I will begin with a slightly unusual telling of a much-rehashed narrative in the history of architectural theory. In a 1957 essay, John Summerson suggested that architects would need to reconstruct their discipline around programming – they should give up designing form and start to think about organizing patterns of activity. A few architects took Summerson’s advice to heart. Christopher Alexander wrote software to manipulate architectural programs; Cedric Price seems to have spent most of his time gathering data and drawing diagrams. Two senses of the term “programming” developed simultaneously in this era. In architecture, programming referred to the practice of organizing functional spaces, while in the field of computation, programming referred to the practice of creating software. The craze for “spatial location-allocation” software around 1970 is one example of how these two senses of programming came together: computer programs could be used to manipulate architectural programs. Price’s Fun Palace (1964), with its computer-controlled flexible spaces, went so far as to imagine that computer programming and architectural programming are fundamentally the same thing.

But just as some architects embraced programming in the 1960s and ’70s, a branch of architecture theory (which I will refer to as “Theory”) gained traction precisely in opposition to this trend. Peter Eisenman’s famous 1976 editorial described how over the previous centuries “architecture became increasingly a social or programmatic art,” just as Summerson had noted. Contrary to Summerson, however, Eisenman suggested adopting a “non-humanistic attitude” that begins with the “negation of functionalism.” Colin Rowe, writing with a different focus but within the same intellectual milieu, equated programming with “naive scientism” and worried that architects would be sidetracked by an “orgy of
expensive but impeccable interdisciplinary collaboration” and end up perpetually “waiting for printout.”

If Eisenman and Rowe represent the common opinion of Theory circa 1980, the famous debate in those years between Alexander and Eisenman can be seen as the moment when Theory “won.” While Alexander’s buildings might have been pleasant and functional, they were also, as Eisenman argued, boring. And worse, they seemed to be part of an insidious program of social engineering. The buildings advocated by Eisenman, on the other hand, aimed to provoke inhabitants to question their own deeply held values. In the following decades, this sort of architecture (call it “critical architecture”) became associated with Theory, at least in the East Coast North American academic context that played such a large role in setting the intellectual standards and agendas during this period.

Summerson’s argument in favor of programming had been tried, tested, and repudiated.

However true the foregoing historical sketch may be, it certainly must be qualified. The Theory that repudiated programming is embodied, for example, in the essays collected in K. Michael Hays’s Architecture Theory since 1968. This theoretical lineage runs through Frankfurt School critical theory, and as such it views programming (in both the architectural and computer science senses) as instrumental and normalizing. Theorists in this vein typically suspect that the rationalism required to conform thought to the dictates of computer logic or functional requirements is akin to totalitarianism. In fact, they seem to have regarded a particular habit of thought as the true enemy. When the uncompromising Frankfurt School figure Theodor Adorno moved to Los Angeles, he saw authoritarian thinking everywhere. From the perspective of North America, consumer culture and technocracy were certainly forces to be reckoned with. Hence the urgent need for opposition, particularly from the intellectual bastions of the East Coast – and particularly via “autonomous” activities such as art and the formalist architecture that aligned itself with art.

Suspicion regarding programming still lingers today. We should note, however, that Alexander and other proponents of programming were not naive functionaries. To the contrary, many were very theoretically minded. This chapter begins with the premise that it is worth questioning the distinction whereby Theory upholds standards of critique while (minor, ad hoc) theories are tied down by the contingencies of practice. This concerns another détente of the Cold War era. It was precisely through the combative, oppositional mode of Eisenman and his cohort that all sorts of minor theories were occluded behind the Iron Curtain of Theory.

An expanded history of theory should begin by noting that programming and its associated theories did not disappear even after they were excluded from Theory. A thriving subculture of programming congealed around computers – in technical classes and computer centers at architecture schools, in computer groups
CHAPTER 4. PROGRAMMING

at corporate firms, and in a global network of nomadic computer consultants. The tradition of programming in architecture continued as a subdiscipline through the 1970s and ‘80s, finally rejoining the mainstream with the digital architecture of the 1990s. Once the distinction between Theory and theories is set aside, we can begin to locate the situated knowledge that has been obscured and to think about how to reincorporate it into the history of theory. (This applies not only to the subculture of programming. Hays has suggested that the 1960s saw the end of the illusion of a single shared “architecture culture,” and that the 1970s were the beginning of an era of fragmentary, insular subcultures. Thus there are likely many subcultures and situated theories waiting to be rediscovered.)

