An Experimental Archaeology of CAD

Using Software Reconstruction to Explore the Past and Future of Computer-Aided Design

Daniel Cardoso Llach(✉) and Scott Donaldson

Computational Design Laboratory, School of Architecture,
Carnegie Mellon University, Pittsburgh, USA
dcardoso@cmu.edu

Abstract. This paper proposes software reconstruction as a method to shed new light into the material, gestural, and sensual dimensions of computer-aided design technologies. Specifically, it shows how by combining historical research and creative prototyping this method can bring us closer to distant ways of seeing, touching, drawing, and designing—while raising new questions about the impact of CAD technologies on present-day architectural practices. It documents the development of two software reconstructions—of Ivan Sutherland's "Sketchpad" and of Steven A. Coons's "Coons Patch"—and reflects on the responses they elicited in the context of two exhibitions. The paper shows how software reconstruction can offer access to overlooked aspects of computer-aided design systems, specially their material and sensual dimensions, and how we may explore its broader potential for research, preservation, pedagogy, and speculative design of design technologies.

Keywords: Software reconstruction · Media archaeology · CAD · Sketchpad · Steven A. Coons · Ivan Sutherland · Computational design history

1 Introduction

Computer-aided design (CAD) systems are artifacts of cultural and technical significance which shape the intellectual labor and professional identities of many architects, engineers, and other designers. Recent scholarship on these technologies, and their impact on architecture, has examined their intellectual and institutional origins [1, 2], studied the dynamics of their adoption by practitioners and educators [3–6], examined their implications in professional cultures [7, 8], and explored their effects on architectural organizations and labor [9–11]. However, the usual vehicles of scholarly research—text and, at best, illustrations—fail to account for their material, sensual, and gestural dimensions, which are central to the new experiences, and the new types of practice, that they elicited. This paper reports on Archaeology of CAD, a research project initiated by the first author combining historical research and creative technology design in order to enrich our understanding of these systems by experimentally reconstructing some of its pioneering technologies. Accordingly, the paper introduces software reconstruction as a method of inquiry into computer-aided design and—more

J.-H. Lee (Ed.): CAAD Futures 2019, CCIS 1028, pp. 105–119, 2019.
https://doi.org/10.1007/978-981-13-8410-3_8

generally—software artifacts, with ties to media archaeological and historical reconstruction practices. It then documents our reconstruction of two foundational CAD technologies: Steven A. Coons's "Coons Patch", and Ivan Sutherland's "Sketchpad". It offers details about their development, which involved the analysis of both original archival and oral sources, as well as a creative process involving technological reinterpretations, translations, and adaptations.

The paper further documents the public installation of the two reconstructions in exhibitions at the Miller Gallery at Carnegie Mellon University, Pittsburgh, and in the SIGGRAPH 2018 Art Gallery, Vancouver. It shows how by approximating the logical, gestural, and ergonomic signature of these systems, the reconstructions helped make visible (and tangible) the new forms of design, drawing, and human-machine interaction that emerged with the rise of interactive computing. Finally, the paper discusses some avenues for future work, as well as implications of software reconstruction for scholarly research, pedagogy, preservation, and speculative technology design.

2 Software Reconstruction as Method

Software reconstruction draws inspiration from well-established practices of experimental reworking in archaeology and the historiography of science and technology. Archaeologists studying material culture have long used the term "experimental archaeology" to describe a host of research methods aiming at reproducing the material conditions of specific practices and processes [12]. More recently, historians of science and technology have used experimental reworkings as a complement to textual analysis, and as a method to shed light on gaps in archival documentation—which typically overlooks "sensual" aspects of scientific practices such as smell or touch [13: 91]. In this way reworkings of historical experiments can offer richer portraits of the material and social conditions surrounding scientific and technological production.

Software reconstruction also derives insight from the "undisciplined discipline" of media archaeology [14: 323], which has sought to enrich the analytical repertoire of media scholars by re-covering and re-contextualizing media artifacts, and by reflecting upon the "regimes of memory" they elicit [15: 2]. In a similar vein, recent work in human-computer interaction (HCI) has sought, for example, to enliven material practices of early computing incorporating them into renewed, feminist accounts of the history of technology [16], or to revisit, through playful prototyping, salient artifacts in the history of cybernetics [17]. These practices strand the scholarly and the artistic, and involve the "creative remediation" [15: 142] of technological artifacts as a path towards scholarly and/or creative inquiry. As proposed here, software reconstruction shares with these works a desire to "thinker" with the past [18], and to performatively "re-presence" it [14] in ways that foreground the materialities and dispositions of technological objects, rather than focusing on their narrative disclosures. Unique to software reconstruction as proposed here is its concern with recuperating and reflecting upon the gestural, ergonomic, and visual repertoires enabled by past design tools.

