever created is one we would still recognize today—the 1937 View-Master, which used custom cardboard wheels with colour 16mm images. Its classic top-loading design, from 1944, did not substantially change until a digital version was attempted in 2015 (and discontinued in 2019).[67]

3.1.2. Sensorama (1962)

As for moving images, the first documented exhibition of a stereo movie (by Robert Elder, in Los Angeles) happened in 1920.[66] However, it would be a further 40 years before stereo movies would be displayed in something we might recognize as VR—on a screen close to the eyes, encompassing a user's entire field of view. Morton Heilig's Sensorama, an entirely analog installation, accomplished this with a fixed-position headset called the Telesphere.[68] The 180-degree stereo film most commonly exhibited with the device documented a motorcycle trip through Manhattan (sometimes misrecorded as Brooklyn in contemporary accounts—the film itself has only recently been restored and made available to view).[69] The Sensorama was a short-lived commercial failure, possibly due in part to Heilig's ambitious efforts to artificially reproduce smells, along with picture and sound. But true, free-viewpoint VR headsets were not far behind.

3.1.3. Sword of Damocles (1968)

Ivan Sutherland's Sword of Damocles headset is considered the first true VR system, meaning the user could roam freely around a controlled environment while wearing it. However they could not necessarily roam *safely*—the headset's name comes from the potentially lethal unsupported weight of its two *CRT* (Cathode Ray Tube) monitors, and the resulting safety cables upon which the user's life depended. (Fortunately, there is no evidence anyone was actually injured while wearing the headset.)[70]

Constructed at the University of Utah, it was driven by a PDP-1 minicomputer and used ultrasonic sensors to track the user's head position. It also used a mirror to reflect the CRT displays onto a transparent viewscreen, arguably making it the first *AR* (Augmented Reality) headset as well. (This depends on your definition—a user would not have been able to focus their eyes within the display area, as with modern waveguide-based optical AR headsets like the HoloLens and Magic Leap.) There were initial plans to use the Sword of Damocles for medical simulation projects, but the high level of risk involved made this impractical. The design approach would later be successfully revisited by Marc Bolas's high-end FakeSpace Boom CRT-based headset (1992), which used a sturdier reinforced mechanical arm for support.[71]

3.1.4. VPL EyePhone (1984)

Jaron Lanier's *VPL* Research (Virtual Programming Languages) group made the first major attempt to commercialize VR headsets, and these efforts would prove to have impacts far beyond their company's relatively brief years in operation. Crucially, VPL was arguably the first to combine a headset, the EyePhone (no relation to the later Apple product), with a hand-tracking controller, the DataGlove (1987), and a full-body tracker, the DataSuit (1989), which provided 6DoF tracking for the whole system.[22] Unfortunately, once the expensive package was fully developed, it turned out to be a swift commercial failure—one of many at the time. The intervening two decades of stagnation in VR technology investment would later come to be known as the VR Winter.[72] The EyePhone was discontinued in 1990, although the DataGlove received a second chance at life when it was licensed as the Nintendo Power Glove in 1989. It was reimplemented in consumer hardware, with two bits of precision per finger instead of eight—only to fail in the market yet again.

However, even as Power Gloves were languishing on toy-store shelves, a separate line of XR technology was finally reaching a mass audience, after nearly a century of parallel development.

3.2. 2D Pointing Devices

> "Well, the light pen is a crude drawing instrument, it's true. You can't do many subtle things, the resolution is low, and the way you operate you're always stopping, waiting for…the computer to accept your line, or the accuracy always seems to be off, but…I discovered that working with program languages to produce graphics is rather hopeless. They're really designed for playing with numbers… I think it's a waste of time in computer graphics or music. My explorations in computer language led me back to conventional animation, back to the paintbrush—a computerized paintbrush."
>
> —John Whitney Sr. (1970)[65]

The problem of *bandwidth* limitations—the quantity of information that can move through the system in a given time—haunts all computer interface discussions, not merely in the context of digital hardware, but also in the bandwidth limitations of the human nervous system. Gary Langolf, in 1976, first attempted to quantify the bit depth available to the neural controls for an elbow (10 bits), a wrist (23 bits), and a finger joint (38 bits).[73] While Langolf's absolute measurements are no longer current in neurology, as a general relative principle of user interface development, this idea has been carried forward by modern XR hardware designers like Savannah Niles.[74] Considered as a single point in space, the strongest influence on the motion of each joint is the next one above it in the hierarchy—which is why hand tracking can quickly become tiring, because hands are largely steered by a fairly imprecise elbow joint. A fingertip or a wand controller, using the higher bandwidth of the wrist joint, can be more precise than hand tracking. And a stylus tip, levered by multiple finger joints, represents an even more significant increase in precision, ideal for drawing and sculpting.[75] [76]

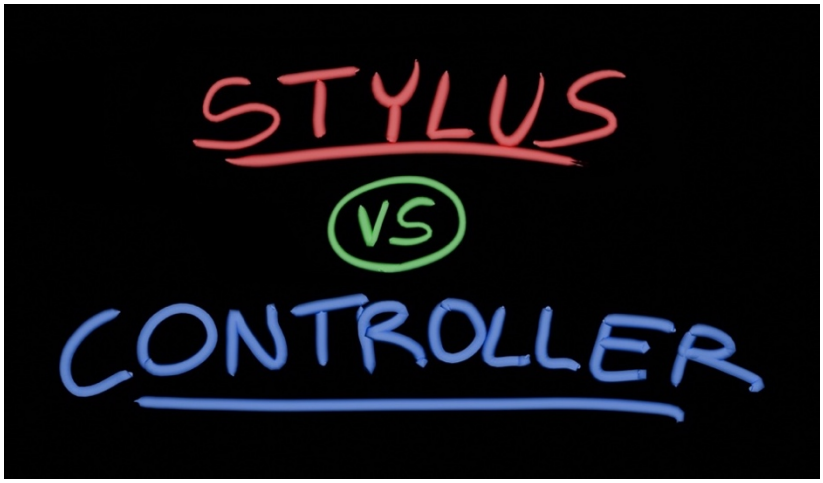

Fig. 4. Subtle differences of midair hand lettering in Open Brush using a stylus and using a controller.

3.2.1. Autopen (1888)

Like Wheatstone's drawings, the autopen was an improvisation created to bridge a technological gap with no existing solution: in this case, the transmission of legally verifiable signatures over long distances. Patented as the Telautograph by Elisha Gray in 1888, it used potentiometers to record electrical impulses in two channels, representing the position coordinates of the stylus, and transmitted them by telegraph to a mechanical plotter which drew a duplicate image.[77] (The generic term *autopen* dates to the 1940s.) The autopen was not the first device to use the concept of representing an image in a two-channel electrical signal—fax machines, or "electric printing telegraphs", were already forty years old. But it was the first that could reproduce an image in realtime, as it was being drawn.

By the early 20th century, autopens were surprisingly common pre-digital office equipment, having

evolved along with communications technology to transmit a signal over telephone lines. In a macabre detail, Dinah Lipschitz, a survivor of the Triangle Factory Fire in 1911, successfully used an autopen connected to a plotter within the building to alert workers on other floors.[78] And possibly the best available document of an original Telautograph in action is the 1956 film *Earth vs. the Flying Saucers*, where one is used as a practical effect to represent the terminal interface of a fictional computer. Versions of the analog Telautograph design remained in production until 1999, still used for witnessing legal signatures.

3.2.2. Light Pen (1951)

The first devices to bring the autopen into the digital realm ended up using a radically different approach. The light pen was originally the light gun, an input device developed by Robert Everett at MIT's Lincoln Laboratory for the SAGE air-defense radar system. The light gun consisted of a grip, a trigger button, and an optical sensor covered by a mechanical shutter. When the trigger was pressed, the shutter opened, and if light hit the sensor, then the SAGE system could guess the position of the device based on the known timing of the CRT's scanning electron beam.[79] Wes Clark, arguably the primary inventor of the digital drawing tablet, redesigned the light gun in a stylus form factor to be used with the experimental TX-2 minicomputer. In 1959, the TX-2 was licensed to DEC as the PDP-1—the first commercially available computer capable of realtime graphics, and eventually the most popular minicomputer ever made. It was on this fortuitous platform that the first true pen-based digital drawing application was created, Ivan Sutherland's Sketchpad (1963).[8]

While revolutionary at the time, the light pen had two serious limitations. One problem was simply available compute power—the PDP-1 could not track the pen position continuously, as would be needed for freehand drawing. Simply performing the location lookup task for the pen on each screen refresh consumed 10% of the PDP-1's CPU resources. Instead, strokes in Sketchpad were drawn as "rubber-banded" lines, adding one point at a time per trigger press. The second problem would not become apparent for several decades: light pen hardware required precise synchronization with a CRT monitor of known dimensions and refresh rate, for example the IBM 2250 Display Console or the Vector General 3D Display ("3D" here referring to built-in early GPU hardware, not stereo viewing capability). Light pens therefore faced increasing compatibility issues in the microcomputer era, as a small number of high-end vector monitor models were supplanted by less expensive third-party raster monitors of varying specifications.[22]

3.2.3. Digital Plotter (1953)

While the plotter itself is solely an output device, and does not strictly fit into a taxonomy of pointing

devices, the development of digital plotters and drawing tools overlaps considerably. When Sketchpad was first demonstrated, its display hardware was still considered experimental, and computers capable of realtime video output were a decade away from wide commercial availability. Instead, computer-controlled plotters became a critical bridge technology for image-based digital art in the 1950s and 1960s. A plotter is a mechanism holding a tool, usually a pen, that can be driven on one or more axes. Forty years before the first television demonstration in 1927, the Telautograph used an early mass-produced analog plotter on the receiving end. In 1953, the Remington Rand typewriter company developed the first digital plotter for the UNIVAC mainframe system, and it quickly found applications beyond its intended purpose of printing text and statistical graphs. Nearly all significant works of visual art created on a computer before 1970 were output on a plotter, including works on film—either printed on paper and photographed separately, or exposed directly using a moving light source.[80]

For example, in 1960, Vera Molnár, arguably the first recognized artist to produce plotter work, began writing pseudocode procedural instructions for what she termed a *Machine imaginaire*—what we would now call a virtual machine. Later named Molnart, this system defined a set of geometric primitives and operations that could rotate, deform, erase, replace, or randomize them. In 1968 she gained access to an IBM System/370 mainframe driving a plotter at the Sorbonne, and began translating Molnart artworks into Fortran and later Basic. It can be difficult to grasp the challenge of this undertaking without the additional context that the process required using "blind computing", or mental visualization alone. The ability to preview instructions on a CRT monitor before committing them to paper was not added to the Molnart software until 1974. (In 2022, at the age of 98, Molnár represented France at the Venice Biennale with a new series of procedural plotter works written in Fortran.)[81]

3.2.4. Light Pencil (1959)

The light pencil, perhaps the single most popular hardware development in the history of digital 2D drawing technology, turns out to be confusingly named. It is not functionally similar to a light pen, and in fact uses no optical hardware at all. The original light pencil, designed in 1959 by the RAND Corporation for the IBM *DAC-1* (Design Augmented by Computer), returned to the autopen's electrical switching approach. (Digital autopen hardware already existed as of 1957, but not as a general-purpose computer peripheral.) The DAC-1 design used a transparent resistive screen placed over a vector monitor, reporting position coordinates where the stylus made contact.[22]

The DAC-1's tablet ended up having a much longer life than its host computer system, repackaged in 1964 as a standalone peripheral without a screen, first known as the RAND Tablet and then as the Grafacon.[82] Not limited to IBM systems, these were also compatible with DEC PDP minicomputers. By

1981, when the Summagraphics company became the first dedicated tablet manufacturer, the cost of a tablet peripheral had dropped from USD $18,000 ($185,000 in 2024) to below $600 ($2,100 in 2024). Summagraphics was responsible for several key innovations that would become standard features, including moving a small CPU into the tablet itself to speed up tracking calculations, and electromagnetically detecting the proximity of the pen to allow cursor control without touching the tablet. This enabled a critical UI change for natural drawing interaction—making contact with the tablet surface itself could count as a "click", instead of requiring a separate button press.[22]

3.2.5. Quantel Paintbox (1981)

The Quantel Paintbox saw over a decade of remarkable success in the broadcast graphics field. It was a compact minicomputer with a 300MB hard drive, a best-in-class 24-bit GPU, and a pressure-sensitive drawing tablet—an innovation that became standard for all drawing tablets that followed.[22] All of its painting functionality was implemented in proprietary hardware, for unprecedented speed. There were only a few hundred ever sold, starting at USD $250,000 each ($875,000 in 2024), a dozen of which are known to remain fully operational today. The Paintbox also inaugurated the relatively brief era of the elite broadcast graphics operator—a tiny number of staff positions at post houses were passed from master to apprentice, and typical pay was USD $500/hour ($1,750/hour in 2024). The original Paintbox ended production in 1993, although versions of it were made for commodity desktop hardware until 2002.

Ironically, Quantel's single most important contribution to computer drawing technology turned out to be a legal one. In 1997, they lost a lawsuit against Adobe over Photoshop—which they claimed was an infringing software port of Paintbox hardware—thanks in large part to expert testimony from former Pixar principal Alvy Ray Smith. Smith argued that Quantel's hardware implementations of common graphics concepts did not equate to ownership of the concepts themselves. As evidence he demonstrated his own work as co-creator of the pioneering paint program *SuperPaint* in 1973, first built in hardware and establishing conventions that were borrowed in nearly every subsequent software paint application.[5] This decision is considered to be a key precedent protecting the software-based implementation and extension of prior hardware innovations.[83]

3.2.6. Wacom Tablet (1983)

The early 1980s saw an explosion of commercialized graphics tablet technology based on both light pen and light pencil concepts. The light pen approach quickly ran into trouble in the microcomputer era, however, because each pen design was tied to specific CRT monitor hardware, while monitors were increasingly becoming commodity items that were sold separately from desktop computers. The light

pencil's wire grids had no such inherent limitation, and by the mid-1980s were being used in popular consumer products like the Apple II's KoalaPad (1984). But Wacom, founded in 1983, was the company that would finally turn light pencil-style graphics tablets into a mainstream computer peripheral.[84]

Wacom's anomalous success—across the entirety of our three taxonomies, it is the only maker of pointing devices that still exists, and continues to operate as an independent company, *and* also continues to lead its product category—can attributed to three key factors. First, the Wacom tablet was the first mass-market design to move all powered electrical components, including sensors, into the tablet itself, with an unpowered, untethered stylus simply moving a small magnet around an electromagnetic sensing grid. Second, their pens came with customizable controls—typically an additional pressure-sensitive "eraser" tip on the back of the stylus, a tilt sensor, and two side buttons, all user-configurable for specific applications. And third, they were chosen as the vendor for the Pixar CAPS system's stylus, which gave them immediate credibility in the highest echelons of paper-based classical animation.[13]

By 1994, an entry-level Wacom tablet, the ArtPad, was available for under USD $200 ($400 in 2024).[85] Drawing tablets were finally mainstream peripherals—Bill Clinton even signed the 1996 U.S. Telecommunications Act with a Wacom pen.[86] (Ironically, given the early and consistent application of 2D pointing devices for document signing, it was not until 2005 that U.S. federal law would finally establish that an autopen signature was as valid as a manual signature.)[87] In 2001, the Wacom Cintiq model added an LCD screen built into the tablet itself—in a sense, returning the light pencil to its original 1959 design.

3.3. 3D Pointing and Tracking Devices

> "It is becoming increasingly easy to bring the body directly to digital form via stereoscopic immersive displays and tracked input devices. Is this space a viable one in which to construct 3D objects? Interfaces built upon [2D] displays and 2D input devices are the current standard for spatial construction, yet 3D interfaces, where the dimensionality of the interactive space matches that of the design space, have something unique to offer…What we see is a space, not exactly like the traditional 2D computer, but rather one in which a distinct and different set of operations is easy and natural."
> —Steven Schkolne (2003)[88]

3.3.1. Lincoln Wand (1966)

Of the many computer graphics innovations to emerge from Lincoln Laboratory in the 1960s, the first

3D pointing device is one of the less well-known.[89] The Lincoln Wand, created by Lawrence Roberts in 1966, used ultrasonic sensors for 3D tracking in a stylus form factor, and proposed to replace both the light pen and light pencil approaches in 2D as well. While that did not come to pass, one key feature of the design, the ability to switch between 3D spatial tracking and 2D surface drawing modes, did become a standard feature in later 3D stylus concepts.[90] And the Lincoln Wand's ultrasonic approach is only the earliest documented example of dozens of implementations of 3D tracking methods that followed.

Some methods were variations on the original, like Charles Csuri's Sonic Pen, developed at OSU in 1970. This replaced ultrasound with less-expensive acoustic tracking—the pen itself emitted sound, measured by three conventional microphones, and a position was obtained by triangulating the results.[70] The SensAble Phantom (1994) used a mechanical solution, a haptic feedback pen mounted on a 6DoF articulated arm.[91] The remarkable Dinosaur Input Device (1991) deserves special consideration, a collaboration between ILM, Pixar, and Tippett Studios. Purpose-built for 3D digital character animation, it used a metal armature rigged with optical encoders to detect relative changes in position. The approach was never widely adopted, thanks to the added labour cost of expert stop-motion animators and armature machinists, although Tippett Studios continued to develop the rigs at least until 1997.[92]

Over time, general-purpose tracking hardware that could be customized for many applications fared better than specialized solutions. The Polhemus Fastrak and the Ascension Flock of Birds (1995) used electromagnetic sensors, which made it possible to create larger tracking volumes, albeit noisy by modern standards, and strongly affected by metal objects in the vicinity.[93] Passive optical motion capture systems, most prominently Vicon (1984) and later Optitrack (1996), are still widely used at present, and under controlled conditions can deliver the best overall tracking quality commercially available. This approach uses retroreflective foam balls with infrared spotlights and high-framerate infrared cameras (in modern systems, typically 240 to 360 fps). Distributed hardware in each motion capture camera handles blob tracking on the reflected areas, and streams 2D points to a central server that triangulates the resulting 3D points.

However, in the 1990s, all these varied methods, when applied to the challenge of 3D pointing, were usually still operated "blind". Much like early 2D graphics technology required mental visualization without any monitor at all, users of 3D graphics technology up to this point often relied on 2D monitors, with the user unable to directly guide a tool to an absolute point in 3D space. Complete 6DoF-capable VR rigs remained fairly rare and expensive at this time—for example, USD $30,000 for a Virtuality brand turnkey system in 1990 ($73,000 in 2024).[94] 3D pointing technology still lacked an accessible stereo viewing technology to guide it.

3.3.2. CAVE (1991)

The first *CAVE* (recursively, "CAVE Automatic Virtual Environment"), created by Carolina Cruz-Neira at the University of Illinois in 1991, used four 1280 x 1024 stereo projectors and active-shutter 3D glasses, each one driven by an individual SGI VGX workstation, combined with a Polhemus electromagnetic tracking system streaming data to a fifth SGI workstation. Her second CAVE version added two projectors, increasing the total number of SGIs required to seven.[95] Even a modern equivalent of these rigs, each component orders of magnitude lower in price, would be difficult to describe as "accessible" technology. However, the CAVE still broke an important barrier in the drive toward practical 3D pointing devices: it used a collection of commodity hardware stitched together by open standards, instead of a closed proprietary ecosystem dependent on one manufacturer. This flexibility is what allowed CAVE-based systems to thrive during the VR Winter period. Over time, less expensive substitutes were found for some of the priciest components—for example, using multiple stereo monitors instead of projectors, sometimes referred to as a *fishtank* approach.[96]

In a typical VR headset, the scene being displayed is rendered from the viewpoint of two side-by-side virtual cameras placed a 3D world, displayed on two corresponding side-by-side physical screens. The key insight of the CAVE was that *any* arbitrary configuration of virtual cameras could be used to display images on *any* corresponding arbitrary configuration of physical screens. So long as tracking hardware can obtain the user's position, and the number of camera views that needs to be rendered stays within the limits of available compute power, the subjective effect of immersion can be maintained. Another long-term problem that the CAVE approach solved was reading text and small UI elements, because video projectors could handle far higher-resolution images than the miniature CRT displays used in contemporary VR headsets. (Reading text on the small displays in an XR headset continues to present a formidable challenge even today, with the resolution of a modern consumer headset about eight times the highest-end 1990s equivalent.)[97]

Multiple CAVE drawing projects established key interface conventions that persist in modern XR—in particular assigning brush and palette tools to dominant and non-dominant hand controllers, and what's now called *world scale* two-controller navigation for gradually making changes of translation, rotation, and scale without disorienting the user. Examples include 3DM (1992), HoloSketch (1995), Surface Drawing (1999), Cavepainting (2001), Freedrawer (2001), and Drawing on Air (2007).[96]

A rare attempt to package and sell a CAVE drawing solution came in 1997 from IMAX—primarily an analog film projector vendor at the time, but supporting innovative art installations as well as conventional theatrical exhibition, and cited as an inspiration by Cruz-Neira in her original paper.