With digital culture becoming second nature across the discipline, now is a particularly good time to uncover and reconsider the theories that accompanied its emergence. A postulate of cultural techniques may help us to identify these situated theories: namely, that practices precede concepts. The usual example is that people made marks of one sort or another long before the concept of “writing” solidified. Likewise, assemblages of processors and input and output devices were used before “the interactive computer” or “computer-aided design” or “digital modelling” became stable concepts. If theories are situated in contexts of practice, cultural techniques stand out as figures of knowledge that mark the former.

This line of reasoning poses problems, however. Does every instance of practice engender its own concepts? If so, then what is the basis for believing that a concept is shared by more than one person? These are particularly pressing questions in the case of early electronic computers because it is not obvious how or why computer-related concepts would have spread. Architects conducted many isolated and ephemeral “experiments” using computers before the 1990s, but few recognizable buildings were produced. Computers themselves did not move at all: until the 1980s, computers were big, immobile boxes to which few people had access. Computers in the 1960s and 1970s are a straightforward case of what Bruno Latour calls “centers of calculation.” Following Latour’s argument, we can imagine that early computers gained their power through the diffusion of “immutable mobiles” – that is, words and images on paper that circulate in lieu of computers themselves. The hypothesis of this chapter is that evidence for shared computer-related concepts in architecture can be found in printouts.

As a methodological point, the approach of cultural techniques assumes that theory can exist in a situation even if it is not explicitly articulated. This raises the question of what exactly a theory is. My working definition is rather broad: theory is about understanding. Newton’s theory of gravitation allows us to understand the motion of planets and why things fall to earth. Theories of programming allow us to understand architecture in certain ways. A theory works this way even if it is never written down or explicitly articulated.
Cultural techniques thus play a mediating role between printouts and theories. Traditionally, aesthetics have often been called upon to play a similar mediating role – to translate subjective judgments of singular things into the shared values of a “community of taste.” Commenting on Immanuel Kant’s aesthetic philosophy and updating it for the mid-twentieth century, Hannah Arendt described the specifically political role of aesthetics. Aesthetic judgments, Arendt says, share with political opinions that they are persuasive; the judging person—as Kant says quite beautifully—can only “woo the consent of everyone else” in the hope of coming to an agreement with him eventually. [...] Culture and politics, then, belong together because it is not knowledge or truth which is at stake, but rather judgment and decision, the judicious exchange of opinion about the sphere of public life and the common world, and the decision [regarding] what manner of action is to be taken in it, as well as to how it is to look henceforth, what kind of things are to appear in it.

Expanding on Arendt’s insight, we could add that some aesthetic/political judgments are also decisions about which theories are allowed to appear in shared life. Though I will need to substantiate this in the following pages, I suggest at the outset that printouts can be productively analyzed in terms of aesthetic schema (which I take to be equivalent to figures of knowledge) on the way to making judgments about them. I will return to the important role of aesthetics for the history of theory in the conclusion.

The following cases are presented as a guidebook to some of the aesthetic categories that can be found in the subculture of architect-programmers in the 1960s. There are several excellent guidebooks on the topic of older printing methods (woodcuts, engravings, etc.); the one by William Ivins that has been around since 1943 is especially perceptive at providing hints about how image types and techniques are related to historical figures of knowledge. Such guidebooks are lacking for computer-generated printouts. I will illustrate the five printing methods Smith identifies, grouped into three sections: X Y Plotter and Drafting Machine output, Standard Printer and Modified Printer output, and CRT output. To connect this with what was going on in architecture, I will correlate Smith’s examples with images created at a computer center that operated at Harvard’s Graduate School of Design. The Laboratory for Computer Graphics and Spatial Analysis (LCGSA), as it was called, is best known for the mapping software it created, but it also developed dozens of smaller programs covering many aspects of design in the 1960s and 1970s. Researchers at the LCGSA took advantage of its location in an architecture
school to collaborate with architects (including Charles Correa and The Architects Collaborative [TAC]), to work across media (from color prints to kinetic art), and to experiment with the wide array of output devices available in the period. The LCGSA’s projects are collected in a binder called the *Red Book*; it is from here that I will pull images.