It is important to note that the goal of software reconstruction is not to replicate or restore the original hardware and software systems as an antiquarian would do—this would result on a cybernetic version of Madame Tussaud's museum—filled with

uncanny, glass-eyed look-alikes of technical artifacts. Accordingly, software reconstruction does not require, for example, old mainframe computers or CRT monitors, nor re-writing programs in assembly language. As the examples documented below show, they can be constructed with modern languages (Java and JavaScript), digital fabrication devices (laser cutters and CNC routers), and hardware components (low-cost screens and controllers). However productive exact replicas might be, the more humble—and more agile—technical repertoire of software reconstruction suits best its goal of approximating the experience of using these technologies by enacting their fundamental logic and their key visual, gestural, and ergonomic signatures.

3 Two Reconstructions

3.1 Reconstructing the "Coons Patch"

The "Coons Patch" is a mathematical technique to calculate curved surfaces, developed in the early 1960s by MIT professor of mechanical engineering, computer graphics pioneer, and early CAD theorist and promoter Steven A. Coons [19]. A direct ancestor of non-uniform rational B-splines (NURBS), Coons's technique was, in essence, a clever interpolation algorithm. It allowed early computer graphics researchers to create smooth surfaces between any four parametrically defined curves (Fig. 1). Displayed in the phosphorescent light of CRT monitors, these "patches" were photographed, animated, and then circulated in both research and industry circles through books, films, and research reports. Robin Forrest, one of Coons's students, offers a succinct overview of Coons's technique: "the algorithm provides a means of generating a free-form curved surface from any four arbitrary, boundary-defining curves, parameterized such that for any t between 0 and 1, one can find the point on the curve at parameter t, and with the curves joined at the endpoints" [20]. These patches were key in demonstrating the computer's potential as a modeling and visualization tool with applications in a variety of fields including aeronautic, automotive, and architectural design. Further, they helped trigger a fledgling computer graphics community as it formed across dispersed university and industry laboratories on both sides of the Atlantic—many of whose members came to see Coons as an inspiring, founding figure [1: 49–72]. The "Coons Patch" thus foreshadowed present-day methods for parametric surface representation and manipulation, which are ubiquitous in architectural, engineering, and product design today.

As an algorithm, the "Coons Patch" does not have an associated hardware interface—it is "platform independent". Its earliest implementations involved direct manipulation of numerical values in matrices, and working with memory registers in mainframe computers such as the TX-2 at Lincoln Labs [21]. Given that a curve in its cubic form can be represented mathematically by two end points and two control points, a "Coons Patch" may be represented by 36 floating point numbers: four three-dimensional endpoints, and eight three-dimensional control points. Individually adjusting 36 numbers in order to transform a design would surely exceed the patience of the average present-day designer, but this is how a computational designer of this era operated. For example, a 1967 Ford Motors product research film produced at the MIT

Lincoln Labs shows the construction of patches and their placement as car parts [22]. The process of preparing this two-and-a-half minute animation "hardcoding" the numerical values describing the shape of the patch at each frame—as well as the coordinates of the camera's position—must have been painstaking.

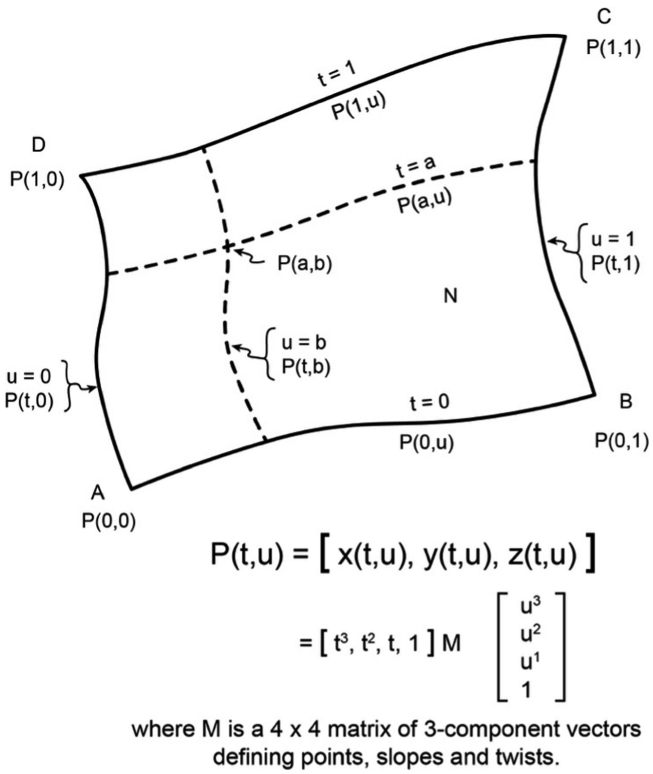

Fig. 1. The "Coons patch" foreshadowed modern methods of surface representation, such as NURBS, which have since become commonplace in architectural and engineering design. Image based on a drawing by Robin Forrest (c. 1970).

Our reconstruction streamlines this process in order to emphasize the geometric plasticity of Coons's technique, and make visible its mathematical structure. A small keypad interface and a knob placed on a 10″ × 10″ podium in front of a projection wall allow users to toggle the point controls on and off, and manually move them along the X, Y, and Z axes. When points are active, their changing numerical coordinates are displayed, making visible the mathematical structure of the patch as the user transforms it (Fig. 2). Aside from manipulating individual points in sequence, a user may choose to randomly transform the entire surface.