Likely the first commercial volumetric hand-drawn animation software, *SANDDE* (Stereoscopic ANimation Drawing DEvice) used a Wacom stylus modified with an Ascension 6DoF ultrasonic tracker, together with polarized stereo projection.[14] Development was led by Roman Kroitor at IMAX and Munro Ferguson at the NFB.[98] In 2007, SANDDE was spun off in a separate company, Janro Imaging, and tasked with developing a less expensive fishtank version for lower-end stylus hardware. Failing to find a market, the project was finally shut down in 2012—ironically, the same year as the Kickstarter campaign that launched the Oculus DK1 VR headset and a subsequent fresh wave of interest in volumetric creation.

3.3.3. Wiimote (2006) and Razer Hydra / Sixense STEM (2011)

With the notable exception of the Wacom tablet, of the purely commercial projects in our taxonomies so far, we see a lot of rapid failures. Pointing devices that benefited from a connection to large public research projects, like MIT's TX minicomputer series, fared much better in making a long-term impact on the field. So it makes sense that, in the 2000s, a related economic effect would trigger another remarkable burst of research into 3D pointing devices—as accessories for gaming consoles, where the hardware is typically sold below cost and subsidized by the later sale of game software. A trailblazer in this regard was the Wii Remote or Wiimote, a wand controller released for the Nintendo Wii console, which used a *3DoF* ("three degrees of freedom") *IMU* (Inertial Measurement Unit) measuring rotation and acceleration. The clever addition of an external LED emitter viewed by an onboard infrared video camera allowed for limited 3D tracking—although noisy and prone to drift, it proved extremely popular as a game peripheral.

The Wiimote was followed by a more precise wired competitor, the Razer Hydra, which offered full 6DoF electromagnetic tracking and a first-party *SDK* (Software Development Kit) for users who wanted to create their own applications. A wireless Razer Hydra sequel, called the Sixense Stem, was also released in a developer version in 2013, but then cancelled—a decision that likely drastically altered the next decade of VR tracking evolution, as many headset makers would have presumably licensed this instead of building their own.[99] The open and hackable nature of this generation of controllers made them useful beyond their intended purpose of console gaming, but their most important legacy was probably not directly related to their hardware capabilities: when Johnny Chung Lee, then a PhD student at CMU, wrote a suite of popular open-source 3D tracking software for the Wiimote, Microsoft hired him to lead development of the Kinect.[100]

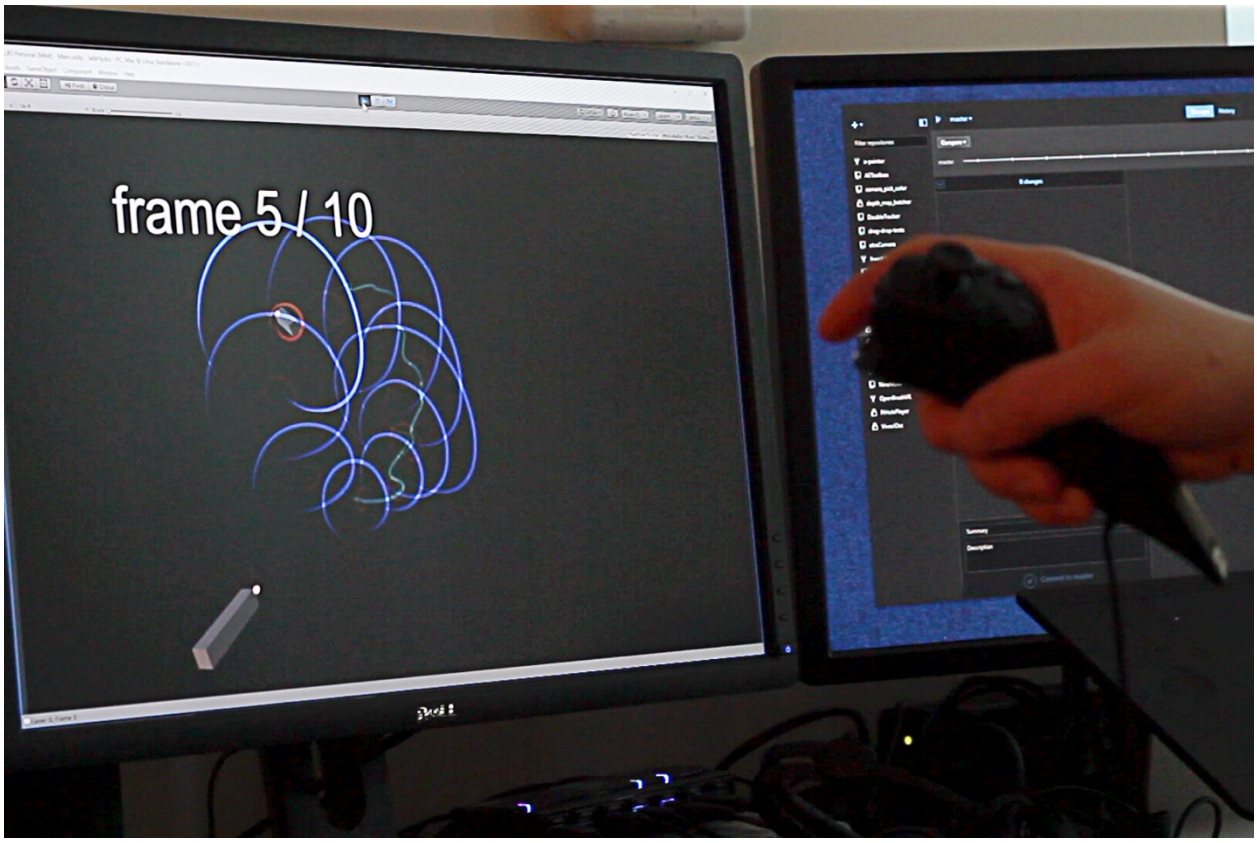

Fig. 5. Drawing with the Razer Hydra in Unity.

3.3.4. Kinect (2010)

The first Kinect remains absolutely in a class by itself when it comes to introducing 3D tracking capabilities to a wide audience—an almost preposterously significant historical impact for what was originally a fairly obscure addon peripheral for the Xbox 360.[101] Using structured light, projecting an infrared laser dot pattern over its surroundings, it returned a pair of calibrated 640x480 images with RGB colour and grayscale depth for each pixel. Continuing the trend of software reimplementing hardware that so irritated Quantel, the Kinect also used a machine learning system trained on Optitrack motion capture to find 3D skeletons in a depth map. Both of these capabilities were not new, but both of them together, in a consumer product that cost USD $150 ($200 in 2024)—heavily subsidized by console economics—were revelatory. In the world of console gaming, the Kinect was considered yet another commercial failure (in part because it used 30fps cameras, half the standard framerate for video games). But just weeks after its release, open hardware advocate Limor Fried put up a bounty of USD $3,000 ($4,400 in 2024) for writing a driver that would allow the Kinect to talk to a desktop computer. Héctor “marcan” Martin claimed the reward within a week.[102]

After the enormous positive response to the hack, and with Microsoft’s public blessing, PrimeSense, the

third-party vendor behind the Kinect's depth sensor, released their own, mostly open-source drivers for Mac, Windows, and Linux. (The skeleton-tracking components were not opened, a decision which would lead to fragmentation in the community later on.)[103] PrimeSense also licensed the Kinect technology to other companies, resulting in numerous "Kinect clones" from companies like Asus and Occipital. This initial profusion of cross-platform options abruptly stopped when PrimeSense was acquired and shut down by Apple in 2013. However, at this point a market for structured-light cameras had been established. New offerings, albeit at an unsubsidized higher cost, would continue to arrive from companies like Orbbec, Intel, and sporadically Microsoft itself. Lee left Microsoft for Google that year, where he led development on another structured-light device descended from PrimeSense hardware, the Tango tablet—a clear predecessor to the Apple Lidar, about which we will hear more shortly.[104]

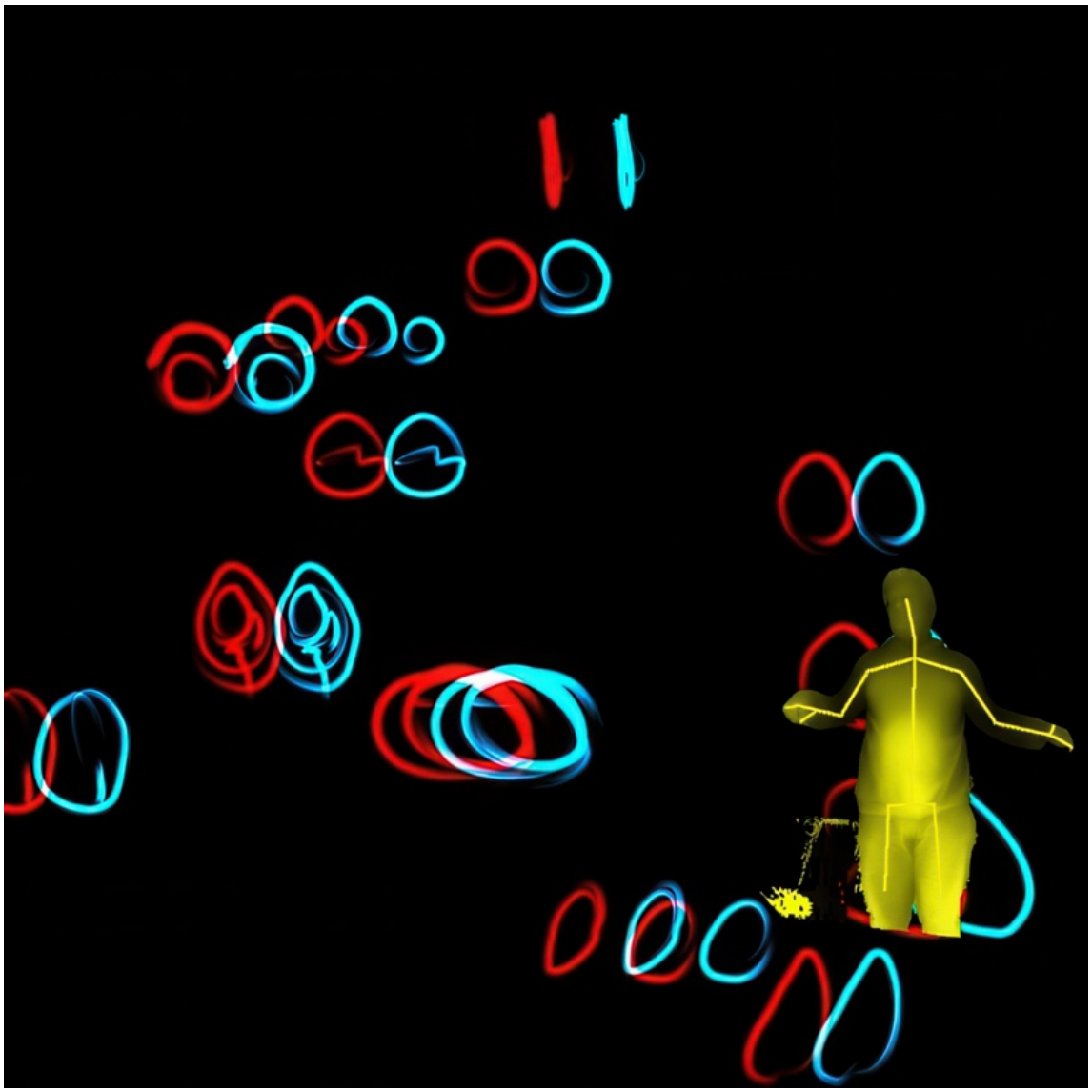

Fig. 6. Drawing with the Kinect and a presentation clicker in Unity.

Fig. 7. Asus Xtion cameras (Kinect 1 clones), driven by openFrameworks apps on Raspberry Pis.

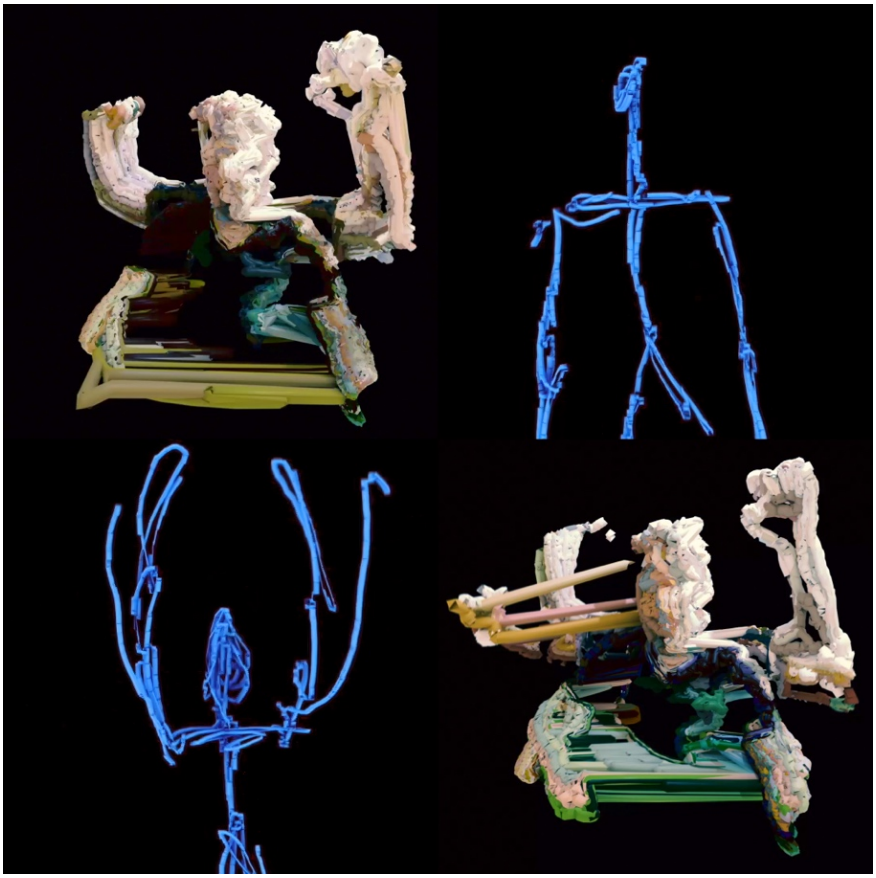

Fig. 8. Latk output from the above capture rig.

3.3.5. Leap Motion / Ultraleap stylus mode (2013)

Although primarily known as a hand tracking controller, using a pair of stereo infrared cameras and an infrared emitter, for the first several years of its existence (2013–2016) the Leap supported a stylus tracking mode. Any pen-shaped object with low reflectivity could be tracked with position and orientation, to sub-millimetre precision, albeit within a small volume.[105] As with many experimental approaches of this kind, the arrival of mass-market 6DoF controllers in 2016 drove competing approaches out of the market; subsequently the Leap rebranded as UltraLeap and returned to an exclusive focus on hand tracking.

Fig. 9. Drawing with the Leap Motion and an improvised stylus in Unity.

3.3.6. Apple Pencil (2015) and Apple Lidar (2020)

In 2015, five years after its initial release, the iPad finally acquired a first-party stylus peripheral, the Pencil. The delay is noteworthy, considering a stylus has been standard equipment for nearly every other tablet device in computing history. It was apparently only Steve Jobs' personal dislike of the tool that delayed the launch of the Apple Pencil till four years after his death.[106] Superficially similar to most other contemporary wireless stylus hardware, the Pencil's potentially interesting properties as a 3D pointing device only became apparent when higher-end iOS devices added solid-state lidar in 2020.

Solid-state or "flash" lidar uses a grid of fixed lasers that all fire at once, instead of a complex mechanism

to rapidly move a laser array over the target. For now, range remains a problem—Apple's has a range of only about 10m under ideal indoor conditions, while 200m in direct sunlight is the current standard for the mechanical lidars used in robotics. However, much as televisions were able to reach a frame rate sufficient for persistence of vision by directing beams with electromagnets instead of aiming them mechanically, rapid improvement in consumer lidar technology may result now that this threshold has been crossed. It could come about by increasing the number of lasers on the grid, splitting the beams to get more mileage out of existing lasers, or using different laser frequencies for better performance in sunlight. (Outside of industrial applications in a controlled setting, increasing laser power is probably not a safe option.)[107]

The Apple Lidar is now arguably the best-in-class implantation of XR navigation in a *magic window* style—tracking the pose of a mobile device and viewing a consistent 3D world through its screen as it's moved through space. Apple's earlier ARKit and Google's ARCore software, which attempted to achieve this effect through computer-vision tracking of RGB video, had major limitations, for example requiring the phone's camera to keep the ground plane (the floor) constantly in view. Google's earlier Tango devices, which used structured light for tracking, also solved this problem but drew too much power, with battery life measured in tens of minutes. However, currently the potential of combining the Pencil and Lidar in a novel 3D interface has not been realized—despite the 2024 launch of Apple's Vision Pro headset, which includes lidar capabilities and could use a hypothetical 3D version of the Pencil as a controller. For now, the Pencil remains a 2D device, but the hardware capabilities are there waiting to be explored.

3.3.7. Valve Lighthouse (2016), Oculus Constellation (2016), Oculus Insight (2019), and Tilt Five (2022)

The Valve Lighthouse tracking system, first distributed as developer kits in 2015 and launched publicly in 2016, represented a decisive leap forward in consumer 6DoF tracking technology. Lighthouse development was led by Alan Yates, and was originally intended for the Oculus Rift CV1, before a falling-out between Valve and Oculus' parent company Facebook led to the tracking system being licensed for the HTC Vive headset instead.[108] This system returned to the light pen concept with spectacular success in tracking quality—placing multiple inexpensive infrared light sensors all over the headset and controllers, and sweeping the tracking volume with infrared lasers from the Lighthouse base stations (one or two in the first hardware version, up to four in the second). With the known timing of the base stations' sweeps provided by a regular sync pulse, and poses for the stations obtained from an initial calibration process, any sensors illuminated by a base station laser could be precisely placed in 3D space.

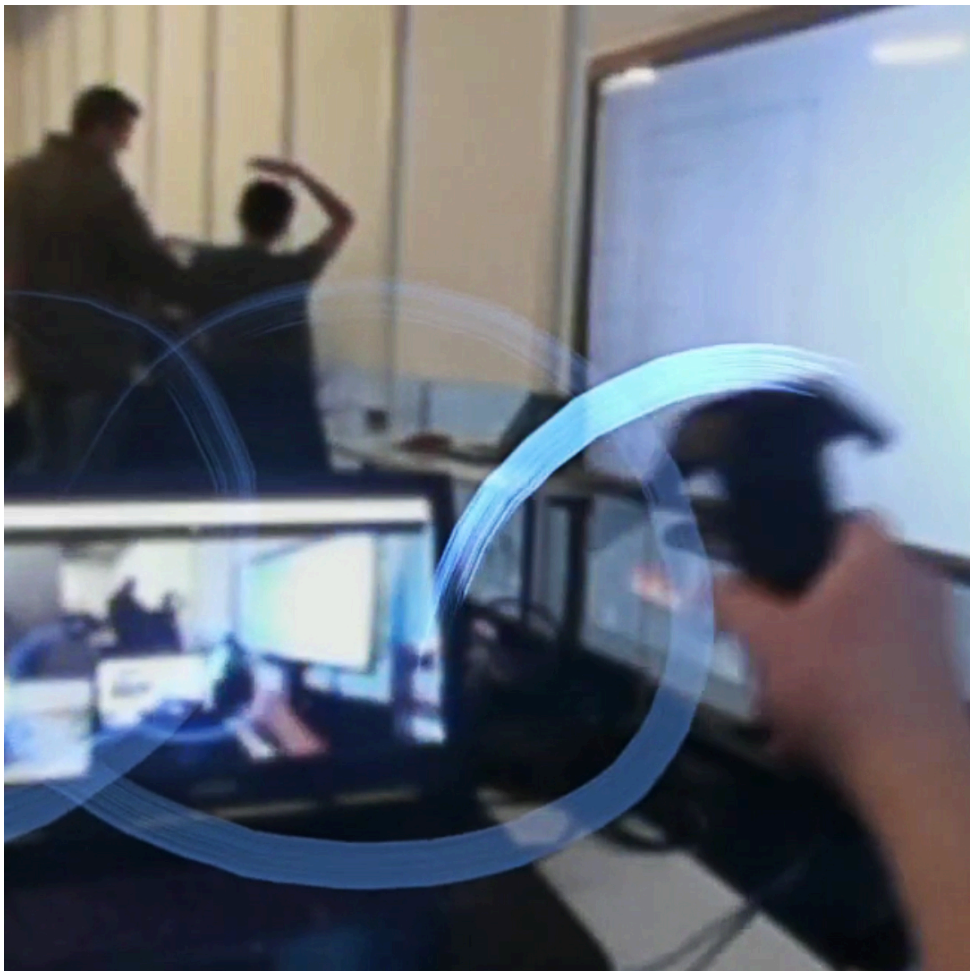

Fig. 10. A blurry early (2019) passthrough experiment using the Vive Pro in Unity.