**X Y plotter and drafting machine output**

If there has been a single “great device” of graphic output for architecture, it is the plotter. Plotters in the 1960s produced not pixels, as they do today, but lines: the pen moved in the X direction while the paper moved in the Y direction. Rather than recreating a one-to-one matrix of pixels, plotters operated through a language of instructions for creating vectors. As Claus Pias so eloquently puts it, “computer graphics of this kind might therefore be described as choreography, as the notation of movements to be performed.”

In the mid-1960s there were, in fact, two choreographic languages for plotters: one for common plotters and another for what were called drafting machines. For Smith, what set these devices apart was their precision. Common plotters were less precise because they could only understand instructions for creating straight lines. Drafting machines, on the other hand, could create complex curves.

These devices understood different sets of instructions because they were used quite differently – though neither was originally used by architects. Plotters were inexpensive, everyday devices for engineers and scientists, but they were still beyond the means of most architects. In any case, plotters lacked the graphic refinement an architect could easily produce by hand (Fig. 1). The common plotter was a device of expediency for those who could afford the luxury and lacked the manual skill.

The drafting machine was a more refined device, but it too did not suit the needs of architects. Drafting machines could accommodate several pens and different line weights and they could draw complex curves, but they lacked a software environment that could handle architectural notation. They operated on a low-level language derived from descriptive geometry and tailored to aerospace fabrication.

The development of the language of drafting machines was a pivotal moment in the history of computation. In the postwar period, a torrent of funding for defense-related research washed over American academia, with the Massachusetts Institute of Technology (MIT) among the largest recipients. A series of grants at MIT to develop manufacturing techniques combined with research surrounding electronic computers, resulting in the Automatic Programmed Tool language (APT). APT quickly became the preferred language for controlling fabrication
machines, which originally were run directly from punched-tape programs. The CAD Project of 1959 aimed to bypass this step and to “provide an efficient mechanism for going almost directly from the requirement for a machined part to the finished product.” One result of the CAD Project was Sketchpad, which became the first proof of the concept of the general-purpose interactive computer. At Sketchpad’s conceptual core was APT, the language of drafting machines.
Fig. 2. Perspective Simulation of Development in the Landscape (Spring 1968), from Selected Projects (Courtesy of the Frances Loeb Library, Harvard University Graduate School of Design)

Fig. 3. Page from "SOM’s Computer Approach," Architectural Record (Mid-August 1980)

One Magnificent Mile comprises three hexagonal concrete tubes bundled together for structural stability. Using the same computer graphics system as for the other studies shown here, the structural engineers were able to display the deflected shape (exaggerated) of the building under wind loading.

The drawing was created utilizing SOM’s Structural Data Management System (SDMS) that was initiated 10 years ago. With the system, a data base is developed that describes the structure, its elements, and the loadings placed upon them. A standard language was developed to describe structures of all kinds regardless of the analysis techniques or programs to be employed. The language can deal with all sorts of "goofy" geometries. It has a library of standard elements, configurations, properties of sections, load distributions, etc. The SDMS can take a list of all the members that have loads on them and turn it into input for STRUDL, STRESS or other analysis programs. The engineer never has to deal with understanding the specific formats for all the different analysis programs. SDMS produces input that is sent to whatever computer happens to run that program.
I tell this well-known story as a reminder that CAD drawings in the 1960s should not be thought of as bare-bones architectural drawings, though that might be what they look like. Rather, they were something like performances of sets of instructions for a manufacturing process. A drafting machine and its drawings would have had an aura similar to that of a supercollider or a space shuttle: they were feats of big engineering used by large, technically sophisticated institutions.