Our reconstruction does this by translating each end and control point by a random amount within a bounded range for each dimension, and animating the transformation between the current and target states. Users may then rotate or zoom the camera to view

the surface from another perspective, manipulate individual points, or restore the surface to its initial form—a unit square on the XY plane, centered at the origin. When left unattended, the patch randomly transforms itself every 20 s.

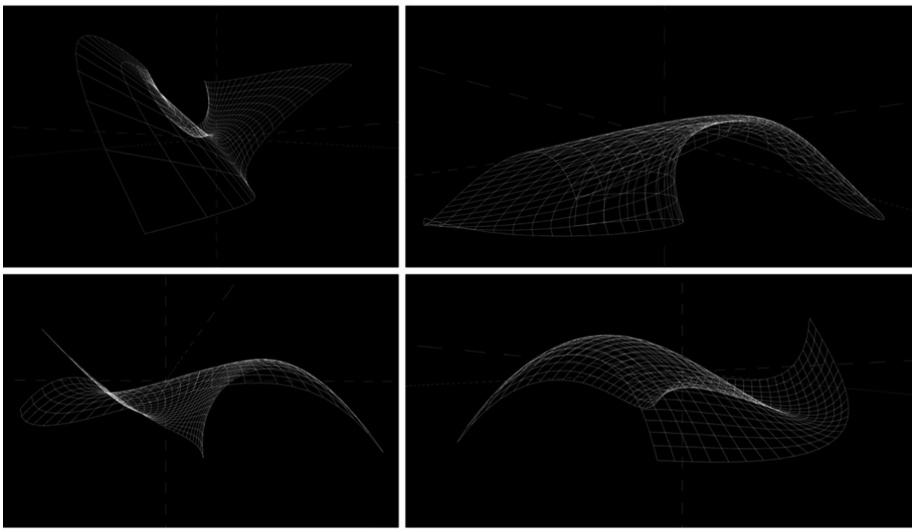

Fig. 2. Examples of patches generated using the software reconstruction of the "Coons Patch". The images illustrate the geometric plasticity of shapes generated using Coons's technique.

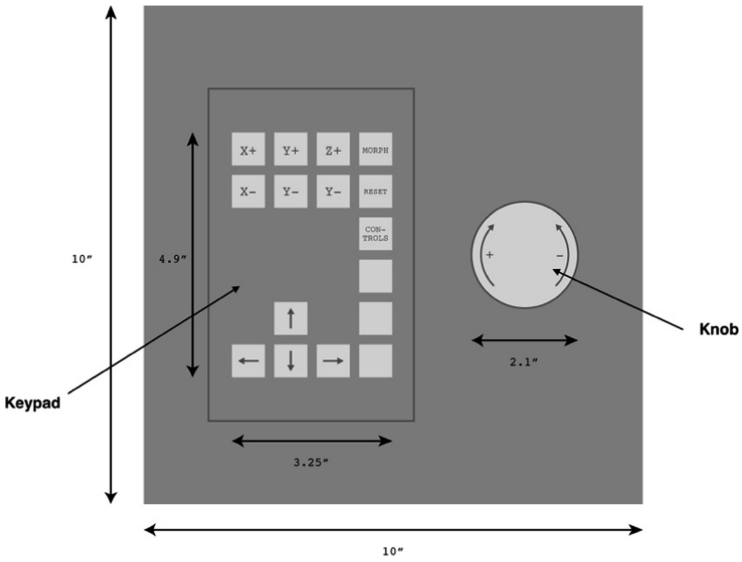

Fig. 3. A simple hardware interface allows users of the reconstruction to visualize and manipulate "Coons Patches".

Our reconstruction is a web-based app implemented using Three.js, a 3d graphics and rendering library for the web and React, a user interface development framework streamlined to work with present-day browsers. An advantage of Three.js is its harnessing of WebGL, a powerful low-level language for web graphics that is hardware-accelerated to optimize computational geometry and rendering. React is a popular component-based framework that streamlines the process of making HTML elements interactive, and centralizes the state of an app. In this reconstruction the state may change depending on whether a user is manipulating the geometric surface, rotating the camera, or viewing the tutorial. The app's visual display updates based on this state (for example, to zoom in or out, or show the tutorial overlay).

On the hardware side, a keypad and control knob integrate with the "Coons Patch" software via user events native to the web (Fig. 3). The app 'listens' for keypress and scroll events and, depending on the React state, performs computations and updates the visual display, a wall projection, accordingly.