One of the most important interface design developments of the period came from an unexpected direction—the groundbreaking volumetric drawing app Tilt Brush. Created by Patrick Hackett and Drew Skillman as an independent project in 2014 and acquired by Google in 2015, Tilt Brush revisited CAVE-era interface ideas and quickly established them as UI conventions in a new era of commodity hardware. Tilt Brush was discontinued by Google in 2021, but fortunately was open-sourced, and is currently maintained as Open Brush, continuing to add newer supported systems and features.[109]

Meanwhile, Facebook rushed to create its own replacement tracking system to complement its newly-acquired Oculus headset. Their first tracking system, Oculus Constellation, ended up using a conventional computer-vision-based approach—infrared cameras tracking LEDs on the headset and controllers—and yielded inferior results. The second iteration, named Oculus Insight, switched from an *outside-in* approach (tracking hardware separate from the headset) to *inside-out* (tracking hardware mounted on the headset), and greatly improved tracking quality. Tracking hardware aside, the desktop

versions of Oculus found a niche advantage in animation production thanks to the Quill drawing software, created by Inigo Quilez. Superficially similar to Open Brush, its interface added timeline controls for drawing in multiple frames and interoperability with VFX pipeline software like Houdini.[7] Wesley Allsbrook's short films *Dear Angelica* (2017) and (with Matthew Niederhauser, Elie Zananiri, and John Fitzgerald) *Metamorphic* (2020) are recognized early examples of volumetric hand-drawn animation created with this set of tools. Thanks to the market-leading popularity of the current versions of the Oculus—now exclusively using standalone Android hardware and rebranded as the Meta Quest—it has arguably become the most important platform supporting XR drawing at present. However, as of 2024 the maintenance status of Quill is uncertain, and it has not been ported to the new generation of standalone headsets.

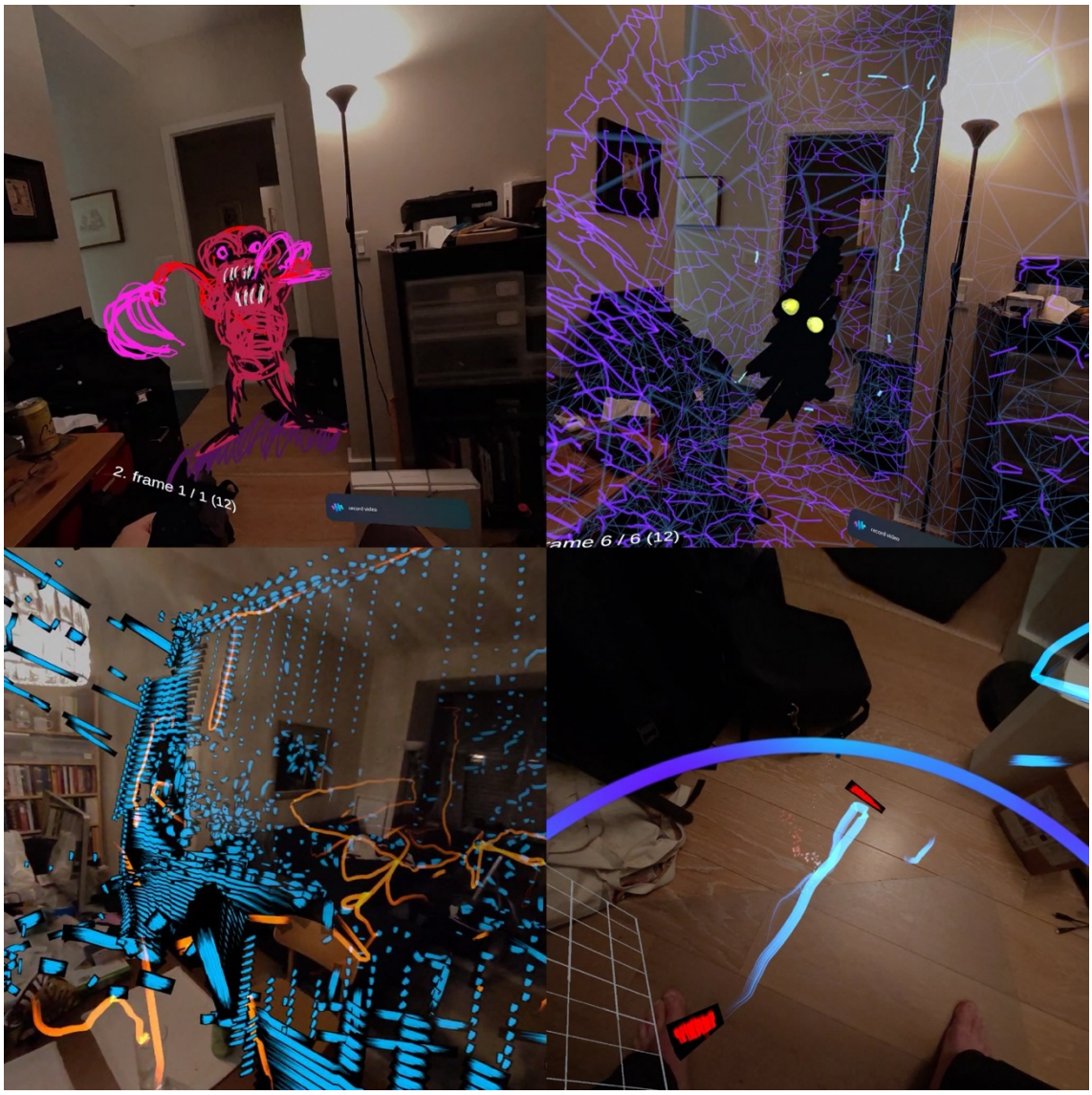

Fig. 11. Drawing with Quest controllers in Unity.

As of 2024, the long-term outcome of a third approach influenced by the Lighthouse research efforts is yet to be determined. Jeri Ellsworth, an early member of the Lighthouse team, retained independent ownership of some patents from her work at Valve. In 2013, she founded Tilt Five (formerly CastAR) to commercialize yet another novel pointing strategy—combining a retroreflective tracking mat, a desktop AR headset, and a wand-style controller; developer kits began shipping in 2022.[110]

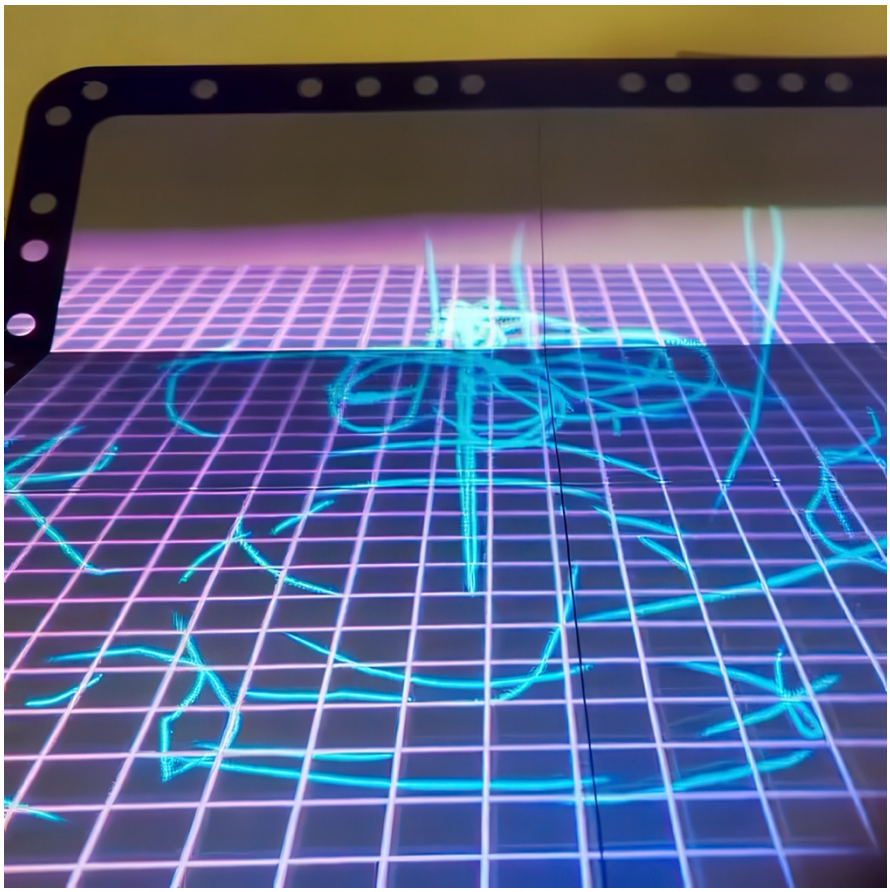

Fig. 12. Drawing with the Tilt Five controller in Unity.

3.3.8. Logitech VR Ink (2019) and MX Ink (2024)

Logitech has attempted two stylus offerings during the current XR era, both featuring a pressure-sensitive tip for drawing on physical surfaces in addition to 6DoF tracking in a pen form factor. The VR Ink stylus, already discontinued, used Lighthouse tracking for Vive and compatible desktop headsets. The newer Logitech MX Ink, launched in September 2024, works exclusively with Quest headsets, and may currently be the last consumer 6DoF stylus product standing. Interestingly, adjusted for inflation, the MX Ink's USD $130 launch price is significantly cheaper than any prior 3D drawing tool, including its own predecessor. Even adding the cost of a low-end Quest headset, this is still the first such tool priced equal to or below the adjusted cost of the original 2D Wacom ArtPad, 30 years earlier. 2D

drawing technology needed nearly a century's worth of iterations to arrive as a consumer product, but maybe 3D drawing will get there more quickly.

Fig. 13. MX Ink stylus used in Open Brush on a Quest 3.

3.4. Speculative drawing tools

> "…[T]here are many barriers to deploying virtual reality technology. Only recently has 3D graphics hardware of minimally acceptable rendering performance become affordable. Trackers and displays are fraught with technical limitations…Standardization of any form, whether hardware or software, is not yet on the horizon. Given this, most virtual reality applications have been built by or for large organizations' specific internal applications, and even these are more often than not proof-of-concept prototypes, rather than something that is in everyday use. Today's commercial virtual reality software offerings are mostly toolkits or demos."
> —Michael F. Deering (1995)[97]

There are strong signs that the 2010s burst of consumer-product interest in the 3D drawing space is not sustainable through the 2020s. While the evolution of pointing devices currently seems to be firmly intertwined with the development of stereo XR headsets, this has not always been the case. In fact, reviewing the past 150 years of stereo exhibition technology, we can see an approximately thirty-year recurring cycle of headset concepts descended from the stereoscope returning to capture the public imagination, striving for mass adoption, and failing to find it.

This is likely due to the approximately one in five people whose senses rebel at the stereopsis illusion—they experience discomfort upon receiving haptic feedback from their eye muscles converged on a real-world object at the "wrong" physical distance relative to where the illusory object seems to be (a movie screen at the far end of a theatre, or an LCD panel in a VR headset). XR technologies may offer

profoundly appealing creative tools to the remaining 80% of users, but this physical constraint may represent an insurmountable barrier to mass adoption as an exhibition medium. The challenge of devising safe, entertaining, and accessible ways to view moving 3D forms becomes especially problematic when trying to meet the demands of formats meant to be simultaneously viewable by a group audience. Even before the pandemic made the sharing of goggles or glasses undesirable, approximately 20% of viewers were thought to experience physical discomfort simply from the stereopsis illusion itself.[111]

The CAVE mix-and-match approach to display and tracking hardware may offer us alternative paths to the effective and scalable exhibition of XR projects. The most promising route at present may be the fishtank configuration—far from being an inferior substitute for projection, lenticular autostereo displays, such as the Looking Glass, are now becoming commercially available in sizes comparable to conventional monitors. These give the viewer the ability to naturally focus their eyes within an illusory volume, with correct haptic feedback, and without wearing anything on their face. Some of the most ambitious autostereo display projects, for example a Google prototype called Project Starline, demonstrated at SIGGRAPH 2024 running on a 65" 8K autostereo display, can reasonably claim to rival headsets for a subjective experience of immersion, albeit for now at a far greater cost.[112] While this is a very promising approach for users individually or in small groups, it is not likely to work as well for large-scale exhibition. A hypothetical autostereo display scaled up for a theatrical venue would run into a second and less tractable limitation: conventional stereo projection exaggerates depth, and our natural perception of depth does not work well at large distances. Perhaps future iterations of 3D pointing and display technology are waiting for complementary software design to catch up with the development of hardware—and a useful set of guiding interface principles could be derived from the following proposed requirements:

> "It cannot be overemphasized that some of the greatest opportunities in the future development of audiovisual performance systems lie in the use of more sophisticated input devices…[The ideal input] system makes possible the creation and performance of dynamic imagery and sound, simultaneously, in realtime. The system's results are inexhaustible and extremely variable, yet deeply plastic. The system's sonic and visual dimensions are commensurately malleable. The system eschews the incorporation, to the greatest extent possible, of the arbitrary conventions and idioms of established visual languages, and instead permits the performer to create or superimpose [their] own. The system's basic principles of operation are easy to deduce, while, at the same time, sophisticated expressions are possible and mastery is elusive."
> —Golan Levin (2000)[113]

Future innovative software design is also going to demand more interoperable commodity hardware simply because of the extreme volatility of the current XR hardware space. Over the time I completed this project, a partial list of Unity implementations that I deprecated due to the target devices no longer being available includes: Google Cardboard, Google Tango, Lenovo Mirage, Magic Leap 1, Microsoft HoloLens 1, Microsoft Windows Mixed Reality, and Orbbec Persee. Software stacks were in general easier to maintain, but for example the below Unreal 4 implementation will need to be completely rewritten for Unreal 5 due to an incompatible dependency. For completeness, all of these are included in Appendix A (8.7) along with my other code outputs.

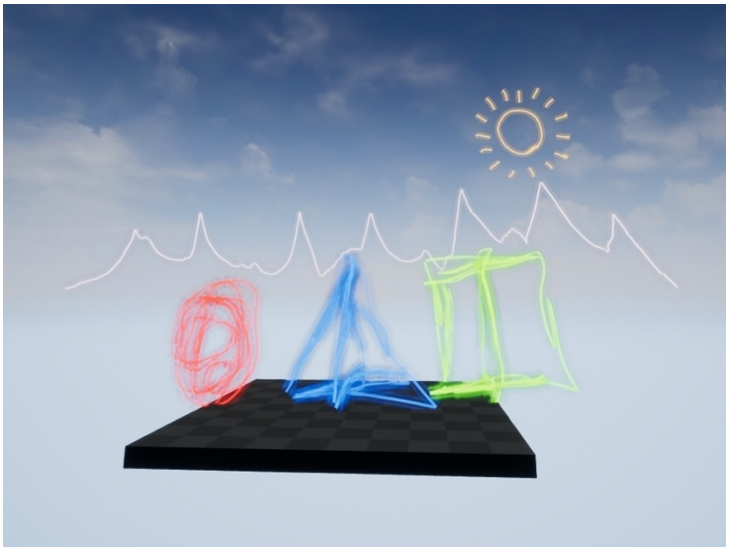

Fig. 14. An Latk test in Unreal.

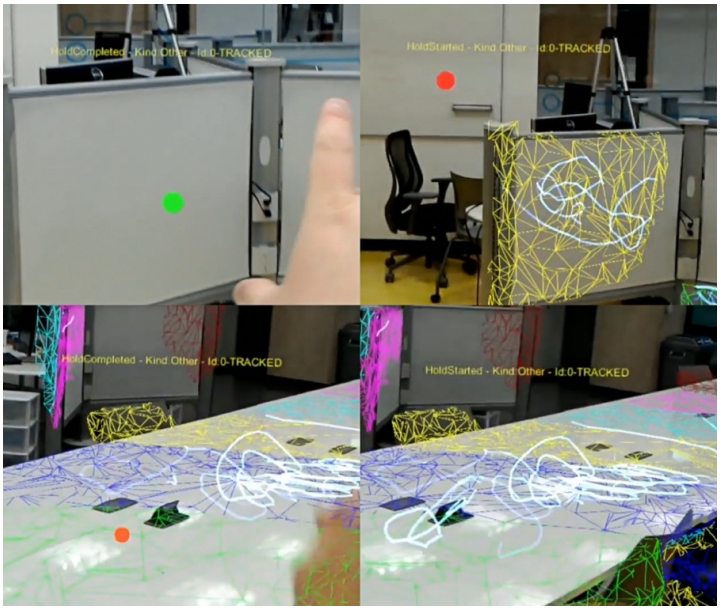

Fig. 15. Drawing on surfaces using the HoloLens.

Fig. 16. Latk running on a Google Tango tablet.

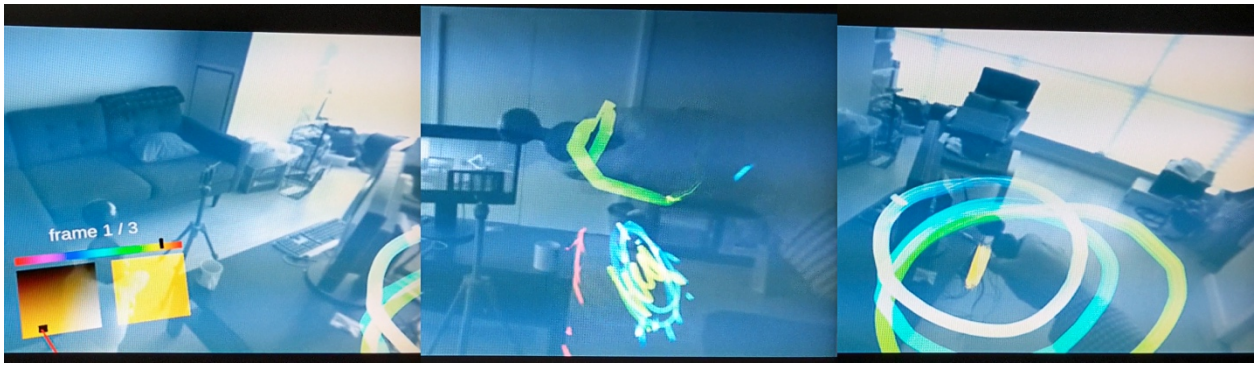

Fig. 17. Latk running on a Lenovo Mirage.

But putting aside the ongoing search for an ideal 3D drawing solution, to solve my immediate problems I arrived at the combination of:

1. A Quest headset combined with an MX Ink stylus for rough gesture work,
2. An iPad with Apple Lidar and Pencil for volumetric capture outside the studio, and
3. An ordinary Wacom tablet and monitor (using Blender's "3D cursor" drawing interface) for fine detail work.

This, then, will suffice to move us along to our next challenge: finding a large source of 3D drawing training data that will help us convert volumetric captures into a form these tools can more usefully manipulate.

4. A Hand-Drawn Volumetric Animation Pipeline

> "Humans…do not understand the world as a grid of pixels, but rather develop abstract concepts to represent what we see. From a young age, we develop the ability to communicate what we see by drawing on paper with a pencil or crayon. In this way we learn to express a sequential, vector representation of an image as a short sequence of strokes."
> —David Ha and Douglas Eck (2017)[15]

Curating a suitable dataset of 3D drawings for training ML models became the next core challenge of this project. The *Quick, Draw!* project emphasized the use of vector graphics as an available shortcut to frame-by-frame manipulation of visual information via the nature of human perception, and through its Pictionary-style gameplay generated an archive of 50 million user-contributed drawings that continue to spark further research. Any ML solution requires a sufficient collection of training data, and most prior approaches have similarly relied on 2D drawings. Fortunately, 3D drawing apps, in particular Google's Tilt Brush and its open-source successor Open Brush, have by now become popular enough with casual users to provide meaningful quantities of licensed 3D drawing data in public archives.[114]

But before I could begin building my prototypes, I first needed to better understand the challenges involved in designing a system to operate on 3D point clouds as opposed to 2D images. I surveyed current research on 3D ML systems, learned best practices for 3D asset input/output, and then began experimenting with pipelines. A primary constraint I imposed on all of these experiments was interoperability with Blender's 3D frame-by-frame drawing mode, Grease Pencil—created by Joshua Leung in 2008, and expanded by the Blender community, in particular by Antonio Vázquez and Clément Foucault, to arrive at close to its present feature set by 2019. The purpose of Grease Pencil mode is to use highly performant brush and fill representations for 3D objects—with default material properties presently limited to vertex colours, a single texture, and shading from a single light source—in order to allow playback and editing of very large frame-by-frame 3D scenes on modest hardware.[115]

4.1. The Latk file format

An immediate problem one encounters when trying to apply 3D drawing-based research to practical production is the file format you are going to use—real-world animation pipelines require moving data between applications, and most of this Grease Pencil information is not trivially portable outside of a Blender project file. In fact, until fairly recently, 3D animation did not have an open standard format capable of storing frame-by-frame data at all. In 2011, the Alembic format arrived to fill this role, but as of 2024 implementations are still inconsistent, even among major 3D creation apps. I wanted a simple, universal exchange format that I could use to move data between any two platforms, including to and

from browsers and mobile devices, and so I created the Latk file format: a very large JSON file, zipped to save disk space, saved with the .latk extension. For choosing what to include in the specification, I used the current feature set of Blender Grease Pencil as my guide—so as Grease Pencil has added features, I have added fields to the JSON spec, while maintaining backward compatibility with older implementations.

The basic Grease Pencil representation of 3D frame-by-frame animation consists of:

1. A top-level Grease Pencil object containing a number of Layers, and an optional transform property (meaning values for position, rotation, and scale)
2. Layers, each of which contain a number of Frames, and an optional transform property.
3. Frames, each of which contain a number of Strokes.
4. Strokes, each of which contain a number of Points, and properties that affect all Points (stroke colour, fill colour, etc.).
5. Points, each of which contain individual properties (position, vertex colour, etc.).