Many of the drawings in the Harvard lab archive were produced by common plotters. They generally have only one line-weight and are made up of a series of line segments – no curves (Fig. 2). Some show evidence of device modification. One drawing creates a Moiré effect though the superimposition of lines made with pens of different widths. Another was plotted with several colors of ink. There is a story about a programmer at the LCGSA who used a plotter to draw dots rather than lines; his idea had to be scrapped, however, because it replaced the usual soothing *whoosh* rhythm of the plotter with a jarring *Thunk Thunk Thunk Thunk* soundtrack as it slammed the pen into the paper hundreds of times.

One LCGSA researcher notes that “watching a plotter was an impressive experience,” like watching a superhuman draftsman. Plotter drawings in the 1960s pointed towards a techno-social future of human-computer symbiosis in which “the computer will [...] aid man in the creative process, making it possible for him to generate wealth with very little labor and emancipating him for activities that are commensurate with his humanity and his spirit.”

We should emphasize that plotter drawings were usually unnecessary from a practical point of view; generally speaking they could just as well have been drawn by conventional means. Plotters were nevertheless used by architects because, as one researcher noted, they added a “commercial value in marketing the product.” The LCGSA came out with a series of books to convince corporate managers that they could use computer graphics to sell their computational expertise to clients. The public relations value of plotter drawings held for decades. One key moment was a 1980 article in *Architectural Record* that featured two-color plotter drawings by Skidmore, Owings & Merrill (SOM) (Fig. 3). What was most impressive was not the drawings themselves but the fact that SOM had the corporate wherewithal to orchestrate their design process around the computer. Some have argued that seeing images in specialized ways and asking others to see them in the same way is a foundation of professional expertise; others have argued that the postwar period was an era in which ideals of “trained judgment” held sway in many disciplines. The plotter drawings of SOM and Harvard’s LCGSA therefore constructed and publicized the expertise and judgment of the architect-programmer.

In these plotter drawings we see a distinct figure of knowledge emerging: the wireframe. Wireframe drawings present a visual logic stripped of superficial details. As with Iakov Chernikhov’s drawings half a century earlier, they seek maximum
spatial-analytical punch with minimal graphic means. The jarring colors suggest a lack of sophistry and an attunement to modern, mechanical functionality. We are allowed to see through buildings with a penetrating intelligence, all the way to their underlying structures and forces. I want to emphasize that these are aesthetic effects: wireframe images seem rational, but they present rationality without explanation. The viewer is left with the feeling of rationality.

The concept most closely associated with the wireframe aesthetic is the canny combination of deep-structural rationality with public relations value: the idea that it is an image of architecture that is most important, but rather than an image of a building, what is offered is an image that gestures towards a building’s structural rationality and performance. The wireframe aesthetic is a distinct aesthetic category or schema and is a key figure of knowledge used by those who approached architecture as programming.

**CRT output**

If the plotter was important in the 1960s and '70s and remains so today, there is another device that was just as ubiquitous but that has since disappeared. This device is the microfilm plotter. Microfilm plotters came in a variety of forms, but all were variations on the theme of a camera mounted to a screen (Fig. 4). As the
phosphor coating on the cathode ray tube lit up, this light would be captured on film. Smith notes that “the primary characteristic of the Cathode Ray Tube as a graphic device is speed. […] Drawings that might take 15 minutes or an hour on a drafting machine or x y plotter can be produced in seconds on a CRT.”

The first microfilm plotters were developed alongside the first CRT screens. They were very expensive and were found only in large research centers. As CRTs and microfilm plotters became less rare in the late 1960s, the conventions of CRT drawings followed two paths. One direction was towards drawings that looked similar to plotter drawings. Several of the examples given by Smith, for example, could have been produced on a drafting machine. One example, however, gestures in the other direction (Fig. 5). This is a geographic drawing of some kind, with a circle around it, which directs our attention towards the round screen from which it came (most screens in the '60s were round rather than rectangular). I will focus

Fig. 5. Cathode ray tube output reproduced in Christopher P. Smith, “Graphic Data Processing” (1964)
on this second set of conventions – conventions that emphasize that CRT drawings are images from a screen. Using today’s terminology, I will call these “screenshots.”