3.2 Reconstructing "Sketchpad"

"Sketchpad" is a computer program developed in 1963 by Ivan Sutherland, which famously pioneered graphic user interfaces and laid out the foundations of present-day CAD systems. "Sketchpad" was the centerpiece of Sutherland's PhD thesis at MIT, for which Steven A. Coons was an advisor [23]. In contrast with Coons's "patch", Sutherland's "Sketchpad" was platform specific. It was in fact made possible by the TX-2 computer at Lincoln Labs, which allowed Sutherland to develop "Sketchpad" as an interactive, rather as a batch processing, system. Using a "light pen", a keypad, and control knobs "Sketchpad" users could conduct a variety of drawing operations on $7'' \times 7''$ CRT monitor. Aside from functions for drawing lines, circles, polygons, and other shapes, Sutherland also implemented functions that extended beyond the capabilities of traditional drafting media. For example, he included "save" and "transform" functions, as well as "rubber-banding" and "linkages"—known today in parametric modeling lingo as "constraints". Notably, many of the innovative features that Sutherland included in "Sketchpad" remain central to modern 3-D modeling and computer-aided design systems today, more than 60 years after its development. "Sketchpad" thus remains a milestone in the history of computing, and an essential point of reference for computer-aided design.

Sutherland programmed "Sketchpad" directly in assembly language on the TX-2 computer—its entities were written, retrieved, and manipulated directly in the computer's memory at the hardware level. In order to store these entities, Sutherland used an innovative data structure he calls "n-component elements". The "n" refers to the variable size of geometric entities—for example, a line segment may require less storage space than a more complex shape. Objects occupying variable amounts of space in memory could be stored as "*n* consecutive registers in storage", all locatable in TX-2's core memory [23: 35]. Thanks to the higher-level abstractions made available by present-day programming languages, this way of working—addressing memory locations directly in the hardware—is mostly a relic of the past. In our reconstruction geometric entities are simply instances of custom objects implemented in the object-oriented programming language Java. A line segment in our reconstruction, for

example, contains references to two points, which we call *p1* and *p2*. This is nearly identical to the way "Sketchpad" stored line segments by referring to its points externally. However, instead of relying on machine-specific registers and memory addresses, we use object named classes and instance data. This abstraction, as well as the support of drawing APIs and the Android software development kit (SDK), allow our code to run, with small concessions, on any Android device. Approximating the original "Sketchpad" interface, a user may use a combination of "light pen" and knobs gestures to draw geometric elements on the screen and to move, manipulate, copy, or delete them (Fig. 4). It is also possible to create compound objects such as, for example, a triangle inscribed in a circle, and for the individual entities to then act as one —moving, scaling, rotating, copying, and deleting in tandem.

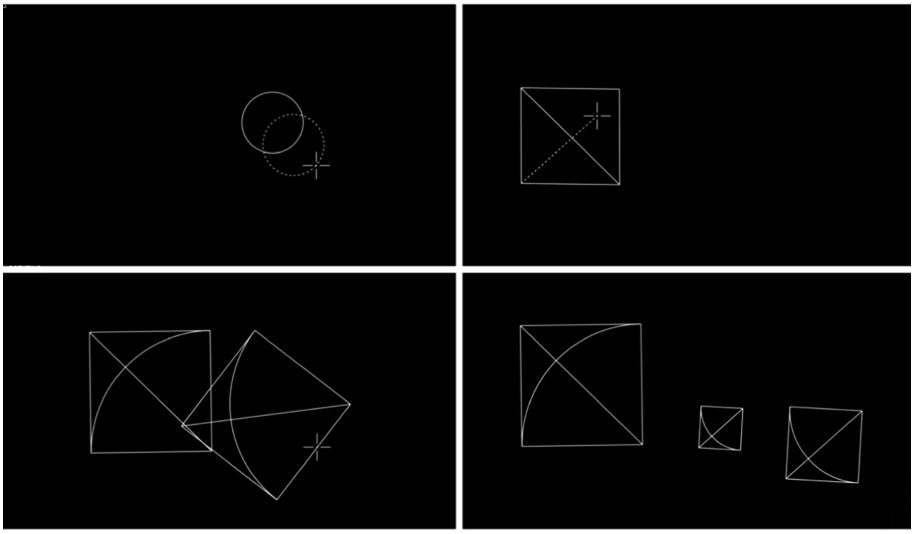

Fig. 4. Examples of drawings made using the "Sketchpad" reconstruction. Top left illustrates the "copy" function. Top right illustrates "rubberbanding". Bottom left illustrates the "rotate function". Bottom right illustrates the instantiation of compound objects.

Our reconstruction relies on modern devices to approximate many of "Sketchpad's" functions, as well as its key visual, gestural, and ergonomic traits. It evokes the TX-2's original user interface of flywheels, switches, and keypad with low-cost hardware including an Android tablet, a series of control knobs, a keypad, and a stylus (Fig. 5). These hardware elements are organized in a custom work station designed to approximate the ergonomics of the TX-2 desk. Instead of reconstructing or restoring an actual mainframe computer, we use a small Android touch-screen tablet. Instead of using a "photodiode and transistor preamplifier" [23: 55] as a light-pen, we use the tablet's stylus, which requires no additional hardware or software integration.