Represented in JSON, this is:

```
{
  "creator": "latk.py",
  "version": 2.9,
  "grease_pencil": [
    {
      "layers": [
        {
          "name": "GP_Layer",
          "frames": [
            {
              "strokes": [
                {
                  "color": [ 0.20259166, 0.032980658, 0.9169371, 1.0 ],
                  "fill_color": [ 0.0, 0.0, 0.0, 1.0 ],
                  "brush_name": "optional",
                  "brush_creator": "optional",
                  "points": [
                    {
                      "co": [ 1.1935601, 0.98816276, -0.74828625 ],
                      "pressure": 0.50230646,
                      "strength": 0.50914043,
                      "vertex_color": [ 0.0, 0.0, 0.0, 1.0 ]
                    }
                  ]
                }
```

```
                ]
              }
            ]
          }
        ]
      }
    ]
}
```

To make the following prototypes, I then created Latk implementations in Python, JavaScript, C# (Unity), Java (Processing), and C++ (openFrameworks).

4.2. Lightning Artist Toolkit 001 (latk_ml_001)

My first attempt at manipulating data in 3D involved voxelizing an Latk frame in a 256^3 volume using Processing, then running artistic-videos, a Lua Torch style transfer system by Manuel Ruder, on 256 image slices at 256 x 256 each, resulting in an unstructured colour point cloud.[116] [117] This approach yielded visually interesting results, but I did not yet have a suitable method for converting point clouds into brushstroke-equivalent lines. It was also highly unpredictable and unscalable.

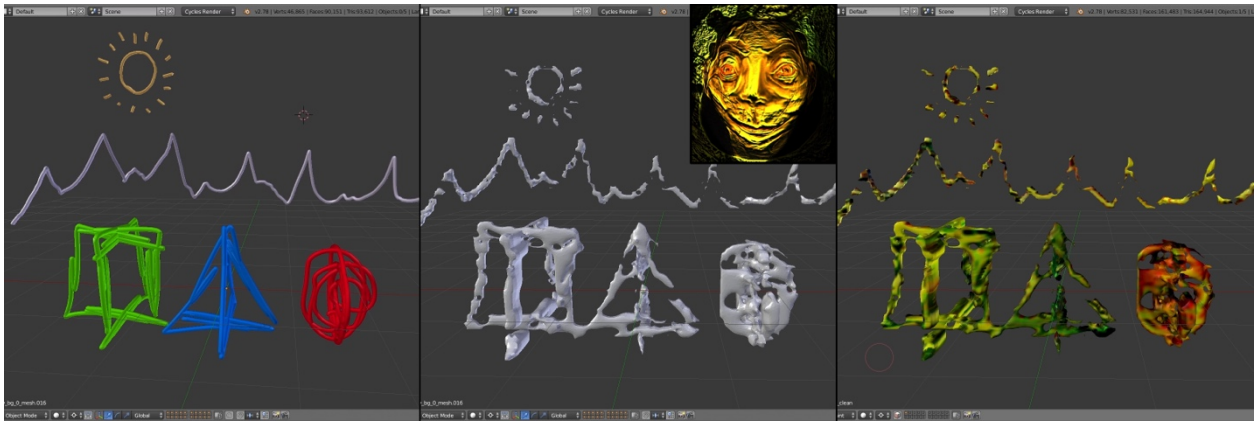

Fig. 18. Style transfer using latk_ml_001.

4.3. Lightning Artist Toolkit 002 (latk_ml_002)

My first successful approach used a 1024x1024 Pix2Pix implementation in TensorFlow by Karol Majek, trained on a dataset of depth map and contour pairs generated by Difan Liu and Aaron Hertzmann's Neural Contours system.[118] [119] I then extracted vector centre lines from the contour images using Martin Weber's AutoTrace, and converted these 2.5D line sets to Latk format for input into Blender.[120]

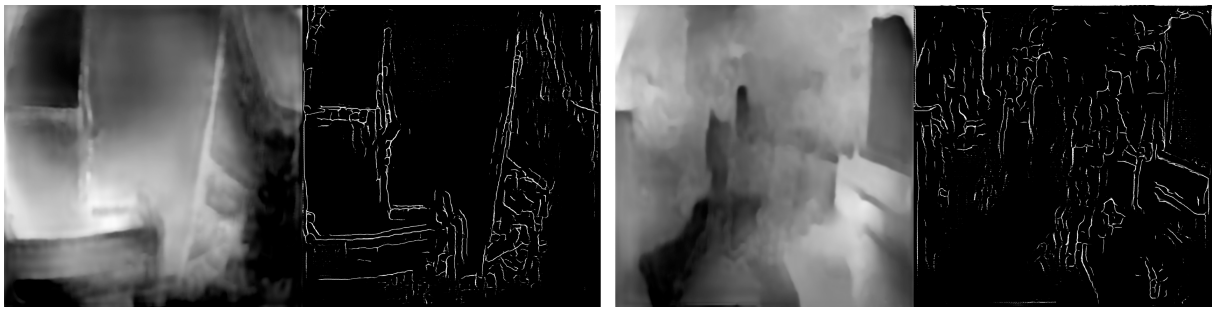

Fig. 19. Initial contour detection test in latk_ml_002.

A variation of this approach used pairs of depth maps and normalized XYZ coordinates reconstructed from camera intrinsics, but this proved highly temporally unstable.

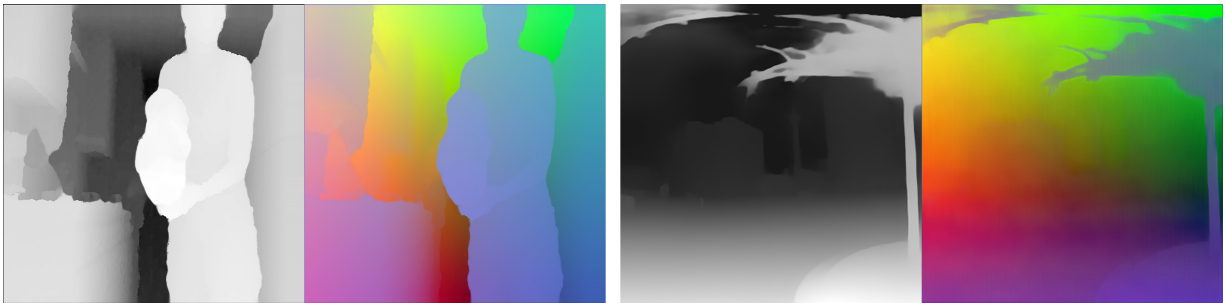

Fig. 20. RGBXYZ test in latk_ml_002.

The most practically useful variation from this iteration used depth maps and contour pairs derived from 2,465 real 3D drawings, Creative Commons-licensed Open Brush sketches downloaded from Google Poly. It quickly became apparent that the Google Poly archive would be the most promising source of training data for future attempts.

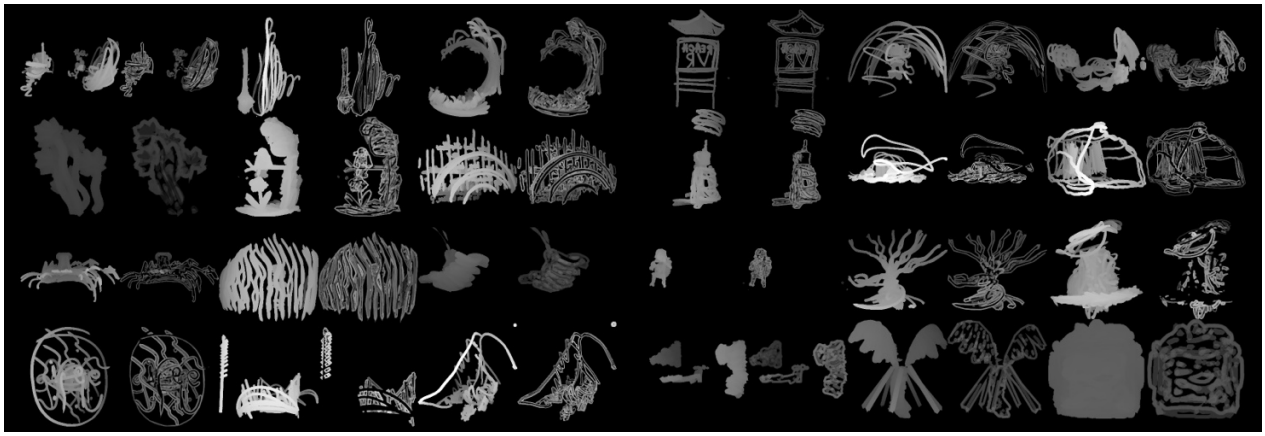

Fig. 21. Alternative depth map to contour tests in latk_ml_002.

4.4. Lightning Artist Toolkit 003 (latk_ml_003)

In 2021, Google Poly shut down, and the Internet Archive preserved its Creative Commons-licensed archive of Open Brush drawings. I created a set of Python scripts to organize the unstructured data dump and check the metadata for CC-BY license information and attribution, creating a final set of 56,723 Open Brush artworks with individual credits for 13,908 artists (provided in Appendix B), which I call TiltSet—as far as I am aware, the largest collection of verifiably licensed 3D drawing data ever published. To facilitate training and postprocessing, I converted the binary Open Brush files to various other 3D formats—Latk, and also GLTF, binvox, and HDF5.

The layout of the final dataset, accessible at *FRDR* (Federated Research Data Repository), is as follows:

Original format, generation 1:

```
./tilt.tar.gz
  Tilt Brush format (tilt)
  Original Open Brush binary files, from the Google Poly archive.
```

Derived formats, generation 2:

```
./geometry_json.tar.gz
  Open Brush JSON format (json)
  Converted from tilt to Open Brush JSON format, with Open Brush.

./glb_draco.tar.gz
  GLTF Draco format (glb)
  Converted from tilt to GLTF, with Open Brush.
  Converted from GLTF to GLTF Draco, with gtlf-pipeline.

./latk.tar.gz
  Lightning Artist Toolkit format (latk).
  Converted from tilt to latk, with the latk Python module.
```

Derived formats, generation 3:

```
./binvox.tar.gz
  Binvox voxel format (binvox), 64^3, 128^3, and 256^3 versions.
  Converted from latk to binvox, with the binvox Python module.
```

Derived formats, generation 4:

```
./hdf5.tar.gz
  HDF5 voxel format (im, seg), 64^3, 128^3, and 256^3 versions.
  Converted from binvox to hdf5, with the h5py Python module.
```

Supplementary information:

```
./license
```

```
  License text, README, and documentation.

./metadata.tar.gz
  Original json metadata per file, from Google Poly archive.

./textures.tar.gz
  Open Brush texture files for use with GLTFs.

./thumbnail.tar.gz
  Original thumbnail previews per file, from Google Poly archive.
```

The HDF5 versions of this dataset were then used to train full 3D brushstroke segmentation models using Vox2Vox, by Marco Domenico Cirillo, on an Nvidia A100 GPU supplied by *DRAC* (Digital Research Alliance of Canada).[121] To create A and B samples for training, I started from 256^3 voxelized originals, and then applied a binary dilation operation to approximate a larger point cloud object. I trained three final Vox2Vox models—at 64^3 resolution, 200 epochs, and batch size 32; at 128^3 resolution, 200 epochs, and batch size 16; and at 256^3 resolution, 200 epochs, and batch size 2. Shell scripts for reproducing these results are included in Appendix A (8.7.2).

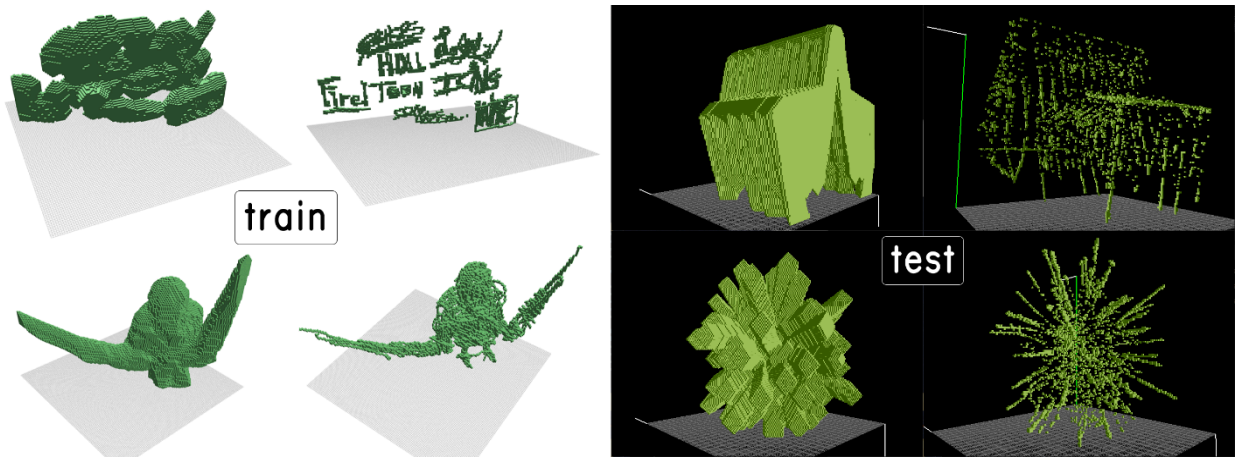

Fig. 22. Training and testing examples using latk_ml_003.

Fig. 23. The 256^3 model, from left, at training epoch 50 and at epoch 200.

I also created a secondary dataset for testing purposes: ABC-Draco is a collection of 751,407 dense point clouds with normals stored in the GLTF Draco format, converted from a subset of the NYU ABC-Dataset of 3D models.[122] By using lossy Draco compression, the total file size is under 50GB—two orders of magnitude smaller than the original, which makes this version useful in more resource-constrained environments. Both datasets are linked in Appendix A (8.3).

Having found suitable sets of training and testing data, I now needed a suitable source of live-action volumetric capture data. Mature volcap solutions are generally expensive subscription products, and without a clear and consistent market demand for high-end volcap facilities, even larger companies' offerings in this area have proven volatile. (During the life of this research project, every commercial multi-camera volcap product that I evaluated has since been either acquired, shut down, or both.)

Fortunately through the efforts of a group of local educators, including Andrew Hogue at the University of Ontario Institute of Technology and Cindy Poremba at *OCAD* (Ontario College of Art and Design University), we were able to benefit from the work of Chris Remde at the Experimental Surgery Berlin group.[123] Remde is the current maintainer of the Kinect-based open-source volcap software LiveScan3D, originally created in 2015 by Marek Kowalski, Jacek Naruniec, and Michal Daniluk.[124] LiveScan3D capture rigs have no ongoing operating costs other than the care and feeding of an array of Kinect

cameras, corresponding USB PCI cards, and a high-spec desktop computer—allowing us to set up long-term installations at multiple locations in and near Toronto, including the Alice Lab here at York, and record all the testing data for the following iterations of Latk.[4]

For the resulting examples, I translated the raw voxel output from the ML system into a network of sparse brushstrokes using various procedural methods, including a simple k-means sort, Difference Eigenvalues,[125] Growing Neural Gas,[126] and SynDraw[127] Once this system generated the final brushstroke information, it was saved to my Latk format, now ready to read into industry-standard 3D creation software such as Blender, Maya, and Houdini; popular creative coding libraries like Processing, p5.js, three.js, and openFrameworks; and game engines like Unity and Unreal.

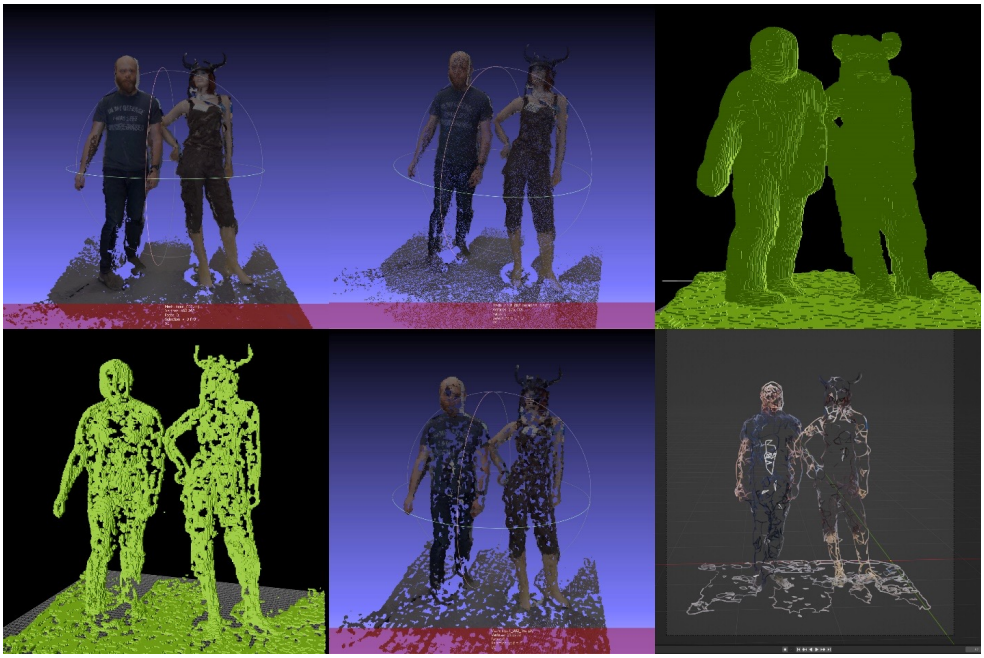

Fig. 24. Output of each step in the latk_ml_003 pipeline, from top left: original point cloud, downsampled point cloud, dilated voxel volume, inference voxel volume, restoration of original colour values, and brushstroke creation.

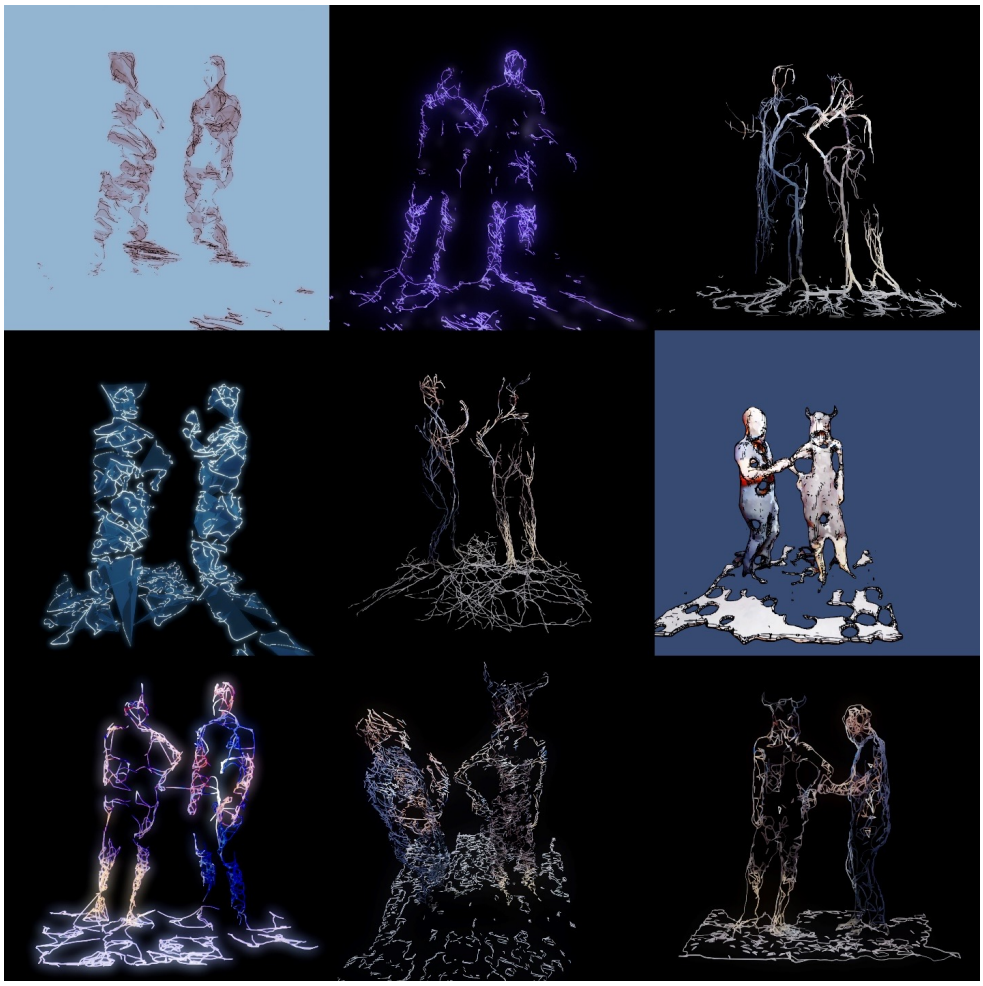

Fig. 25. Nine different examples of latk_ml_003 final output, using the same volumetric capture sequence (provided by Andrew Hogue), from top left: Houdini, Processing, Blender, Processing, Blender, Houdini, Processing, Blender, Blender.

4.5. Lightning Artist Toolkit 004 (latk_ml_004)

The pipeline's fourth iteration returned to the single-viewpoint 2.5D approach of latk_ml_002, for location shooting scenarios without access to a volcap array. RGB and depth maps are processed with the informative-drawings system by Caroline Chan,[58] then vectorized using an implementation of the Zhang-Suen thinning algorithm by Lingdong Huang.[128] While not generating fully 3D output like latk_ml_003, this version comes closest to meeting the test case for this research: the generation of a collection of brushstrokes from a point cloud that approximates what an artist might draw from scratch in XR.[129]

Fig. 26. The stages of the latk_ml_004 pipeline, from top left: RGB input, depth map input, ML system output, and final projection from camera.