Only two images in the LCGSA’s Red Book clearly came from CRTs. Both have white lines on black backgrounds, an inversion of the pen-on-paper look common to architectural drafting (Fig. 6). The color scheme of light lines on a black background is one convention of 1960s screenshots. Others include: showing incomplete or partial views, which emphasizes that the computer screen offers a framed view of a virtual object with a reality beyond any particular representation; showing examples of what software can do rather than a single definitive image of a project; implying that what is shown in the image involves computation in some way, often by including unnecessary annotations; and a look that is, by the standards of other media, unpolished and without the normal niceties of visual communication. These conventions add up to an image type that represents “the interactive computer” and the process of using one.\textsuperscript{50} In an era when most computer use involved punch cards and printouts – with no screen and no interaction – screenshots conjured an unusual situation.

The shift towards interaction and simulation opened new possibilities for many disciplines, including architecture.\textsuperscript{51} Donald Greenberg, a faculty member at Cornell and pioneer in computer graphics, included dozens of screenshots in a 1977 article in Architectural Record.\textsuperscript{52} He created the first “flythrough” (of a new I. M. Pei building) and produced colorful photorealistic renderings (of Le Corbusier’s Ronchamp) – techniques that fit nicely with the reigning architectural
Imagine a PC CAD system which allows you to design in three dimensions—and then revise that design as often as you want.

Imagine, too, the ability to see a design in plan, elevation, section or perspective; all at the stroke of a key.

Imagine the ability to communicate to and from other DXF compatible PC CAD systems.

Think of creating a design that provides the production drawings, model design and presentation images, all from one model.

Think about “walking around” a shaded model in real time to show all sides of the project.

Think Solid Vision. The newest member of CalComp’s CAD Continuum of PC-based software through turnkey systems.

For more information or a demonstration, call your local dealer. Or contact us at CalComp, P.O. Box 3250, Anaheim, CA 92803; 1-800-CALCOMP.

Circle 22 on information card

Fig. 7. Advertisement from Architecture (February 1987)
phomenology of the period.\textsuperscript{53} By 1987, an issue of *Architecture* focusing on computers was chock full of advertisements featuring screenshots of new CAD software (Fig. 7).\textsuperscript{54} Screenshots serve as a reminder that computers are not reducible to “tools” or “electronic brains,” which were persistent tropes already in the 1960s.\textsuperscript{55} The computer came to be seen also as a window into a simulated world and a means by which to manipulate the latter as one would a physical model.

Some of the conventions of 1960s CRT drawings have been carried forward into contemporary screenshots. We now typically bypass the camera and create bitmap images with the press of a button, but the vestigial menu bars and default colors of screenshots still represent the idea that something is native to the computer. Just as the snapshot aesthetic of the 1950s captured everyday urban life,\textsuperscript{56} the screenshot aesthetic conveys the supposed authenticity of the digital environment at the center of contemporary architectural production.\textsuperscript{57}

The main theoretical postulate involved in the screenshot aesthetic is that finished buildings are less important than the methods and technologies of architectural production. The idea is that architecture resides not in well-composed physical objects but in the processes that precede them, which can only be grasped and manipulated through the computer screen. The screenshot aesthetic draws the viewer to empathetically imagine using an interactive computer.

**Standard printer and modified printer output**

Although screenshots are among the easiest images to produce with a computer today, they were rare in the 1960s. In terms of popular appeal, the 1960s equivalent of the screenshot was output from what Smith calls a standard printer. Also known as chain printers, these devices operated much like typewriters: through the impact of ink and metal on paper. They were fast and cheap, and they were the only output device every computer center could be expected to have.\textsuperscript{36}

With a little imagination and programming, a chain of standard characters can produce a drawing. The graphic coherence of standard printer drawings seems to congeal despite their flimsy material support (Fig. 8). The flowcharts and graphs that Smith gives as examples use elements outside the normal grammar of graphic design. Repurposed characters fall on a regular grid, forming into lines and shapes seemingly against their own will. The effect is of a shimmering field with figures just barely coming into focus.