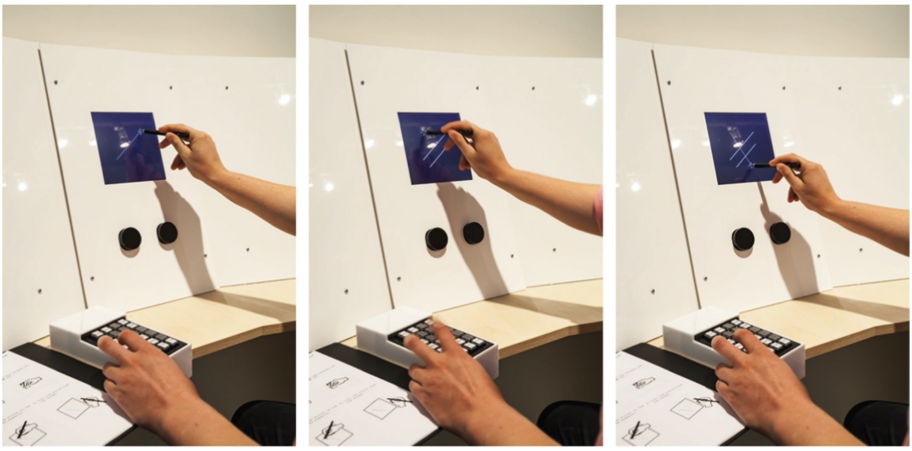

Fig. 5. Demonstration of a drawing procedure using the "Sketchpad" reconstruction. Photo credit: Tom Little (2017).

4 Results

This section documents the software reconstructions in the context of two different exhibitions, and our observations of their engagements with the public.

4.1 Engaging an Architectural and Academic Audience

The reconstructions were first included as elements of the design of the exhibition *Designing the Computational Image, Imagining Computational Design*, curated by the first author. The exhibition, which was on display between September 22 and November 12 of 2017, showcased "rare photographs, film, high-quality reproductions, and interactive software reconstructions examining the formative period of numerical control and Computer-Aided Design technologies, along with a selection of experimental work by computational designers working today" [24]. The exhibition, which was free and open to the public, was visited by an estimate of 1,600 people, in addition to the 200 who visited the exhibition during the opening. In this context, the software reconstructions complemented the exhibition's curated artworks, which comprised mostly visual materials.

The "Coons Patch" reconstruction was placed in a section entitled "Structured Images" among a selection of historical materials describing the origins of computer graphics and computational geometry. Directly behind the reconstruction was a series of handwritten notes by Steven A. Coons describing the technique, and its potential for design (Fig. 6). Nearby video screens displayed works from Lincoln Labs [22] along with contemporary pieces. The "Sketchpad" reconstruction was placed in a separate section of the exhibition entitled "Interaction and Intelligence" among a selection of materials and artworks describing the influence of early artificial intelligence ideas in design (Fig. 7). "Sketchpad" was placed besides a 1963 procedurally-generated

painting by design theorist George Stiny. Juxtaposing two radically different paradigms of computational design—one visual, and another numerical—were thus juxtaposed.

Aside from offering additional context to the historical and contemporary pieces, the reconstructions helped enliven the space and became points of attraction. Additionally, as weeks passed bugs and areas of improvement were identified, some of which were addressed for the reconstructions' second show.