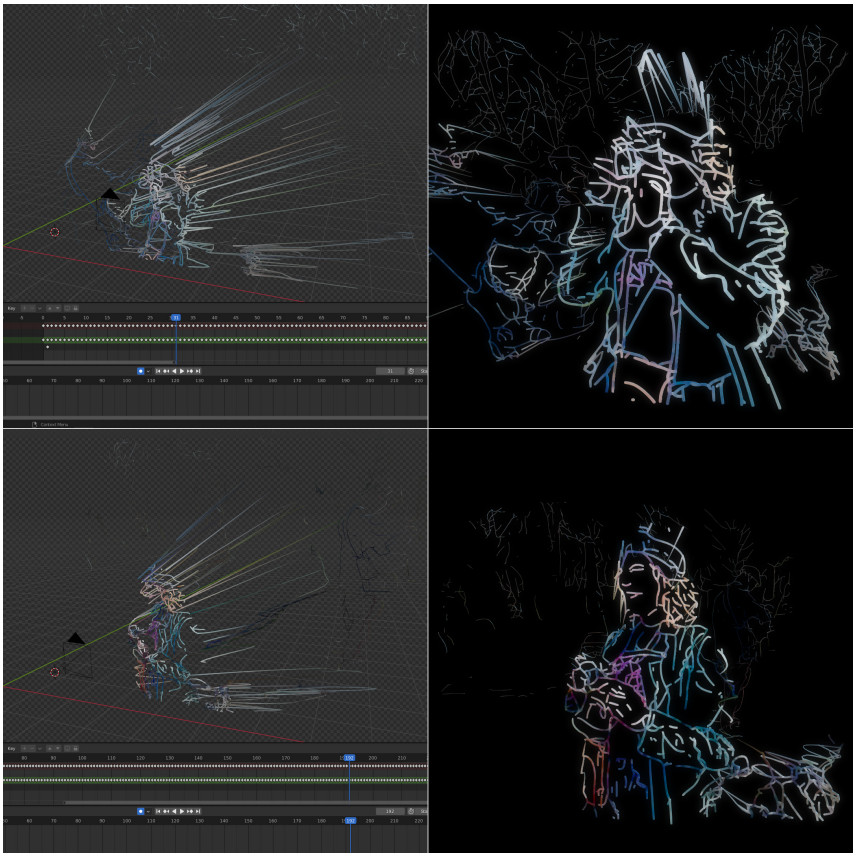

Fig. 27. Two frames of final output from latk_ml_004.

4.6. Lightning Artist Toolkit 005 (latk_ml_005)

The fifth and final iteration of the pipeline combines latk_ml_002, latk_ml_003, and latk_ml_004, implementing both PyTorch and *ONNX* (Open Neural Network Exchange) backends for each, in preparation for wrapping them all together in a cross-platform Blender Python addon. ONNX is of special interest here because it is the first widely accepted standard for cross-platform inference;[130] models are first trained in another framework, usually PyTorch, then converted. ONNX models can be used in Unity as well, via Unity's Sentis (formerly Barracuda) system, and this will have some interesting implications as we now move on to production applications.

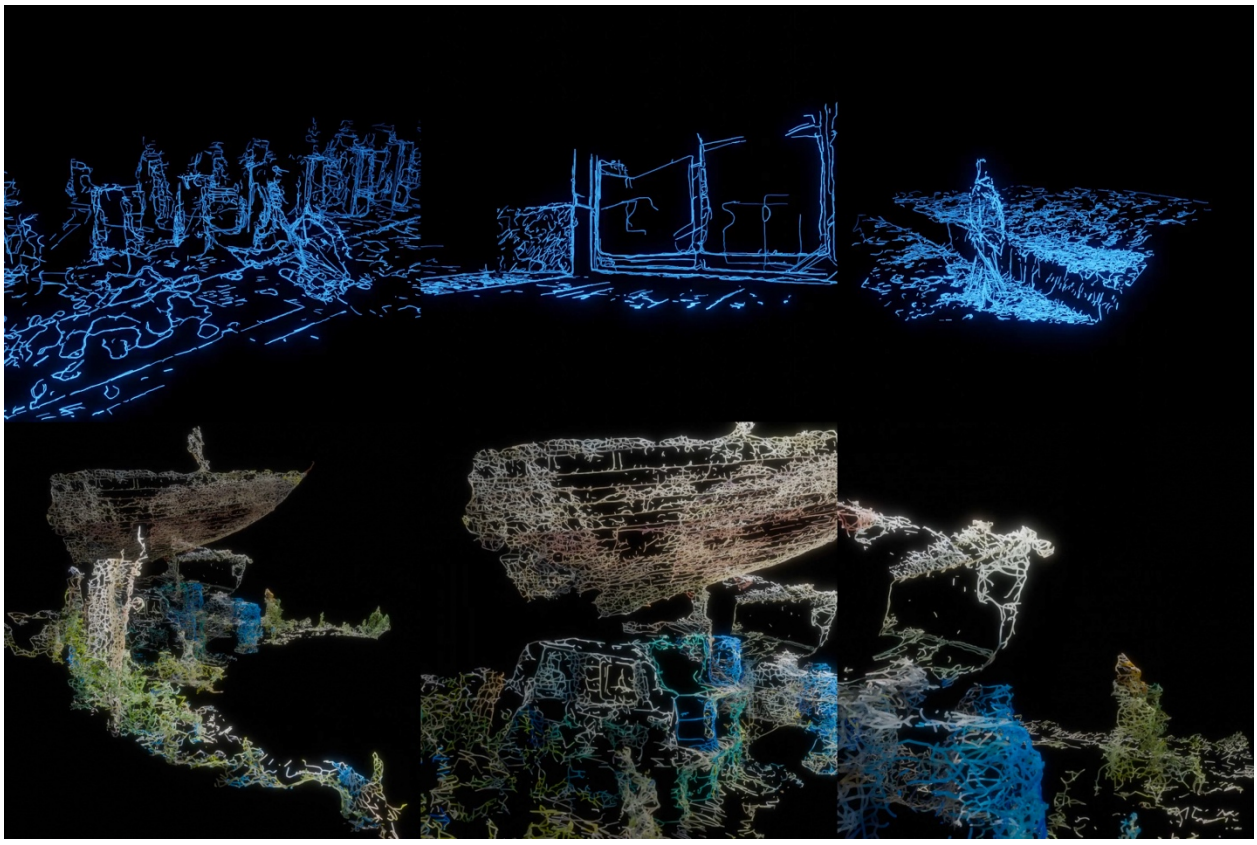

Fig. 28. Stills from two outdoor lidar scans in Grease Pencil.

5. The Pipeline in Production

> "…CGI is really just…playing with computerized puppetry. You can stretch and squash somewhat, but distorting as desired isn't always possible. So they stretch and squash and do very fast motions and hide it all under a veneer of out-of-focus. It ends up looking like just fast motion and not a true distortion…"
> —Michael Sporn (2013)[131]

> "One evening, after thinking it over for some time, Harold decided to go for a walk in the moonlight. There wasn't any moon, and Harold needed a moon for a walk in the moonlight. And he needed something to walk on. He made a long straight path so he wouldn't get lost. And he set off on his walk, taking his big purple crayon with him. But he didn't seem to be getting anywhere on the long straight path. So he left the path for a short cut across the field. And the moon went with him."
> —Crockett Johnson (1955)[132]

5.1. Unity XR headset app (latkUnity_OpenXR)

The primary source of the original Latk material documented here is a collection of Unity XR apps running on a wide range of platforms and devices. From 2020–2022, the Vive implementation was my main development focus; by 2023, it was superseded by the Quest implementation. With a standalone form factor, MX Ink stylus support, depth scanner, and colour passthrough, the Quest currently remains the most robust overall platform for Latk creation.

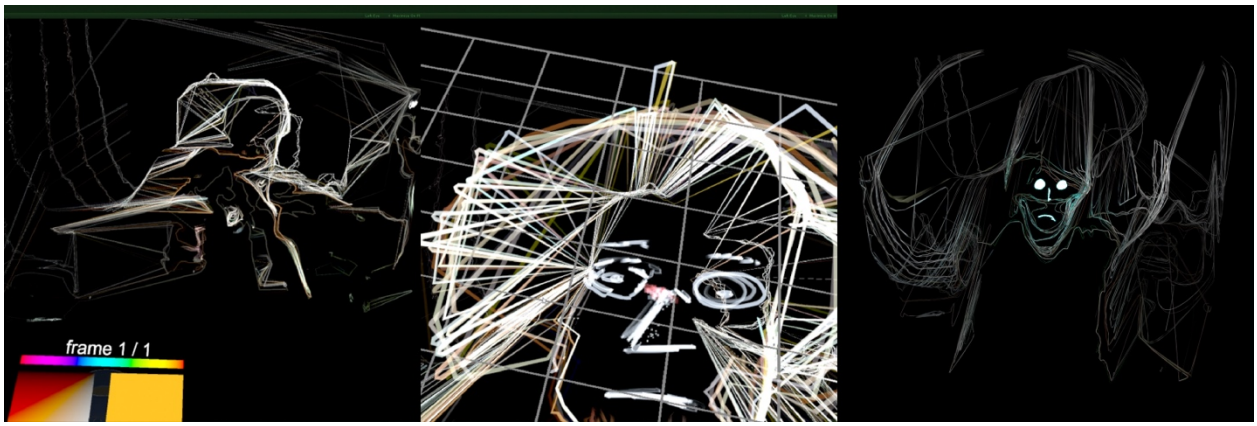

Fig. 29. Drawing in a Vive headset, demonstrating the colour palette and collision grid features.

Fig. 30. Drawing in a Quest 3 headset, also demonstrating passthrough and onion skinning (centre).

As I began developing working habits with these tools, a pattern developed in which I would first create rough sketches using environmental volumetric captures and 6DoF controllers in XR, then bring them via the Latk exchange format into Blender Grease Pencil for fine detail work and integration into a larger animated scene. I originally anticipated that "round-tripping" from Blender back to the XR app would be a common occurrence in this pipeline, but—while this is technically possible by exporting Latk files from Blender and importing them back into the XR app—I found this was rarely of great practical use. The "3D cursor" interface that Blender Grease Pencil uses for 3D stroke drawing in a 2D interface offers precise manipulation with a Wacom tablet once its conventions are mastered, making it the logical end point for cleanup and fine detail work.

The typical control scheme is as follows:

```
Controller 1:
  Trigger 1 = Draw
  Pad 1 centre = Material type
  Pad 1 left = Brush size smaller
  Pad 1 right = Brush size larger
  Pad 1 up = Toggle collisions
  Menu 1 = Show colour palette + Sample colour + Push vertices
  Grip 1 = Move

Controller 2:
  Trigger 2 = New frame
  Pad 2 centre = Toggle play
  Pad 2 left = Frame forward
  Pad 2 right = Frame back
  Pad 2 up = Go to first frame
  Menu 2 = Toggle onion skin
  Grip 2 = Move
```

```
Controller 1 + 2:
  Menu 1 + Trigger 1 = Erase stroke
  Menu 1 + Trigger 2 = Duplicate frame
  Menu 1 + Menu 2 = Delete frame
  Menu 1 + Pad 2 up = New layer
  Menu 1 + Pad 2 left = Next layer
  Menu 1 + Pad 2 right = Previous layer
  Pad 1 down + Pad 2 down = Save Latk file

Controller 1 + 2 (Extra button on Quest only):
  Extra1 + Extra2 = Do ONNX inference
```

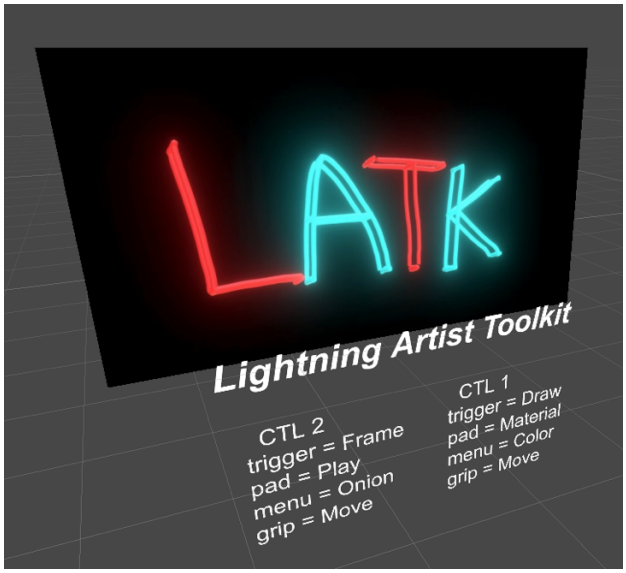

Fig. 31. The Latk Unity app splash screen.

All of these Unity examples use world scale navigation—a name originally chosen by the developers of Tilt Brush for the XR navigation convention that I consider ideally suited to drawing, sculpting, and other precision tool use. It is unique in allowing the user to simultaneously control, in a single gesture, position, rotation, and scale—the entire transform of the world origin, with seven degrees of freedom assuming scale is applied across all axes equally. Most implementations of world scale use two 6DoF tracked controllers; this would theoretically also be possible to do with hand tracking, if a method exists with enough of a range of motion to be practical.

Once world scale is activated—usually by engaging both controllers' grip buttons (a standard feature of controller button layouts starting with the Vive)—the user can navigate using the following methods:

1. Translation by moving both controllers while preserving their relative distance from each other,
2. Rotation by changing the angle between controllers, and
3. Scaling by moving the controllers closer or further apart.

Some implementations add a fourth method, activated with the grip button on a single controller:

4. Simultaneous translation and rotation, relative to corresponding changes in controller position and orientation. This is less precise than the two-controller modes, but useful for incremental adjustment while using a tool.

5.2. Unity XR tablet app (latkUnity_ARFoundation)

Among the wide range of non-headset devices that I experimented with designing apps for, a standout combination was an iPad with lidar and Apple Pencil. While the Quest 3 does have a depth scanner, it's not directly accessible in realtime—you have to record a static scan with a first-party app, then read the resulting mesh data in your own app.

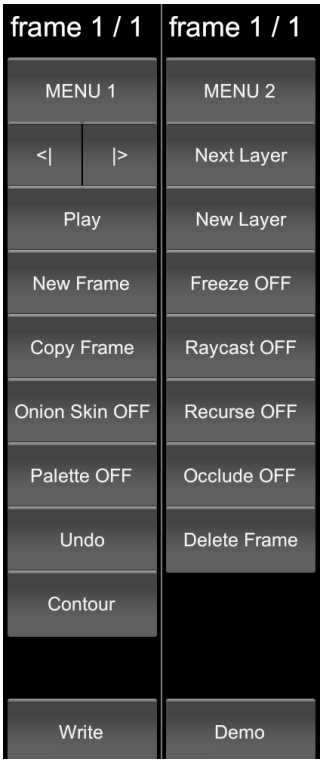

Fig. 32. iOS button menus, with lidar features.

[Menu 1: Forward, Back, Play, New Frame, Copy Frame, Onion Skin, Palette, Undo, Contour, Write]

[Menu 2: Next Layer, New Layer, Freeze, Raycast, Recurse, Occlude, Delete Frame, Demo]

Meanwhile third-party apps on the iPad can straightforwardly access realtime lidar and rgb camera data, making the device an ideal testing ground for ONNX ML models in the field. (Ironically, as of 2024 this is not as easy to do on Apple's own XR headset, the Vision Pro.) Moving the 2D pencil while relying on the tablet's 6DoF tracking for depth is less intuitive than a 6DoF controller, much less a true 6DoF stylus, but the ease of importing scan data and immediately editing it compensates for this shortcoming.

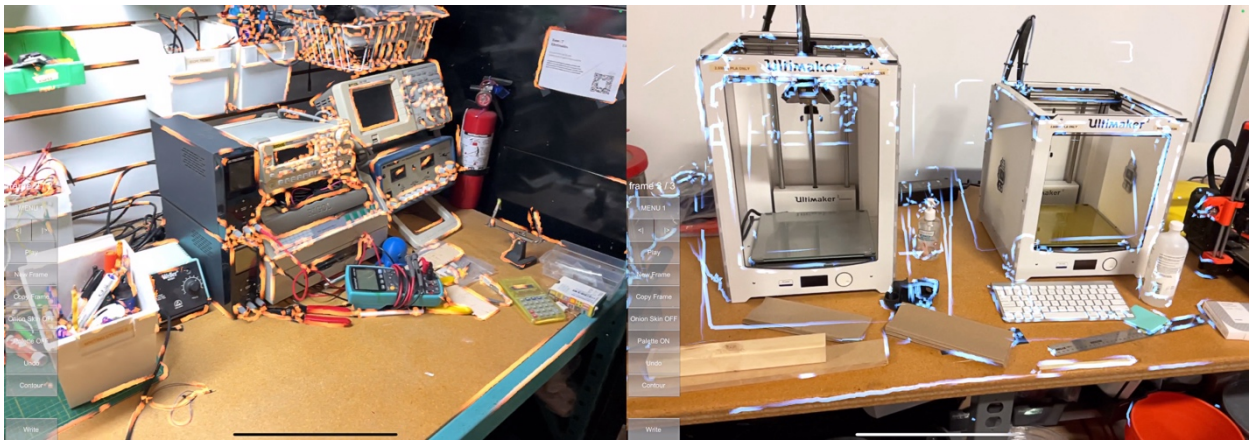

Fig. 33. Generating brushstrokes from depth data with the iPad lidar and Unity Sentis.

Fig. 34. Drawing on the walls with the depth capabilities of the Quest 3 (top) and iPad Pro (bottom)

5.3. Blender addon (latk_blender)

Finally, the most complex product of this research is the Blender addon—a Python GUI wrapper that controls the ML models from the latk_ml_002, latk_ml_003, and latk_ml_004 systems, as well as automating more mundane pipeline functions. In particular, it can generate meshes from Grease Pencil strokes so they can be used with Blender render engines other than Eevee, such as the Cycles raytracer or the third party diffusion engine Dream Textures (which we will explore further later). The addon also serves as scaffolding that can support the future use of other ML models within Blender.

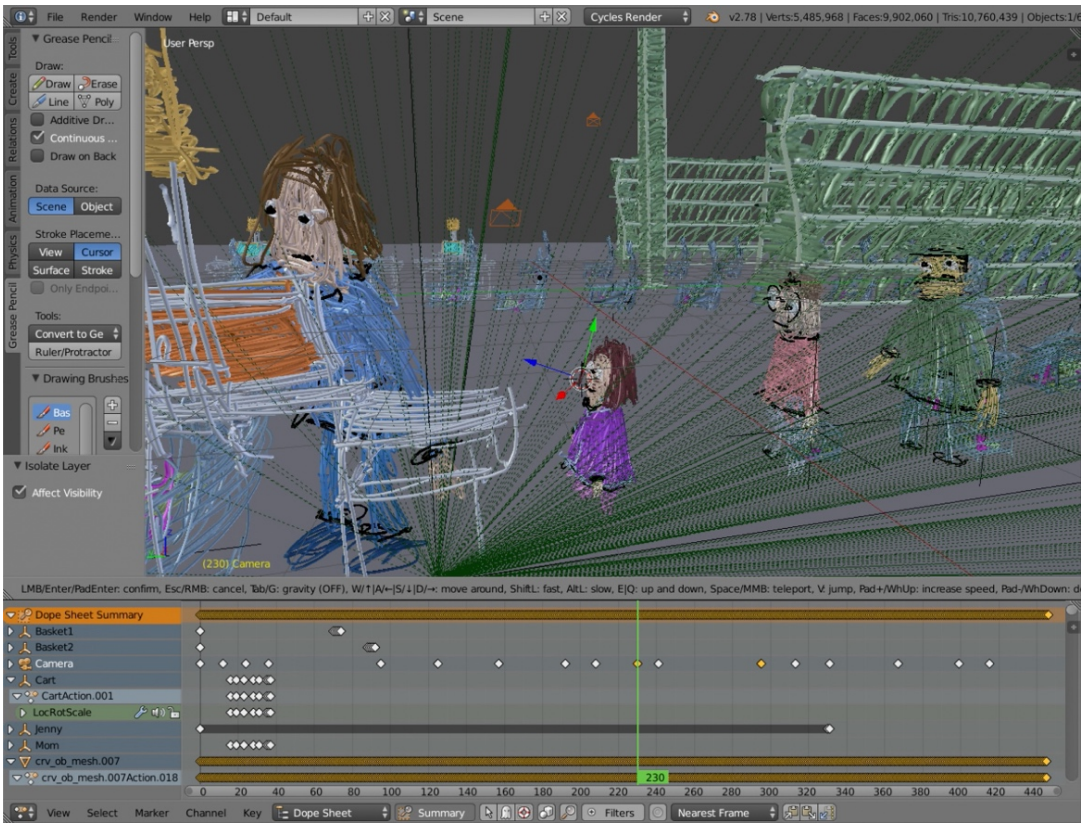

Fig. 35. A Blender scene from *Jenny in the Self-Checkout Line* (2017), with tube meshes generated from Grease Pencil strokes ready for rendering in Cycles.