After standard printers, Smith describes modified printers, which add special characters for business, engineering, or scientific graphics that allow relatively sophisticated drawings of molecules, circuit diagrams, and the like. Modified printers supplied the look of the made-in-house business and scientific graphics of the 1960s.
Fig. 8. Standard printer output reproduced in Christopher P. Smith, "Graphic Data Processing" (1964)
In its desire for popular relevance, Harvard’s lab developed its own form of modification for standard printers. Rather than requiring a chain of special characters, programmers used a trick that allowed characters to be printed over the top of one another. A period or a zero would make a light spot; several characters – O X A V – printed on top of one another would make a dark spot (Fig. 9). By obliterating recognizable characters, this technique brought printers closer charcoal on paper – the impressionistic realm of tone. Drawings could now be produced through pixels, long before raster monitors or the concept of the bitmap were common. A great deal of the LCGSA’s theoretical and programming activity went into defining and smoothing tone-based boundaries – much like the bitmap filters that would later help make Photoshop so popular. While plotters required programmers to think in terms of a choreography of lines, standard printers asked them to think in terms of an even field of data.

Programmers at the LCGSA incessantly explored the visual potential of their printers. They wrote routines to create drop shadows, for example, and ran their paper through their printers multiple times with different colored ribbons (Fig. 10). As much as any other image from this period, these drawings convey the
atmosphere of 1960s hacker culture: extracting the furthest range of surprising possibilities from the limited equipment available. These printouts also certainly take part in the information aesthetic—“the ‘pleasure’ of thinking” and “the excitement of discovering new rules, laws, and limitations”\(^6\) — that drove much experimentation in computer art in this period.\(^6\) LCGSA researchers looked for patterns in the built environment by obsessively studying the figures created by overlapping layers of information.\(^6\) When Colin Rowe later poked fun at “the enthusiasts for data collection,” it was certainly people like the researchers at the LCGSA that he had in mind.\(^6\)

Fig. 10. Shadow (1971) from Selected Projects (Courtesy of the Frances Loeb Library, Harvard University Graduate School of Design)
CHAPTER 4. PROGRAMMING

The theory involved in standard printer drawings could be characterized as positivist: the information aesthetic certainly depends on belief in the value of empiricism, that knowledge is cumulative, and that societies and environments are undergirded by patterns and laws. Time spent obsessively computing the connections between activities and rooms in a suburban house, for example, contributed a small but solid brick to the edifice of human knowledge.\(^6^4\) It is also important, however, to understand that practitioners of the information aesthetic were typically driven to put such knowledge in hands of “the people.” In other words, the work of Harvard’s lab fit the technocratic ethos of the 1960s, but it was also a forerunner of the countercultural ideals of the personal computer era.\(^6^5\) To understand the information aesthetic, it is important that these contradictory ideas be kept in tension. We might add, more generally, that theories often contain such internal contradictions, and that aesthetic categories sometimes stand as figures that mark these irresolutions.

Conclusion

According to Smith, the methods listed above exhaust the computer output methods available in 1964. Smith’s list is confirmed by the LCGSA archive: all the images in the *Red Book* were originally output by devices that fall into his categories.\(^6^6\) They thus appear to provide a reasonably accurate window onto the 1960s subculture of architect-programmers.