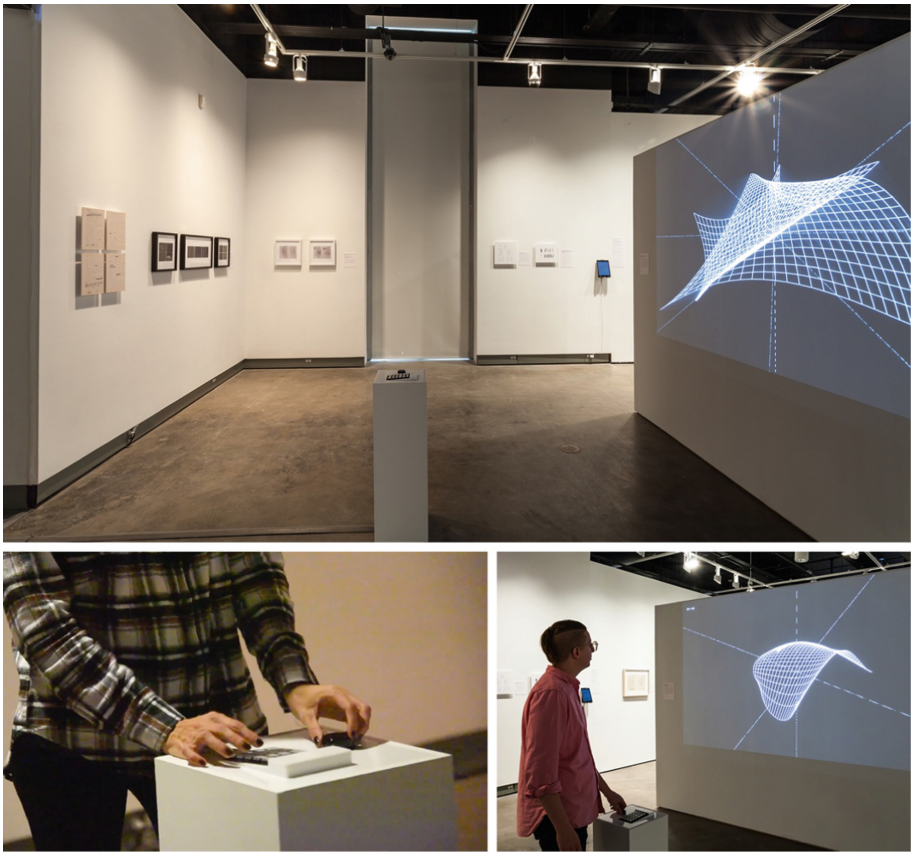

Fig. 6. At the Miller Gallery, the "Coons Patch" software reconstruction was installed among a curated selection of materials from the history of computer graphics. Image Credits: Tom Little (top and bottom right) and Joshua Brown (bottom left).

Fig. 7. The reconstruction of "Sketchpad" was placed in the "Interaction and Intelligence" section of the exhibition, among artworks that described the postwar intersection of ideas about artificial intelligence, cognition, and design. Photo credits: Tom Little (2017).

4.2 Engaging a Diverse, Computer-Focused Audience

The second exhibition took place in the SIGGRAPH 2018 Art Gallery in Vancouver. The show was "conceived as a dialogical space that enables the viewer to reflect on man's diverse cultural values and rituals through contemporary creative practices" and explored the concept of "origins"—broadly understood to encompass not only the technological but also the cultural and ethnic dimensions of the term.

Accordingly, in contrast to the previous exhibition, which had a clear focus on the history and contemporary practice of computer-aided design, this exhibition

encompassed a more diverse selection of artworks and themes, including new media artworks by native American artists, interactive films, and historical pieces by computational art pioneers, curated by Andres Burbano. Featured artists included Ruth Wilson, Skawenatti, Nicole L'Huillier, and John Edmonds, among others. During this relatively short period the reconstructions were used by hundreds of people.

Fig. 8. Top: A knowledgeable visitor uses the "Sketchpad" reconstruction to explain the concept of parametric linkages. Bottom: The reconstruction encouraged different kinds of social interaction—from the playful to the rigorous. Photos by the authors.

In this context, the reconstructions addressed the exhibition's concept of "origin" by invoking the earliest computer graphics and computer-aided design techniques, which were central to the origin of the conference itself. Given the different context, both pieces were concentrated in a single space—on opposite sides of a "fat wall". Which also and incorporating a tablet for the interactive display of contextual information about the pieces. The whole installation occupied an area of 120″ × 190″ with

the "fat wall" dividing it through the short side at 60″. Instead of additional pieces from the history of CAD, on the "Sketchpad" side a wall mounted iPad was installed offering additional context through a selection of historical materials and films.

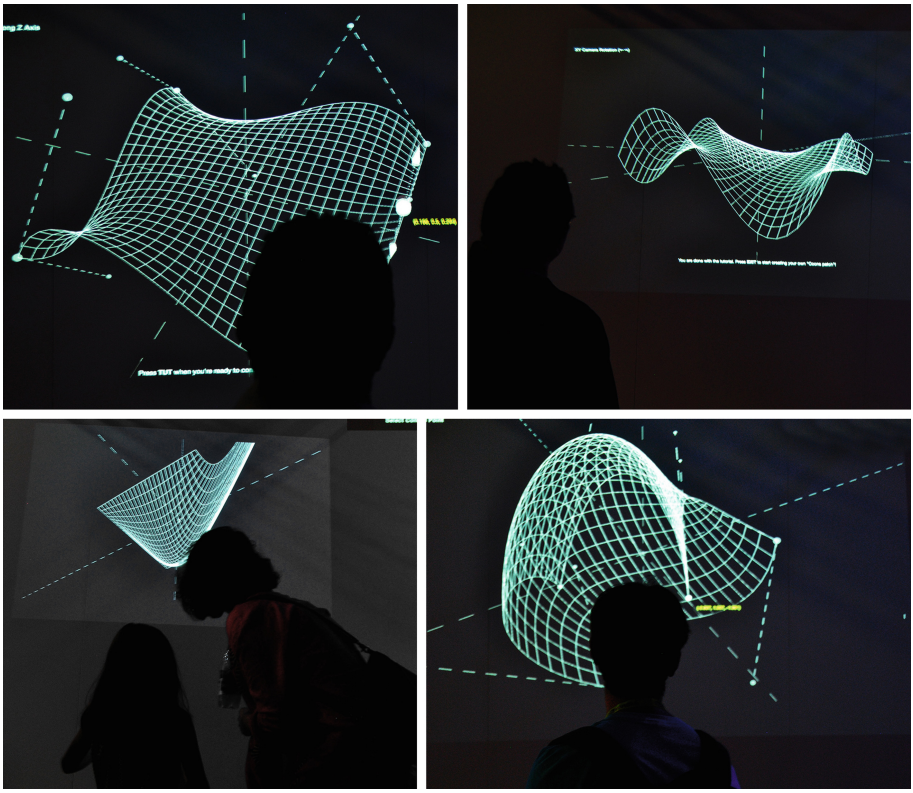

Fig. 9. The "Coons Patch" reconstruction attracted different kinds of users: the geometry savvy, and the curious. Photos by the authors.