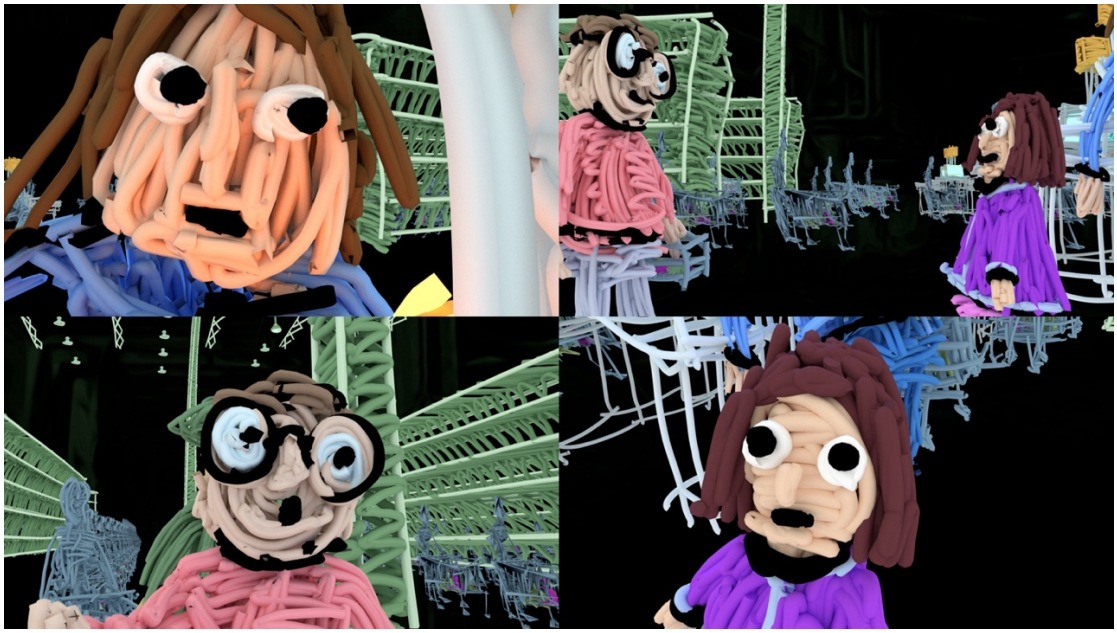

Fig. 36. Final Blender Cycles renders of the above scene.

5.3.1. Blender Setup UI Panel

The Setup UI Panel includes the option to toggle parts of the main interface on and off, and to switch between a Pytorch or ONNX ML backend. MPS support (a form of hardware acceleration for ARM Macs), can be enabled here as well. Next come buttons that run install scripts for the various components, and finally checkboxes for a wide range of extra import and export formats.

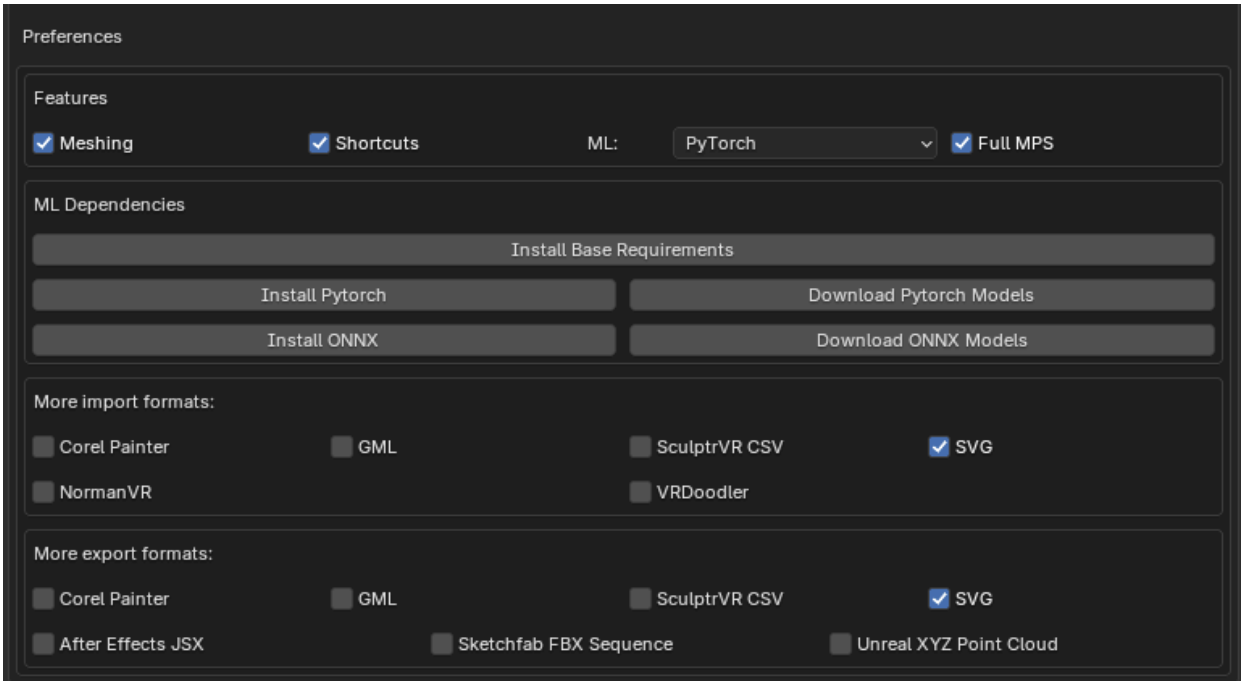

Fig. 37. Blender Setup UI Panel.

[Install Base Requirements, Install PyTorch, Install ONNX, Download PyTorch Models, Download ONNX Models, More import formats (Corel Painter, GML, Norman VR, SculptVR CSV, SVG, VRDoodler), More export formats (After Effects JSX, Corel Painter, GML, SculptrVR CSV, Sketchfab FBX Sequence, SVG, Unreal XYZ Point Cloud)]

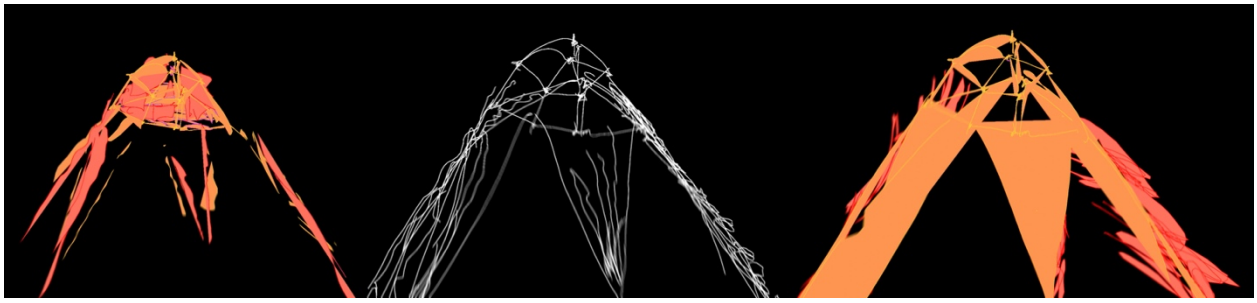

Fig. 38. Alternative export formats—three frames from an After Effects vector shape export.

5.3.2. Blender Meshing UI Panel

The Meshing UI Panel converts Grease Pencil strokes into various mesh geometries relevant to brushstroke work, including ribbons, tubes, and hulls.[133] It also has a number of other useful features, like adding UV coordinates for placing textures on strokes, creating a colour palette of common

materials, converting materials to vertex colours, and remeshing strokes to remove unwanted intersections. The meshing methods are currently implemented entirely in Python, and processing a large number of frames becomes impractically slow to run in an interactive Blender session. Speed can be considerably improved by running in a headless session, but this can be inconvenient as well as discouraging to novice users. A future improvement would move meshing functions to native Geometry Nodes called from Python instead.

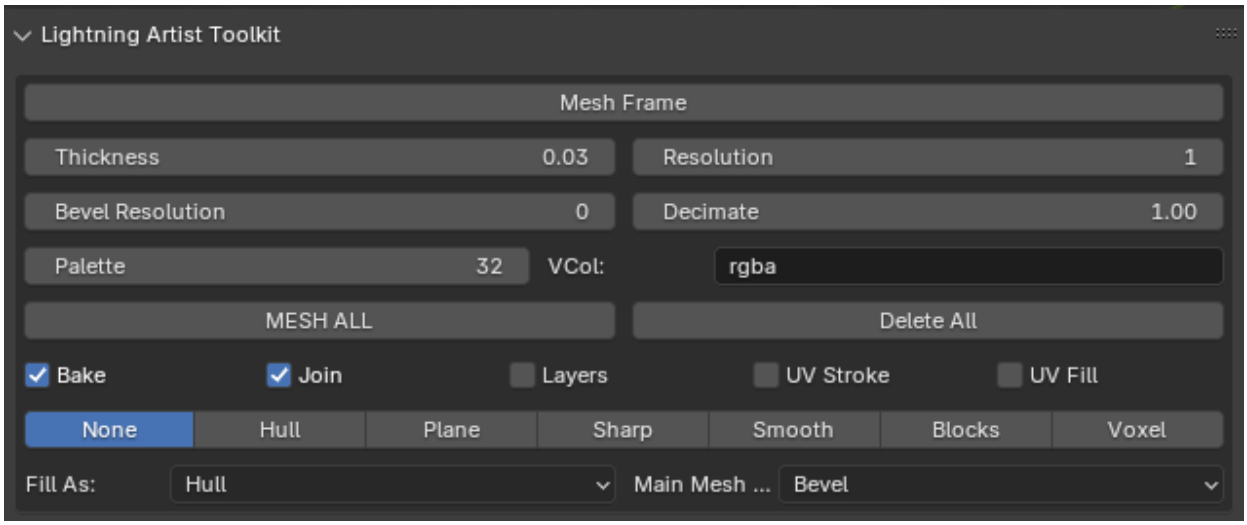

Fig. 39. Blender Meshing UI Panel.

[Mesh frame, Thickness, Resolution, Bevel resolution, Decimate, Palette colour count, Vertex colour attribute name, Mesh all, Delete all, Bake on/off, Join on/off, Layers on/off, UV Stroke on/off, UV Fill on/off, Remesh mode (None, Hull, Plane, Sharp, Smooth, Blocks, Voxel), Fill as (Hull, Plane), Main mesh mode (Bevel, Extrude, Solidify)]

Fig. 40. From left, original Grease Pencil stroke, tube mesh, block mesh, and voxel mesh results.

5.3.3. Blender Shortcut UI Panel

The Shortcut UI Panel automates various geometry and animation pipeline operations, and is primarily intended as a development tool, disabled by default. Most of its functionality is superseded by the Geometry Node features available in more recent versions of Blender.

Fig. 41. Blender Shortcut UI Panel.

[Bool+, Bool-, Subd, Smooth, Clean mesh, Decimate, Curves bake, Anim bake, Scope, Hide, Hide scale, Loop, Points, Fills, Root, Shader, Shader type (Principled, Diffuse), Clean GP, Clean factor, Split layers, Split frames, Pressure, Min, Max, Remap mode (Clamp pressure, Remap pressure, Clamp strength, Remap strength)]

5.3.4. Blender latk_ml_004 UI Panel

The latk_ml_004 UI Panel's key creative feature is the ability to choose between three informative-drawings models ("Anime", "Contour", and "Opensketch", trained by the original author) and four Pix2Pix models (trained by me), and apply them in one or two passes.

Fig. 42. Blender latk_ml_004 UI Panel, set to apply the Anime model in one pass.

[004 Frame, 004 All, Model 1 (Anime, Contour, OpenSketch, PxP 001, PxP 002, PxP 003, PxP 004), Model 2 (Anime, Contour, OpenSketch), Line Threshold, Dist Threshold, csize, iter, Source Image (RGB, Depth)]

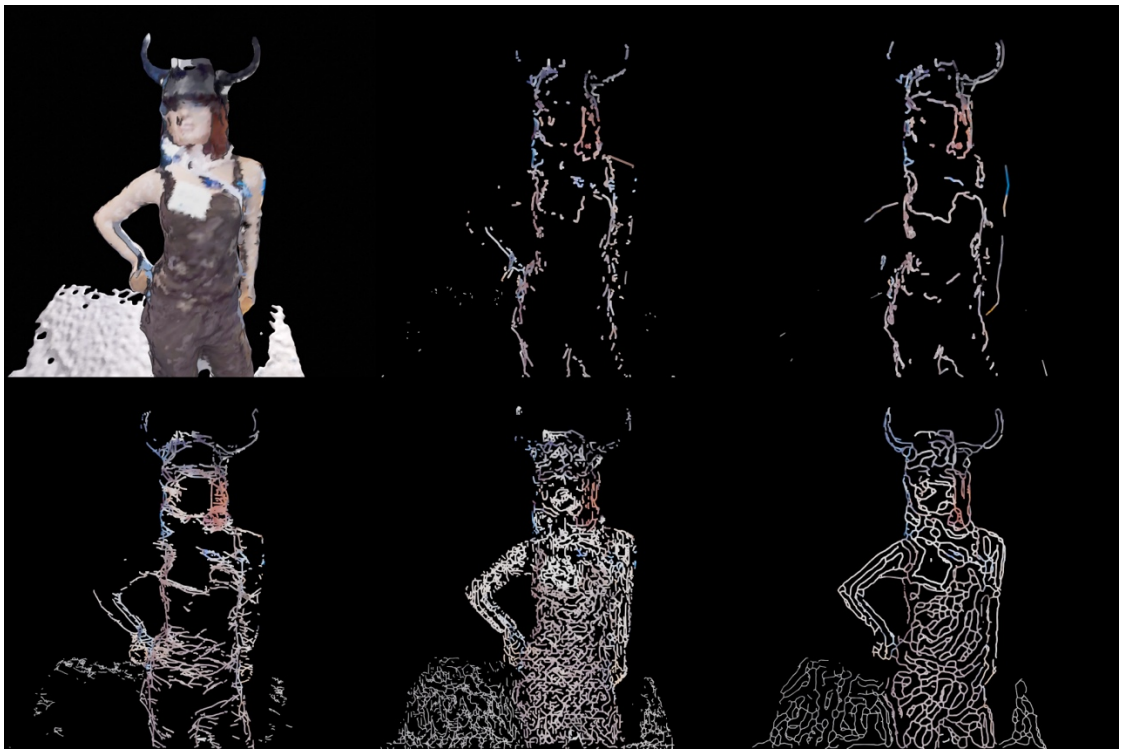

Fig. 43. Demonstrating latk_ml_004 inference results, from top left: original mesh, Anime, Contour, Opensketch + Anime, Pix2Pix 004 + Anime, Pix2Pix 002 + Contour.

5.3.5. Blender latk_ml_003 UI Panel

Finally, the latk_ml_003 UI Panel selects between my Vox2Vox models in three resolutions (64^3, 128^3, or 256^3) and a procedural connection method for the resulting output.

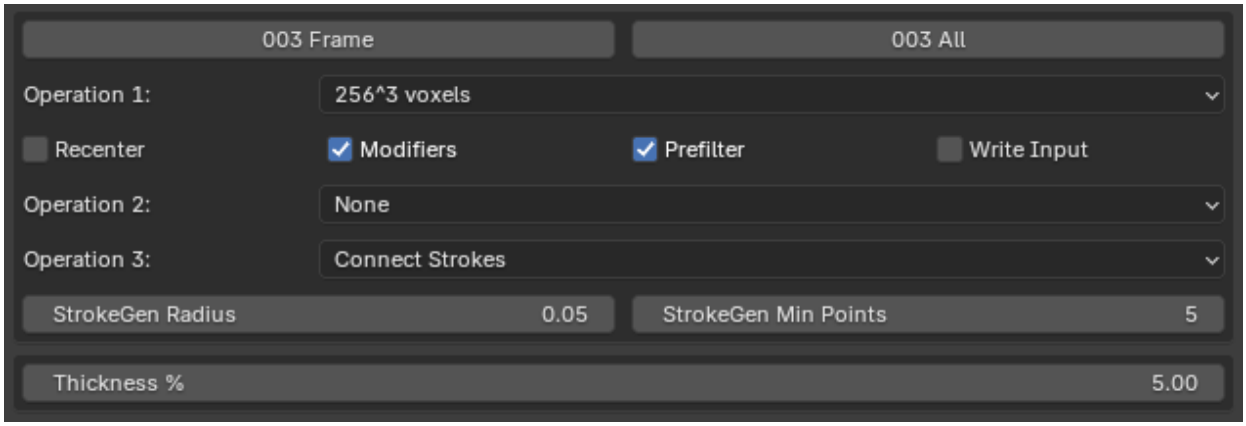

Fig. 44. Blender latk_ml_003 UI Panel, set to run 256^3 inference and nearest-neighbour connection.

[003 Frame, 003 All, Operation 1 (64^3 voxels, 128^3 voxels, 256^3 voxels), Recenter on/off, Modifiers on/off, Prefilter on/off, Write Input on/off, Operation 2 (None, Get edges), Operation 3 (Connect strokes, Growing Neural Gas, GNG + Connect), StrokeGen radius, StrokeGen min points, Thickness %]

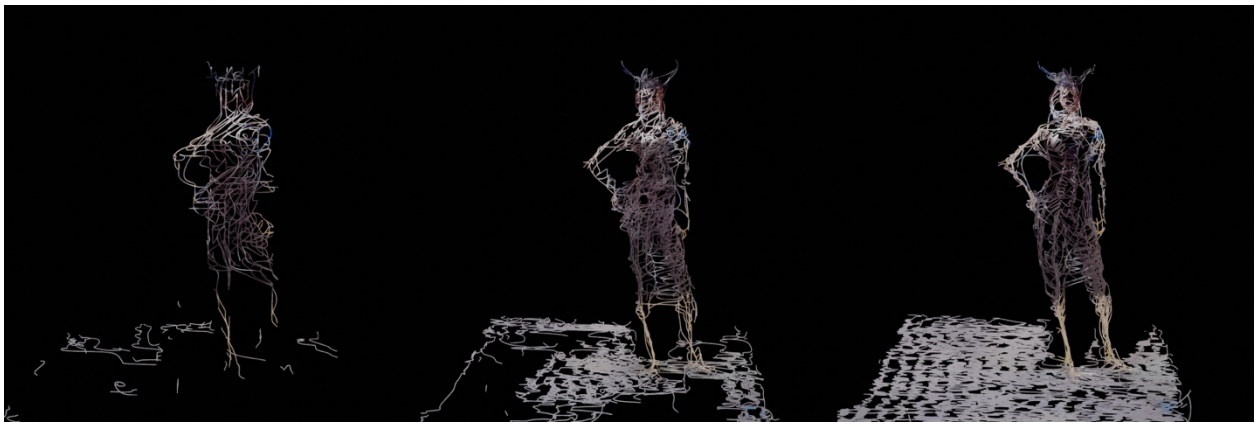

Fig. 45. Demonstrating latk_ml_003 inference results, from left: 64^3, 128^3, and 256^3 voxel volumes.

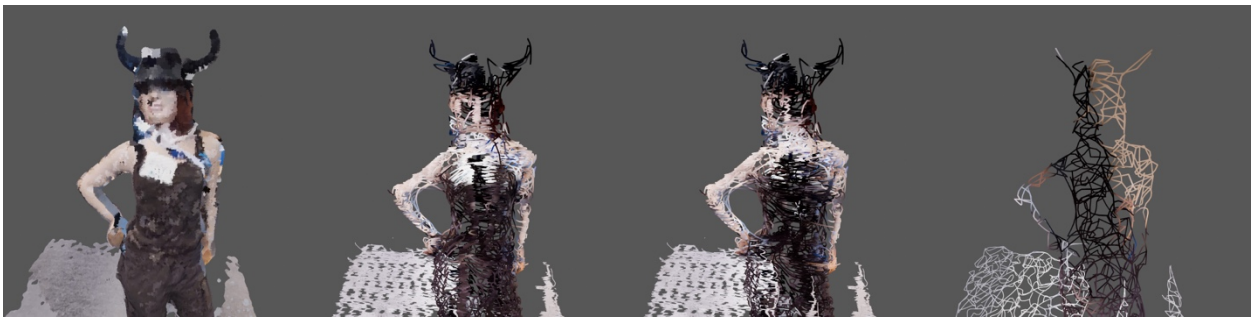

Fig. 46. Demonstrating latk_ml_003 connection results, from top left: original point cloud, basic nearest-neighbour connection, Difference Eigenvalues, and Growing Neural Gas.

Fig. 47. An additional rendering of, from left, the original point cloud and the Difference Eigenvalues connection option.

Fig. 48. A completed Latk animation sequence from *Glasfilm III* (2024), rendered in Blender Dream Textures. The full version of this short film is linked in Appendix A (8.2.1).