Some printouts that originated in the LCGSA circulated outside of this subculture, largely in the hands of a new class of professionals in the world of architecture: computer consultants. Allen Bernholtz and Eric Teicholz are representative figures here.\(^6^7\) Both were trained as architects, and both spent time as researchers in the LCGSA during its early years. Bernholtz joined the lab shortly after it was founded in 1965. He worked on a series of architectural projects with Marshall McLuhan, SOM, Perkins and Will, and several other smaller firms, and ended up employed by Canada’s Ministry of State for Urban Affairs in 1972. Teicholz likewise worked on a string of projects with Ivan Sutherland, Charles Correa, TAC, and other smaller firms in Boston, and developed a suite of architectural software that he sold to Digital Equipment Corporation before moving into the world of facilities management. During their stints at the LCGSA, Bernholtz and Teicholz wrote essays for journals such as *Design Quarterly* and *Architectural Forum*, presented at trade conferences (for both architecture and computation), and met one-on-one with countless clients and colleagues. If programming was their common mode of practice, printouts were the objects they placed at the center of attention. Indeed, Bernholtz opined that computer programming would make architectural programming into “an explicit exercise where we could sit down and point to these differences and
MATTHEW ALLEN

have the designers discuss it, argue it, and eventually come to some compromise regarding the building program. He argued for a new subdiscipline of programming within architecture, calling himself a “programmatic engineer” — a job he described as “some mix of architect, engineer, [and] computer-psychologist type.” This professional ideal was developed in a shared subculture and became visible to wider audiences via printouts.

Once aesthetic techniques begin to circulate outside the context in which they were first developed, they tend to be used with little regard for the ideals of their original context. Historians, who are inclined to be aware of the many ways that theories tag along with practices, are often called upon to remind architects of the contexts they may have forgotten. I hope that the foregoing historical sketch suggests just how difficult this sometimes is. If every situation of practice — every subculture — has its own concerns and theories, historians have a lot of recontextualization work to do. Compounding the difficulty, historical theories often conflict with the theories and convictions held by historians themselves. If we do not subscribe to a particular theory, we might not find it worth our time to examine it. If we are committed to critical theory, we might have trouble seeing the value of “instrumental” theories and elaborating them on their own terms.

As I suggested above, aesthetics offers a method by which to approach such forgotten, questionable, subcultural theories. Between a singular History of Theory and isolated microhistories of theories, aesthetic categories stand as figures of knowledge that allow us to map the local situations of theories in terms of the larger cultural territory they share.

This approach requires us to think about aesthetics in a slightly unusual way, however. In the quotation above, Arendt agrees with Kant that aesthetic judgment is the moment when taste becomes public. The problem is that Arendt assumes that it makes sense to talk about “the public,” “universal” judgment, and a singular culture. Kant and Arendt (and aesthetic philosophers generally) are biased towards the universal. For Arendt, Greece and Rome stand in for all human values. She worries about how the singular culture she evidently loves (so-called Western culture) appears to be in the process of dissolution under the pressure of mass culture (which, she says, is not culture at all but “entertainment” — what culture becomes when it is instrumentalized).

It is helpful to situate Arendt in the mid-century discourse of “the crisis of man,” a term I borrow from Mark Greif. Writing on the heels of several decades of catastrophe, critics such as Arendt tended to think in terms of enormous existential conundrums. “Mankind” and “culture” were universal values to be defended, and they was often seen in terms of sweeping binary paradigms: culture versus civilization (Paul Ricœur and Kenneth Frampton), autonomy versus instrumentality (Adorno and Eisenman), or even, most strangely, the Japanese high cultural “snob”
versus the American subcultural “animal” (Alexandre Kojève). Such antinomies could only work for polemic and caricature. Last century’s age of crisis continues to resonate today (Frampton, for one, has repeatedly updated Arendt’s paradigm), but it is showing signs of age. We have now passed through a several-decades-long celebration of multiculturalism, and cultural constructionism has become the lingua franca of the humanities. Champions of a singular, universal “culture” are looking more and more like relics of the Cold War.

The Kantian aesthetics we have inherited from figures such as Arendt aim towards the universal, but such an aim is not necessary. Why not update aesthetics? Can we imagine aesthetic judgments to be shared but not universal?

One direction would be to see aesthetic schemas as analogous to emotions, with all their variety and culturally situated complexity. William Reddy describes emotions as “loosely connected thought material” that we come to name and categorize through a long process of enculturation. In other words, the connotations of and boundaries between emotional schemas depend on our upbringing in a specific culture. Once such a system of schema is in place, it provides templates or shorthands for rapid response to situations encountered in everyday experience. I suggest that aesthetic categories should also be understood as loosely connected thought material that are put into schemas that we share with the people around us and use to make quick judgments.