Located among a diverse collection of artworks—many of which employing more sophisticated technologies—the reconstructions were a constant point of attraction for visitors and groups in the gallery, many of whom recognized the historical context and welcomed the opportunity to engage with it visually and tangibly. At times, the reconstructions became pedagogical tools visitors used to explain the core concepts of the technologies (Figs. 8 and 9).

5 Discussion

Two key insights can be drawn from the above experiences concerning the potentialities of software reconstruction as a method in the study of computer-aided design and—more generally—interactive technologies.

Defamiliarizing Technologies to See Them Anew
The software reconstructions forced experienced users of CAD systems to temporarily "unlearn" the logic of present-day CAD software in order to adopt an unfamiliar repertoire of postures and gestures. It was common during the two exhibitions to see such users spending a significant amount of time—between 5 and 10 min—in a mixture of amusement and bewilderment, trying to figure out how to use the pen, knobs, and keypad, to draw. A tutorial left on the "Sketchpad's" desk helped guided them on this. "Sketchpad" thus had the effect of de-familarizing conventional CAD tools, and—by invoking an alien paradigm of interaction—prompting a reflection on the embodied experience of designing computationally both *then*, and *now*.

De-mystifying Technological Novelty to Explore the Poetics of Computation in Design
Beyond the functional advantages they offered, architects adopted computer-aided design software because of the novel visual languages that they made possible [25]. However, close to sixty years after their inception and widely adopted by the profession, these languages are far from novel. As a result, computationally-generated imagery has become a new vernacular language of architectural representation, prompting a "post-digital" reaction in many present-day architects and designers. This attitude is often expressed through a nostalgic attachment to analog drawing media and (somewhat ironically) through the simulation of these analog media through software remediations. While architects in the "post-digital" space reserve the romanticism to hand drawing, the software reconstructions described here recuperate the undeniable sense of wonder that arises when one draws a line on a screen and stretches it, twists it around, and entangles it with others to form more intricate shapes.

6 Conclusion

This paper has outlined software reconstruction as a method combining historical research and creative technology prototyping, and documented its potential to shed new light into the material, gestural, and sensual dimensions of computer-aided design technologies. Specifically, it shows how software reconstruction can bring us closer to distant ways of seeing, touching, drawing, and designing while raising new questions about the impact of these historical practices on our present. As playful artifacts of scholarly and design inquiry, software reconstructions can operate at multiple registers, and for multiple publics.

From an architectural and design perspective, software reconstruction confront us with the origins of a visual, gestural, and ergonomic repertoire that has shaped the ideas, practices, and professional identities of those in these fields over the last 30 years. By creating a situation in which present-day CAD systems are *defamiliarized*, software reconstructions create the opportunity for the tacit knowledge [26] involved in their operation to be recuperated and made subject to historical analysis. In other words, software reconstruction makes computer-aided design technologies visible (and tangible) again as devices that fundamentally re-structure architectural labors, bodies, and the intellectual life of practitioners. While focusing on CAD, software reconstruction is naturally transferable to other systems.

From a media and science and technology studies perspective, we acquire a deeper understanding of the decision-making process behind the systems we study, and of the logical and material constraints confronted by their authors. Experimentally reconstructing "Sketchpad", for example, highlighted for us the limitations of early data structures for geometric representation and hardware, and the critical interplay between these constraints and the image of design that emerged in conjunction with it. This insight was organic to the messy processes of adapting, translating, and re-interpreting the system's functionality in a modern programming language, and would be difficult to acquire (let alone convey) through documentary accounts. Therefore, software reconstruction complements and adds nuance to our reading of the historical texts and interpretation of oral accounts.

Finally, from a pedagogical perspective, the process of making experimental reconstructions is useful to introduce students to concepts of interaction, programming, digital fabrication, and hardware design. Software reconstruction then not only enriches our understanding of technological and design histories, outlining the origins of contemporary architectural languages and subjects, but also engages with present-day technological frameworks, thus opening avenues for speculative tool-making and design.

Acknowledgments. The first author wishes to thank CAD pioneers Timothy E. Johnson, Robin Forrest, and Malcolm Sabin for valuable contributions to this project through interviews, documents, and informal conversations—any inaccuracies and undue liberties in this paper are not their fault; Francois Penz, The Martin Centre of Architectural and Urban Studies at the University of Cambridge, UK, and Allan Blackwell, who helped make some of these conversations possible; and The Graham Foundation for Advanced Study in the Fine Arts and the Berkman Fund for Faculty Development at Carnegie Mellon, for granting essential material support. Thanks also to Margaret Cox and Kara Skylling at the Miller Institute of Contemporary Art for valuable assistance as coordinators of the 2017 exhibition *Designing the Computational Image, Imagining Computational Design*. Thanks to the SIGGRAPH 2018 Art Gallery team, specially to curator Andres Burbano for selecting the two installations for the *Original Narratives* exhibition in Vancouver, and to Elizia Artis for expert management and coordination.