6. Conclusion

> "I was considered actually a good camera person, or a decent camera person, but I couldn't take…this fascism of me always deciding what the whole world was going to see. You know, it was this eye and this zooming in and out that was going to decide what the picture would be…And there is no blinking of the eye."
> —Steina Vasulka

> "It was the eye she rebelled against, because she accepted the camera as an instrument, but not the eye as the supreme selective power. So in that way, completely different from a cinematic idea about 'camera eye'…even in computer-generated images for film, it's still the eye as the supreme observer of the event."
> —Woody Vasulka

> "So this is the kind of image attitude I always have…here is the world and I can suck it into my camera, and I can get all those images and they are all mine. And this is the kind of, gluttony, idea of using landscapes that I'm still in."
> —Steina Vasulka (1985)[63]

Taken altogether, the Latk pipeline encompasses all three of the Kino-Eye (volumetric capture), Kino-Brush (hand-drawn animation), and Kino-Stomach (ML post-processing) approaches to digital image-making. However, while the most novel of the three in this context, the Kino-Stomach approach is not itself a new development in experimental cinematic image-making. From the beginning, it has constituted a parallel evolutionary branch that actually adopted computer technology—first analog computers, then digital—while for the most part confining it to a secondary role. This approach persists into the present day—for example, in the analog chemical processing practice of Steven Woloshen (coincidentally, literally evoking organic digestion), the results are scanned and projected digitally because the film becomes too brittle to run through a mechanical projector.[134]

A version of Kino-Stomach that uses digital technology as a primary image-making process briefly emerged in the intermediate era where digital image processing became computationally practical, but storage could not yet hold substantial amounts of footage. (Recall Murch's Digital Sandwich, yet another food-adjacent metaphor.) Hybrid tools like the Video Toaster applied digital effects to whole frames in realtime as they passed between two analog tape decks; the Sandin Image Processor did an "open-face" variant, generating digital images on a computer and sending them through an analog processing chain and ultimately to an analog tape destination.

The subsequent arrival of plentiful digital storage then becomes the takeoff point for the Kino-Brush, enabling the manual editing of frames, and sophisticated intervention in only part of one frame, making the Kino-Brush the preferred mode for approximately the past 30 years. But in the 2020s so far, with diffusion rendering, we have seen the surprisingly rapid return of a now-purely-digital Kino-Stomach. Naïve use of text-to-image and image-to-image rendering usually means a return to operating on the whole frame, just as in the analog video era. The application of *ControlNets*—control layers derived from types of information including edges, depth maps, or normal maps—in diffusion rendering also calls to mind equivalent control layers of earlier systems that used the Kino-Stomach mode.[135] The Video Toaster had a mouse GUI for configuring effects settings; the Sandin system used a keyboard and the GRASS scripting language, which generated visually simple digital graphics that fed into the more complex analog video processors. The Blender addon I use for the diffusion rendering examples in this project, Dream Textures by Carson Katri,[136] uses a node-based interface for configuration; it remains to be seen which UI paradigm will prove most effective in this domain.

But if I can risk a specific prediction, the current excitement around the prompt art genre will eventually fade, because "A map is not the territory".[137] A description of the thing does not contain enough information to fully replace the thing; the illusion of novel complexity mostly comes from retrieving chunks of prior human work from a very large, highly compressed archive. The illusion works, to some extent, simply because the archive is far bigger than we are used to. Serious engagement with the current state of the art in diffusion rendering tools demonstrates the finite capacity of those archives: while vast, for the particular result the individual user is interested in, the number of examples available is inevitably constrained, to the point that even an inexperienced user can find obvious references to the source material with sustained experimentation.

Despite this current limitation, diffusion rendering may still offer a unique suitability for graphics output, as opposed to the similar systems that generate text. The average of all legal opinions applied to a set of initial assumptions cannot be a reliably correct legal opinion, and the average of all medical diagnoses applied to a set of initial assumptions cannot be a reliably correct diagnosis—at least, not without additional computational layers involved to further direct the outcome. But the average of a significant fraction of all the photographs ever taken, applied to a set of sufficient 3D control data, may already be at least as interesting and compelling a route for animation production as firing simulated light rays at that same data. The question then becomes how to provide this control, and the solution may be to use a scene graph—in fact, many current state-of-the-art ML image-to-image pipelines already extract scene-graph-adjacent semantic data from image input.[58] In particular, tooling based on the *USD* (Universal Scene Descriptor) format (increasingly referred to as "OpenUSD" to make it easier to

search for) seems to be a promising next step for Latk.

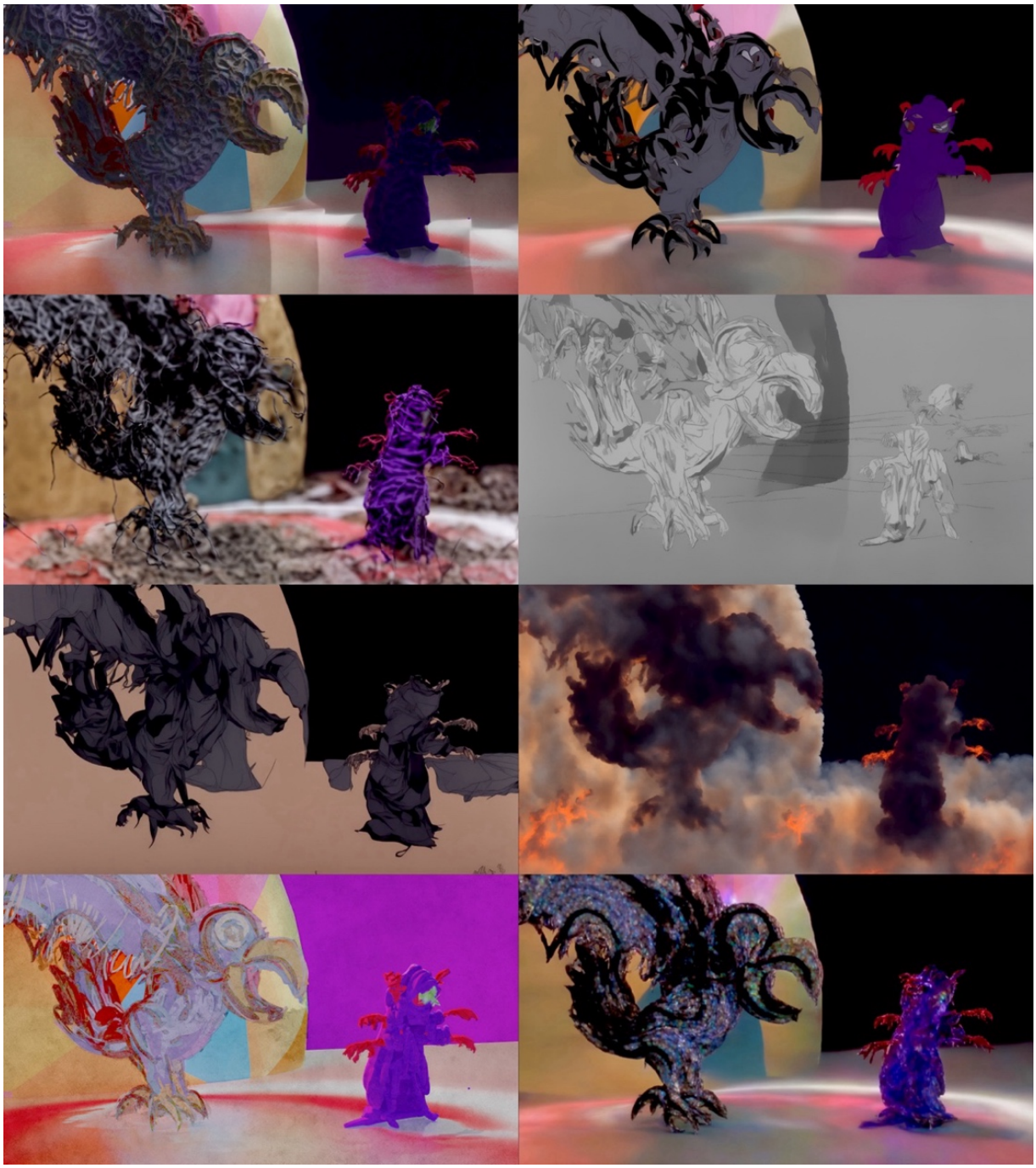

Fig. 49. Eight Blender Dream Textures diffusion renders of the same Open Brush drawing.

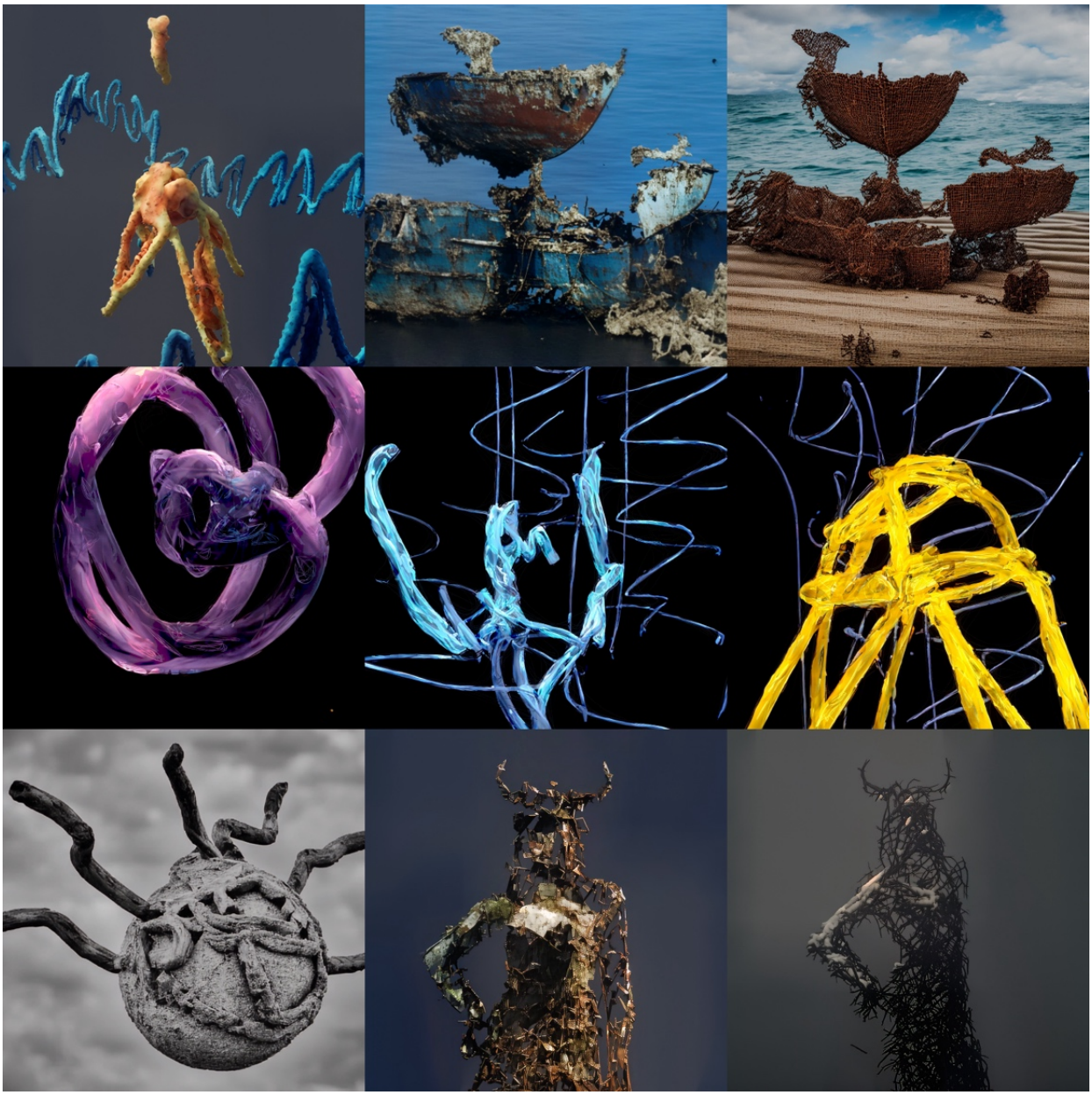

Fig. 50. More diffusion renders of Blender scenes.

Fig. 51. Original point clouds from a volumetric capture (in this case, with calibration errors).

Fig. 52. Nine stills from a diffusion render of the same volumetric point cloud sequence.

There are also potentially significant practical improvements that could still be made to the Latk tools. For example, a separate pass with an ML model designed to fit curves to 3D points (such as Neural Edge Fields by Yunfan Ye et al, or 3Doodle by Changwoon Choi et al) may outperform my existing procedural solutions for the crucial step of connecting raw point cloud output—albeit at the cost of also introducing new limitations, for example a relatively low number of maximum points per stroke or maximum strokes per scene.[138] [139] And the latest versions of Blender expose Grease Pencil to the highly performant native Geometry Nodes system, which (despite introducing many breaking changes to Python scripting) will greatly speed up the task of meshing large quantities of strokes compared to Python code alone, or even Python with optimizations such as SWIG.[115]

In the meantime, I have begun integrating Latk support into major open-source projects where I am a frequent contributor—including multiple Processing libraries; Open Brush, successor to Google's Tilt Brush; and LiveScan3D, the only open fully-functional multi-camera Kinect capture solution. Perhaps most importantly for the long-term viability of Latk, in 2022 I co-founded Common Volume, an international affinity group to further the open-source development of volumetric technology in general and LiveScan3D in particular. Common Volume includes faculty and graduate students from York, OCAD, and Ontario Tech as well as prominent independent media artists, theatre practitioners, and representatives from open-source-friendly volumetric capture companies. This group, and the community it represents, will be my primary means of disseminating Latk. These efforts currently include distributions for package managers like npm, PyPI, the Processing Contribution Manager, and OpenUPM, related curation sites like ofxaddons.com, and plugins for 3D creation apps like Blender, Houdini, and Open Brush.

Fig. 53. *You're Not Wrong* (2020), using the Processing Latk library (with Meagan Williams, Avi Engel).

Fig. 54. *Feed Stairs on Vectrex* (2023), using the openFrameworks Latk addon.

And as interest grows in finding viable alternatives to the retail game engines that dominated artistic software development in the 2010s, Latk has also proven to work very well with three.js, A-Frame, and related web technologies, as the following examples demonstrate:

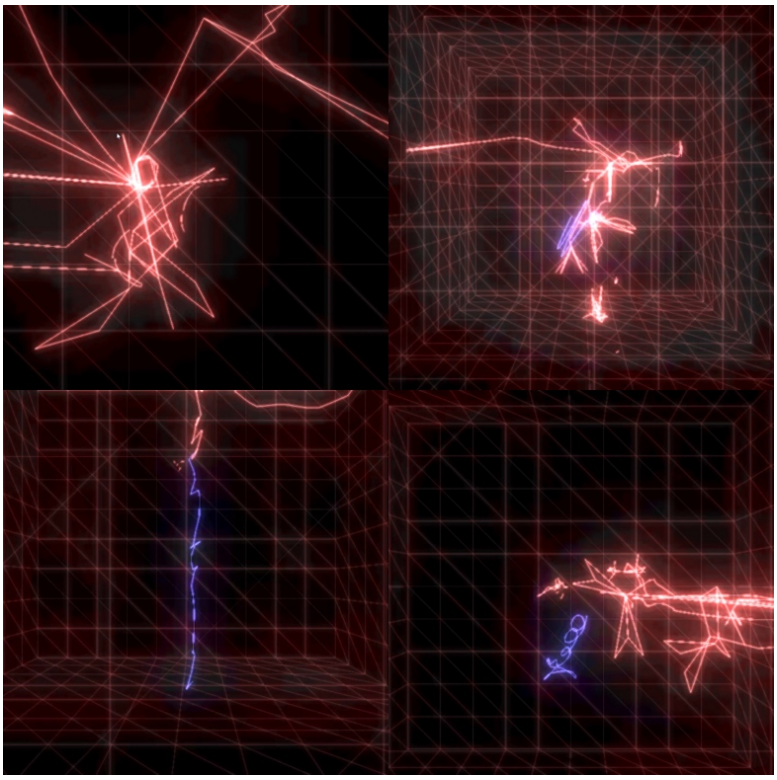

Fig. 55. Multiplayer three.js demonstration—local player's strokes in blue, remote player's in orange.

Fig. 56. Live 3D drawing using the Processing Latk library.

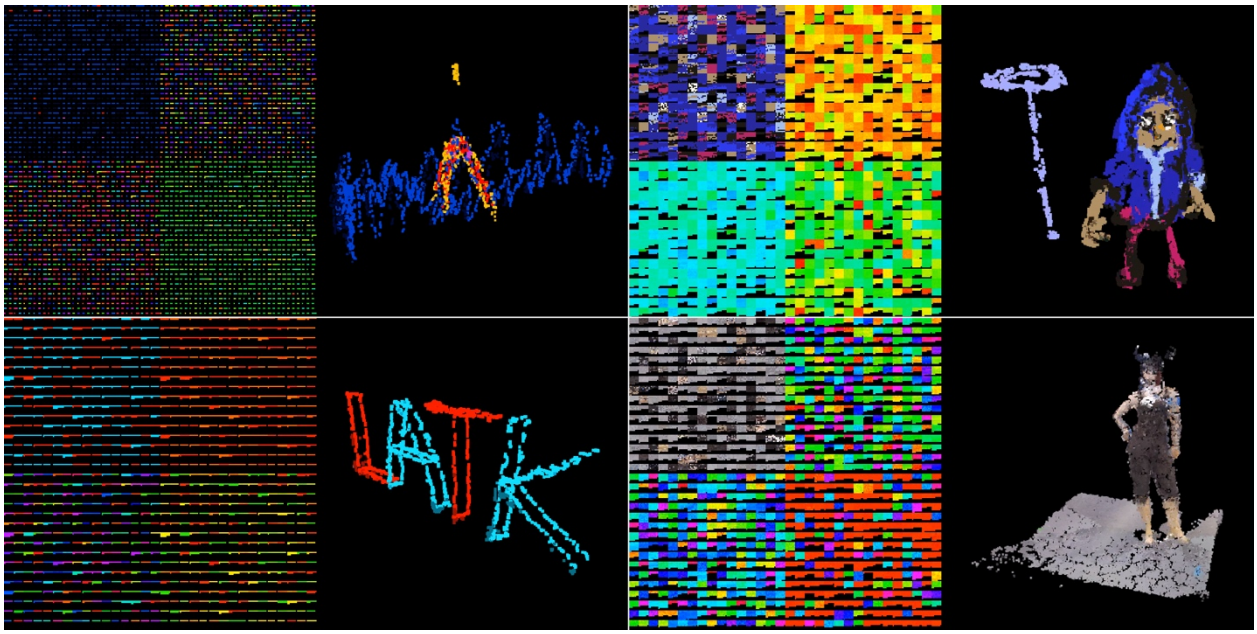

Fig. 57. Latk drawings streamed as point cloud video for use in web AR.

I do think there's cause for optimism in the narrow scope of XR art-making tools, because nearly every technical solution enabling the 2010s "VR Renaissance" was, as a matter of basic science, maturely developed by the end of the 20th century: low-power CPUs, lightweight high-resolution screens, gyroscopes, accelerometers, and 3D tracking methods such as structure from motion. Now, the next cohort of researchers sifts through the ideas of that era, made broadly and inexpensively accessible thanks to a quarter-century of further developments in computational power. As a result, we have a chance to return to a toybox filled with these earlier innovations and craft a set of practical art-making tools. And if we choose this as a goal, we need to consider that many art practices from the first wave of VR—dependent on a handful of expensive installations on the premises of large institutions—were not able to sustain themselves for long without very high levels of institutional recognition and support. So rather than being a secondary consideration, starting with the largest possible audience for these tools was a prerequisite to nurture a future culture of their virtuosic and ethical use—a bar that has now been cleared.

7. References

[1] Moritz, William. “Towards an Aesthetics of Visual Music”. *ASIFA Canada Bulletin*, 1986.

[2] Speicher, Maximilian, et al. “What Is Mixed Reality?,” CHI, 2019.

[3] Yu, Emilie. “Interactive 3D Sketching in VR”. Technical University of Denmark, 2020.

[4] Hogue, Andrew, et al. “A Visual Programming Interface for Experimenting with Volumetric Video”. IEEE GEM, 2022.

[5] Shoup, Richard. “SuperPaint: An Early Frame Buffer Graphics System”. *IEEE Annals of the History of Computing*, 2001.

[6] Glowski, Janice M., ed. *Charles A. Csuri: Beyond Boundaries, 1963–present*. Ohio State University, 2006.

[7] Horne, Chris, et al. “Oculus Story Studio’s Dear Angelica: Inventing, Integrating, and Rendering Quill.” VRDC (presentation), 2017.

[8] Sutherland, Ivan. “Sketchpad: A Man-Machine Graphical Communication System”. MIT, 1963.

[9] Wheatstone, Charles. “On Some Remarkable, and Hitherto Unobserved, Phenomena of Binocular Vision”. 1838.

[10] Jarman, Baird. “Quick as a Flash: Victor Collodion and the Development of the Lightning Artist”. *Nineteenth-Century Art Worldwide*, 2020.

[11] Burtnyk, Nestor, and Marceli Wein. “Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation”. CACM, 1976.

[12] Durno, John. “Glenn Howarth’s Telidon Art”. University of Victoria, 2019.

[13] Robertson, Barbara. “Disney Lets CAPS Out of the Bag”. *Computer Graphics World*, 1994.

[14] St. John, Rebecca. *SANDDE Version 2.1 User Guide*. Janro Imaging Laboratory, 2012.

[15] Ha, David, and Douglas Eck. “A Neural Representation of Sketch Drawings”. Google Brain, 2017.

[16] York University Faculty of Graduate Studies. “Doctoral Dissertation: Doctoral General Requirements”. York University, 2025.