References

1. Cardoso Llach, D.: Builders of the Vision: Software and the Imagination of Design. Routledge, London; New York (2015)
2. Steenson, M.W.: Architectural Intelligence: How Designers and Architects Created the Digital Landscape. The MIT Press, Cambridge (2017)
3. Fallon, K.K.: The AEC Technology Survival Guide: Managing Today's Information Practice, 1st edn. Wiley, Hoboken (1997)
4. Andia, A.: Reconstructing the effects of computers on practice and education during the past three decades. J. Archit. Educ. **56**(2), 7–13 (2002). https://doi.org/10.1162/104648802604725 12
5. Coyne, R.: The impact of computer use on design practice. In: Computer Aided Architectural Design Futures: Education, Research, Applications [CAAD Futures 1991 Conference Proceedings/ISBN 3-528-08821-4] Zürich (Switzerland), pp. 413–424 (CUMINCAD 1991), July 1991. http://papers.cumincad.org/cgi-bin/works/paper/403c

6. Akin, Ö.: Computational design instruction: toward a pedagogy. In: The Electronic Design Studio: Architectural Knowledge and Media in the Computer Era [CAAD Futures 1989 Conference Proceedings/ISBN 0-262-13254-0] Cambridge (Massachusetts/USA), pp. 302–316 (CUMINCAD 1990) (1989). http://papers.cumincad.org/cgi-bin/works/paper/450c
7. Downey, G.L.: The Machine in Me: An Anthropologist Sits Among Computer Engineers. Routledge, Abingdon (1998)
8. Loukissas, Y.A.: Co-Designers: Cultures of Computer Simulation in Architecture. Routledge, New York (2012)
9. Gutman, R.: Architectural Practice: A Critical View, 5th edn. Princeton Architectural Press, New York (1997)
10. Yaneva, A.: Mapping Controversies in Architecture, 1 edn. Routledge, London; New York (2016)
11. Cardoso Llach, D.: Architecture and the structured image: software simulations as infrastructures for building production. In: Ammon, S., Capdevila-Werning, R. (eds.) The Active Image. PET, vol. 28, pp. 23–52. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56466-1_2
12. Ingersoll, D., Yellen, J.E., Macdonald, W. (eds.): Experimental Archeology. Columbia University Press, New York (1977)
13. Fors, H., Principe, L.M., Sibum, H.O.: From the library to the laboratory and back again: experiment as a tool for historians of science. Ambix **63**(2), 85–97 (2016). https://doi.org/10.1080/00026980.2016.1213009
14. Sobchak, V.: Afterword: media archaeology and re-presencing the past. In: Huhtamo, E., Parikka, J. (eds.) Media Archaeology: Approaches, Applications, and Implications, pp. 323–333. University of California Press (2011)
15. Parikka, J.: What Is Media Archaeology? 1 edn. Polity, Cambridge (2012)
16. Rosner, D.K., et al.: Making core memory: design inquiry into gendered legacies of engineering and craftwork. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, pp. 531:1–531:13. ACM, New York (2018). https://doi.org/10.1145/3173574.3174105
17. Pangaro, P., et al.: "Colloquyofmobiles," Colloquy 2018 (2018). https://www.colloquyofmobiles.com
18. Huhtamo, E.: Thinkering with media: on the art of paul DeMarinis. In: Beirer, I., Himmelsbach, S., Seiffarth, C. (eds.) Paul DeMarinis: Buried in Noise, pp. 33–46. Kehrer Verlag, Heidelberg and Berlin (2010)
19. Coons, S.A.: Surfaces for Computer-Aided Design of Space Forms, MAC-TR, 41. M.I.T. Project MAC, Cambridge (1967)
20. Forrest, R.: Interview with the first author (2016)
21. Johnson, T.E.: Personal communication with the first author (2016)
22. Surface Generation by Computer: Ford Motors Product Research at MIT Lincoln Labs, 16 mm (1967). Courtesy of Timothy E. Johnson
23. Sutherland, I.E.: Sketchpad, a Man-Machine Graphical Communication System. Massachusetts Institute of Technology (1963)
24. Cardoso Llach, D.: Designing the Computational Image, Imagining Computational Design (Exhibition Catalogue). Carnegie Mellon University School of Architecture, Pittsburgh (2017)
25. Bruegmann, R.: The pencil and the electronic sketchpad: architectural representation and the computer. In: Blau, E., Kaufman, N. (eds.) Architecture and Its Image. Montreal (1989)
26. Collins, H.: Tacit and Explicit Knowledge. Reprint edition. University of Chicago Press, Chicago (2012)