[17] Fox-Gieg, Nick. “Lightning Artist Toolkit: A Hand-Drawn Volumetric Animation Pipeline.” SIGGRAPH, 2024.

[18] Naimark, Michael. “First Word Art / Last Word Art”. *FineArtForum*, 2001.

[19] Lyon, Richard. “A Brief History of Pixel”. IS&T/SPIE Symposium on Electronic Imaging, 2006.

[20] McLaren, Norman. *Pinscreen*. 35mm film, 38m55s, 1973.

[21] Faria Lopes, Pedro. “The Pinscreen in the Era of the Digital Image”. Iberoamerican Congress of Digital Graphics, 1999.

[22] Smith, Alvy Ray. *A Biography of the Pixel.* MIT Press, 2021.

[23] Edwards, Benj. “The Never-Before-Told Story of the World’s First Computer Art”. *The Atlantic*, 2013.

[24] Haas, Maarten de. “The History of Interactive Computer Graphics (Parts 1–5)”. *Wigglepixel* (blog), 2019

[25] Sutherland, Ivan. “The Ultimate Display”. IFIP Congress, 1965.

[26] Baecker, Ronald. “Digital Video Display Systems and Dynamic Graphics”. SIGGRAPH, 1979.

[27] Franklin, Ursula M. *The Real World of Technology*. House of Anansi Press, 2004.

[28] Naimark, Michael. “Apr 12 Notes and Leads”. *A Nonlinear History of New Media* (blog), 2004.

[29] DeFanti, Tom, et al. “BASIC Zgrass: A Sophisticated Graphics Language for the Bally Home Library Computer”. University of Illinois at Chicago, 1978.

[30] Long, Amanda. “Copy-It-Right the Distribution Religion: The Media Archaeology of the Sandin Image Processor”. ISEA Third Summit on New Media Art Archiving, 2023.

[31] Sito, Tom. *Moving Innovation: A History of Computer Animation.* MIT Press, 2015.

[32] Bonin, Vincent. “Digital Image Articulator”. *Fondation Daniel Langlois* (blog), 2004.

[33] Brychkov, Eugeny. *V9938 MSX-VIDEO Technical Data Book Programmers Guide.* ASCII Corp. / Nippon Gakki Co., 1985.

[34] Weibel, Peter. “On the History and Aesthetics of the Digital Image”. Ars Electronica, 1984.

[35] Ehrenberg, Rachel. “Square Pixel Inventor Tries to Smooth Things Out”. *Wired*, 2010.

[36] Porter, Thomas, and Tom Duff. “Compositing Digital Images”. SIGGRAPH, 1984.

[37] Levoy, Marc. “A Color Animation System Based on the Multiplane Technique”. SIGGRAPH, 1977.

[38] @maxplanar. “Bosch FGS 4000.” *Vintage Computer Federation Forums* (blog), 2022.

[39] @MisJiF. “Amiga 500 Demo?” *New Zealand Vintage Computer Forums* (blog), 2015.

[40] Dreher, Thomas. *History of Computer Art. 2nd Ed.* Lulu Press, 2020.

[41] Eve, Martin Paul. *Warez: The Infrastructure and Aesthetics of Piracy.* Punctum Press, 2021.

[42] Finnish Heritage Agency. “Demoscene, Musical Saw Playing and Horsemanship of the Roma: 12 New Elements Inscribed on the National Inventory of Living Heritage”. National Museum of Finland, 2020.

[43] Carlsson, Anders. “The Demoscene as a UNESCO Heritage in Sweden”. *Goto80* (blog), 2025.

[44] Salter, Anastasia, and John Murray. *Flash: Building the Interactive Web.* MIT Press, 2014.

[45] Fleischer, Richard. “100 Years of Rotoscoping”. *Fleischer Studios* (blog), 2015.

[46] Ward, Paul. “Independent Animation, Rotoshop, and Communities of Practice”. *Animation*, 2011.

[47] Murch, Walter. In the Blink of an Eye: A Perspective on Film Editing, 2nd Ed. Silman-James, 2001.

[48] Boers, Mike, and Christopher Prevoe. Immersion Room Demo. SIGGRAPH Toronto (presentation), 2021.

[49] Peddie, John. “Is It Time to Rename the GPU?” *IEEE Computer Society Tech News* (blog), 2018.

[50] Owens, John D., et al. “A Survey of General-Purpose Computation on Graphics Hardware”. Eurographics, 2005.

[51] Tamasi, Tony. “The Evolution of Computer Graphics”. Nvision (presentation), 2008.

[52] Parker, Steven G., et al. “OptiX: A General Purpose Ray Tracing Engine”. SIGGRAPH, 2010.

[53] Turnock, Julie. *Plastic Reality: Special Effects, Technology, and the Emergence of 1970s Blockbuster Aesthetics*. Columbia University Press, 2015.

[54] Waelder, David. "Jimmy Songer and the Development of Video Assist." *695 Quarterly*, 2014.

[55] Seymour, Mike. "Lessons from the Mandalorian". *FXGuide* (blog), 2020.

[56] Saharia, Chitwan, et al. "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding". NeurIPS, 2022.

[57] Ramesh, Aditya, et al. "Zero-Shot Text-to-Image Generation". PMLR, 2021.

[58] Chan, Caroline, et al. "Learning to Generate Line Drawings That Convey Geometry and Semantics". CVPR, 2022.

[59] Mikolov, Tomas, et al. "Efficient Estimation of Word Representations in Vector Space". ICLR, 2013.

[60] Vertov, Dziga. *Kino-Eye: The Writings of Dziga Vertov*. Translated by Kevin O'Brien. University of California Press, 1984.

[61] Manovich, Lev. "What Is Digital Cinema?" 1995.

[62] Benjamin, Walter. "The Work of Art in the Age of Mechanical Reproduction". Translated by Harry Zohn. *Zeitschrift für Sozialforschung*, 1935.

[63] Vasulka, Steina, and Woody Vasulka. "About Video and the Arts: Rene Coelho Interviews Steina & Woody Vasulka". Interview by René Coelho, 1985.

[64] Weinbaum, Stanley G. "Pygmalion's Spectacles". *In A Martian Odyssey and Others*, Fantasy Press, 1949.

[65] Youngblood, Gene. *Expanded Cinema*. E. P. Dutton, 1970.

[66] Patterson, John. "A History of 3D Cinema". *The Guardian*, 2009.

[67] Buxton, William. "From Postcards to Virtual Reality and Back". TorCHI (presentation), 2018.

[68] Heilig, Morton. "Sensorama: The Cinema of the Future". 1955.

[69] Heilig, Morton. "Sensorama MIII Motorcycle". USC School of Cinematic Arts, 1960.

[70] Carlson, Wayne. *Computer Graphics and Computer Animation.* Ohio State University, 2014.

[71] Cruz-Neira, Carolina, et al. "The CAVE: Audio Visual Experience Automatic Virtual Environment". SIGGRAPH, 1992.

[72] Evans, Benedict. "The VR Winter". *What Matters in Tech?* (blog), 2020.

[73] Langolf, Gary, et al. "An Investigation Of Fitts' Law Using A Wide Range of Movement Amplitudes". *Journal of Motor Behavior*, 1976.

[74] Niles, Savannah. "Input and Interaction: Magic Leap, 2015–2017". 2019.

[75] Jiang, Ying, et al. "HandPainter: 3D Sketching in VR with Hand-Based Physical Proxy". CHI, 2021.

[76] Arora, Rahul, et al. "Experimental Evaluation of Sketching on Surfaces in VR". CHI, 2017.

[77] Gray, Elisha. *Telautograph*. Patent US386815, 1888.

[78] Von Drehle, David. *Triangle: The Fire That Changed America*. Grove Press, 2004.

[79] Atkinson, Paul. "A Bitter Pill To Swallow: The Rise And Fall Of The Tablet Computer". *Design Issues*, 2008.

[80] Noll, A. Michael. "Early Microfilm Plotters at Bell Labs". USC, 2015.

[81] Lawson-Tancred, Jo. "Meet Vera Molnár, the 98-Year-Old Generative Art Pioneer Who Is Enjoying New Relevance at the Venice Biennale". *Artnet*, 2022.

[82] Davis, Malcolm, and Thomas Ellis. "The RAND Tablet: A Man-Machine Graphical Communication Device". The Rand Corporation, 1964.

[83] Phillips, Richard L. "Computer Graphics in Court: The Adobe/Quantel Case". *Computer Graphics*, 1998.

[84] Smith, Ernie. "An Early Touchpoint". *Tedium* (blog), 2017.

[85] Keramidas, Kimon, et al. "The Interface Experience: 40 Years of Personal Computing". Bard College, 2015.

[86] Kanji, Takamasu. "The Pen Is Mightier than the Mouse". *Japan Spotlight*, 2006.

[87] Nielson, Howard C., Jr. "Whether the President May Sign a Bill by Directing That His Signature Be Affixed to It". 2005.

[88] Schkolne, Steven. "3D Interfaces for Spatial Construction". CalTech, 2003.

[89] Roberts, Lawrence G. "The Lincoln Wand". MIT, 1966.

[90] Buxton, William, et al. "Interaction at Lincoln Laboratory in the 1960s". CHI, 2005.

[91] Snibbe, Scott, et al. "Springs and Constraints for 3D Drawing". Phantom Users Group Workshop, 1998.

[92] Knep, Brian, et al. "Dinosaur Input Device". SIGCHI, 1995.

[93] Liverani, Alfredo, and Serena Morigi. "Efficient 6DOF Tools for Free-Form Surface Modelling". *The Visual Computer*, 2004.

[94] Barnard, Dom. "History of VR Timeline of Events and Tech Development." *VirtualSpeech* (blog), 2019.

[95] Cruz-Neira, Carolina, et al. "Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE". SIGGRAPH, 1993.

[96] Keefe, Daniel, et al. "Drawing on Air: Input Techniques for Controlled 3D Line Illustration". *IEEE Transactions on Visualization and Computer Graphics*, 2007.

[97] Deering, Michael F. "HoloSketch: A Virtual Reality Sketching/Animation Tool". CHI, 1995.

[98] Wolfe, Jennifer. "IMAX Co-Inventor Roman Kroitor Dies". *Animation World Network* (blog), 2012.

[99] Robertson, Adi. "The Sixense Stem Was Once the Ultimate VR Controller. Where Is It Now?" *The Verge*, 2017.

[100] Lee, Johnny Chung. "Hacking the Nintendo Wii Remote". *IEEE Pervasive Computing*, 2008.

[101] Mosher, Dave. "Six Months Later, Kinect Hacks Flourish". *Wired*, 2011.

[102] Bunting, Geoffrey. "Ghost Hunting, Pornography, and Interactive Art: The Weird Afterlife Of Xbox Kinect". *The Guardian*, 2025.

[103] Schmidt, Alex. “Microsoft Makes Hacking Kinect Easier”. *NPR* (blog), 2011.

[104] Lee, Johnny Chung. “Google Research Bio”. *Google Research* (blog), 2022.

[105] Weichert, Frank, et al. “Analysis of the Accuracy and Robustness of the Leap Motion Controller”. *Sensors*, 2013.

[106] Patel, Nilay. “Jobs: If You See a Stylus or a Task Manager, They Blew It”. *Engadget*, 2010.

[107] Stern, Peter. “Did the New iPhone Lidar Put Velodyne (and Luminar, and Ouster) Out of Business?” *The Latest in Lidar* (blog), 2020.

[108] Williams, Elliot. “Alan Yates: Why Valve’s Lighthouse Can’t Work”. *Hackaday* (blog), 2016.

[109] Hackett, Patrick, and Drew Skillman. “Three Years of Tilt Brush”. VRDC, 2017.

[110] Vitillo, Anthony. “Tilt Five Hands-On”. *The Ghost Howls* (blog), 2022.

[111] Speranza, Filippo, and Laurie Wilcox. “Viewing Stereoscopic Images Comfortably”. SPIE Electronic Imaging, 2002.

[112] Lawrence, Jason, et al. “Project Starline: A High-Fidelity Telepresence System”. *ACM Transactions on Graphics*, 2021.

[113] Levin, Golan. “Painterly Interfaces for Audiovisual Performance”. MIT, 2000.

[114] Barrera-Machuca, Mayra D., et al. “The Effect of Spatial Ability on Immersive 3D Drawing”. ACM Creativity and Cognition, 2019.

[115] Falk, David. “The Next Big Step: Grease Pencil 3.0”. *Blender Developers* (blog), 2023.

[116] Gatys, Leon A., et al. “A Neural Algorithm of Artistic Style”. JOV, 2015.

[117] Ruder, Manuel, et al. “Artistic Style Transfer for Videos”. 2016.

[118] Isola, Phillip, et al. “Image-to-Image Translation with Conditional Adversarial Networks”. 2016.

[119] Liu, Difan, et al. “Neural Contours: Learning to Draw Lines from 3D Shapes”. 2020.

[120] Weber, Martin. “AutoTrace Converts Bitmap to Vector Graphics”. 2016.

[121] Domenico Cirillo, Marco, et al. “Vox2Vox 3D-GAN for Brain Tumour Segmentation”. 2020.

[122] Koch, Sebastian et al. “ABC: A Big CAD Model Dataset for Geometric Deep Learning”. NYU, 2019.

[123] Remde, Christopher, et al. “Sparse Camera Volumetric Video Applications: A Comparison of Visual Fidelity, User Experience, and Adaptability”. Frontiers in Signal Processing, 2025.

[124] Kowalski, Marek, et al. “LiveScan3D: A Fast and Inexpensive 3D Data Acquisition System for Multiple Kinect v2 Sensors”. 3DV, 2015.

[125] Dena Bazazian, et al. “Fast and Robust Edge Extraction in Unorganized Point Clouds”. 2015.

[126] Sergio Orts-Escolano, et al. “3D Colour Object Reconstruction Based on Growing Neural Gas.” IJCNN, 2014.

[127] Wailly, Bastien, and Adrien Bousseau. “Line Rendering of 3D Meshes for Data-Driven Sketch-Based Modeling”. jFIG, 2019.

[128] Zhang, Tongjie, and Ching Yee Suen. “A Fast Parallel Algorithm for Thinning Digital Patterns.” SIGGRAPH, 1984.

[129] Hertzmann, Aaron. “Why Do Line Drawings Work? A Realism Hypothesis”. *Perception*, 2020.

[130] Singha, Abhishek, et al. “A Perspective on Decentralizing AI”. MIT, 2024.

[131] Sporn, Michael. “Pops and Smears”. *Splog* (blog), 2012.

[132] Johnson, Crockett. *Harold and the Purple Crayon*. HarperCollins, 1955.

[133] Rosales, Enrique, et al. “Adaptive General and Predictable VR Ribbon Brush”. *ACM Transactions on Graphics*, 2021.

[134] Woloshen, Steven. *Recipes for Reconstruction: The Cookbook for the Frugal Filmmaker*. Scratchatopia Books, 2010.

[135] Zhang, Lvmin, et al. “Adding Conditional Control to Text-to-Image Diffusion Models”. ICCV, 2023.

[136] Katri, Carson. “Dream Textures: Stable Diffusion for Blender”. *Blender Artists* (blog), 2022.

[137] Korzybski, Alfred. “A Non-Aristotelian System and Its Necessity for Rigour in Mathematics and Physics”. 1931.

[138] Ye, Yunfan, et al. “NEF: Neural Edge Fields for 3D Parametric Curve Reconstruction from Multi-View Images”. 2023.

[139] Choi, Changwoon, et al. “3Doodle: Compact Abstraction of Objects with 3D Strokes”. SIGGRAPH, 2024.

8. Appendix A: Latk Code and Resources

8.1. Websites

8.1.1. **Project website**: https://lightningartist.org

8.1.2. **File specification**: https://lightningartist.org/spec

8.1.3. **Reference viewer**: https://lightningartist.org/viewer

8.1.4. **GitHub org**: https://github.com/LightningArtist

8.1.5. **Open Brush docs**: https://docs.openbrush.app/differences-between-open-brush-and-tilt-brush

8.2. Demos

8.2.1. Example short film *Glasfilm III* (4K video, 0m42s, 2024)

https://fox-gieg.com/glasfilm3.html

https://archive.org/details/glasfilm3

8.2.2. Web demos

https://vr.fox-gieg.com

https://n1ckfg.github.io/latk_video_001/threejs

https://lightningartist.github.io/latk.js/examples/p5js.html

https://lightningartist.github.io/latk.js/examples/threejs.html

8.3. Datasets

8.3.1. **TiltSet dataset**: https://doi.org/10.20383/103.0917

8.3.2. **ABC-Draco dataset**: https://doi.org/10.5683/SP3/QGGXYJ

8.4. Code Repositories

8.4.1. Blender addon

https://github.com/n1ckfg/latk_blender

https://doi.org/10.5281/zenodo.14927542

8.4.2. Unity app for iOS and Android

https://github.com/n1ckfg/latkUnity_ARFoundation

https://doi.org/10.5281/zenodo.14931984

8.4.3. Unity app for Quest/Oculus and Vive

https://github.com/n1ckfg/latkUnity_OpenXR

https://doi.org/10.5281/zenodo.14931980

8.5. Package manager distributions

8.5.1. JavaScript library *[npm install latk]*

https://github.com/LightningArtist/latk.js

https://doi.org/10.5281/zenodo.14933485

8.5.2. openFrameworks addon

https://ofxaddons.com/#:~:text=ofxLatk

https://github.com/LightningArtist/ofxLatk

https://doi.org/10.5281/zenodo.14933475

8.5.3. Processing library

https://processing.org/reference/libraries/#:~:text=Latk

https://github.com/LightningArtist/latkProcessing

https://doi.org/10.5281/zenodo.14933451

8.5.4. Python module *[pip install latk]*

https://github.com/LightningArtist/latkpy

https://doi.org/10.5281/zenodo.14933479

8.5.5. Unity package

https://openupm.com/packages/org.lightningartist.latk

https://github.com/LightningArtist/latkUnity

https://doi.org/10.5281/zenodo.14933477

8.6. Google Colab notebooks

8.6.1. latk_video_001: https://colab.research.google.com/drive/1fikgDB7aYJ3x0iN00TsBhtzSgc__iNPP

8.6.2. latk_ml_003: https://colab.research.google.com/drive/1S0-_CBR-J_-4xvbsk2slAENoZusoNxfc

8.6.3. latk_ml_004: https://colab.research.google.com/drive/1SziTJKHRlxZoJFn8uUcO7u3NU3ncIO81

8.7. Additional examples

8.7.1. Unreal 4 example: https://github.com/LightningArtist/latkUnreal

8.7.2. Blender Python development repos (superseded by Blender add-on)

https://github.com/n1ckfg/latk_ml_001

https://github.com/n1ckfg/latk_ml_002

https://github.com/n1ckfg/latk_ml_003

https://github.com/n1ckfg/latk_ml_004

https://github.com/n1ckfg/latk_ml_005

8.7.3. Unity additional examples

https://github.com/n1ckfg/AltspaceExample

https://github.com/n1ckfg/latkUnity_ARCore

https://github.com/n1ckfg/latkUnity_ARKit

https://github.com/n1ckfg/latkUnity_AzureKinect

https://github.com/n1ckfg/latkUnity_Cardboard

https://github.com/n1ckfg/latkUnity_Tango

https://github.com/n1ckfg/latkUnity_Mirage

https://github.com/n1ckfg/latkUnity_HoloLens

https://github.com/n1ckfg/latkUnity_Kinect

https://github.com/n1ckfg/latkUnity_LeapMotion

https://github.com/n1ckfg/latkUnity_MagicLeap

https://github.com/n1ckfg/latkUnity_Persee

https://github.com/n1ckfg/latkUnity_Hydra

https://github.com/n1ckfg/latkUnity_TiltFive

https://github.com/n1ckfg/latkUnity_Vive

https://github.com/n1ckfg/latkUnity_ViveSR

https://github.com/n1ckfg/latkUnity_ViveXR

https://github.com/n1ckfg/latkUnity_Wacom

https://github.com/n1ckfg/latkUnity_WMR

8.8. App store distributions (not stable for archival purposes)

8.8.1. SideQuest (Quest): https://sidequestvr.com/app/41616/lightning-artist

8.8.2. Apple App Store (iOS): https://apps.apple.com/us/app/lightning-artist/id1204862214

8.8.3. Steam (Vive): https://store.steampowered.com/app/1044670/Lightning_Artist

8.8.4. Google Play (Android): https://play.google.com/store/apps/details?id=com.foxgieg.latk

8.9. Papers

8.9.1. SIGGRAPH 2024: https://doi.org/10.1145/3664221

Fox-Gieg, Nick. “Lightning Artist Toolkit: A Hand-Drawn Volumetric Animation Pipeline”.

8.9.2. IEEE GEM 2022: http://dx.doi.org/10.1109/GEM56474.2022.10017777

Hogue, Andrew, et al. “A Visual Programming Interface for Experimenting with Volumetric Video”.

9. Appendix B: TiltSet Artist Credits

Credits follow for the 13,908 artists who created the 56,723 CC-BY licensed Open Brush artworks used in the TiltSet dataset. Links to the dataset and the credits list in CSV format are supplied in Appendix A (8.3.1); expanded to the 11-point Linux Libertine O font used in the rest of this document, the list would be 43 pages long.