Once we see aesthetic judgments as not universal but subcultural, we can also see a finer grain of aesthetics categories. Arendt, like Kant, generally limits her discussion to beauty. Kant sometimes also writes about the sublime; Rozenkratz added ugliness to the repertoire. Writing about the world of postindustrial labor, Sianne Ngai adds the cute, the zany, and the interesting. In a significant update to aesthetic theory, Ngai argues that

our aesthetic experience is always mediated by a finite if constantly rotating repertoire of aesthetic categories […], which are by definition conceptual as well as affective and tied to historically specific forms of communication and collective life.

As an example of what this expanded cast of aesthetic categories might look like, we could turn to Benjamin Buchloh’s analysis of the fine-tuned aesthetic work done by conceptual artists in the postwar period. In order of appearance, he lists:

- the aesthetic of administration
- the aesthetic of the speech act
- the aesthetic of linguistic convention and legalistic arrangements
- the aesthetic of the handcrafted original
the aesthetic of administrative and legal organization and institutional validation
the aesthetic of contemplative experience
the aesthetic of mural painting
the aesthetic of Conceptual Art
the aesthetic of permutation
the aesthetic of the studio
the aesthetic of production and consumption
the aesthetic of declaration and intention
the aesthetic of the newly established power of administration
the aesthetic of anonymity

The foregoing analysis suggests a specific role for aesthetics in the history of theory:
to describe concerns and concepts in a way that is schematic and shared but not universal. Rather than reflect on (universal) culture, we could investigate situated processes of cultural production. Confronted with practices and images that seem to be attached to some theoretical content within a subculture, we could begin by identifying a set of aesthetic categories that appear to be at work. We could then elaborate the “thought material” with which they are associated. In this way, aesthetic categories can be taken as figures of knowledge that mark situated theories.

Looking at the printouts that emerged from the subculture of architect-programmers in the 1960s, we can see evidence of theoretical investment that goes well beyond the trope of computers as mere tools or calculators. Other concepts present themselves. Wireframes point towards theories of the deep structure of architecture and the expertise required to control it. Screenshots are evidence of theories of interactivity and the idea that finished buildings are less important than the processes of architectural production. Screenshots came along with techniques of simulation that matched contemporaneous architectural phenomenology. The clunky pixelizations of chain printer drawings were developed alongside theories of information and visualization and countercultural ideals of hacking. They habituated architects to thinking about drawings as fields of data. New aesthetic categories encapsulated these situated theories, and printouts were the medium by which they spread.

Returning to the observations with which I began, I will reiterate that situated theories and Theory share a common denominator: they are about understanding. One big difference has to do with the attitudes that characterize them: localized, operative theories are usually characterized as “positive” while Theory is “critical.” Rather than linking positivity with positivism and scientism, I suggest linking it with desire. The situated theories I outlined above do not begin with a hermeneutics of suspicion; unlike the model offered by the Eisenman/Alexander debate,
they do not imagine theory as a battlefield. Why not take a cue from aesthetics and approach theory – initially, at least – in terms of pleasure and appreciation? Historians of theory could begin by grappling with situated desires to understand worlds of practice on their own terms and save judgment for theorists of theory.

Notes

1. I would like to thank Hilde Heynen and Sebastiaan Loosen for their insightful comments on my manuscript and Laura Frahm for her advice and encouragement on this project from the beginning.
8. The other field where “Theory” means the same thing is literary studies. See, e.g., “Theory: Death is not the End,” *n+1* 2 (Winter 2005).
14. An extreme version can be seen by bringing together Hannah Arendt’s “Crisis in Culture” and *Eichmann in Jerusalem*: her arguments set up a certain affinity between the philistine who approaches art in terms of use value and Adolf Eichmann, the Nazi officer who perpetrated genocide under the guise of bureaucratic logistics. (Arendt was not part of the Frankfurt School, though she shared many concerns and was in the same generation as, e.g., Adorno.) Hannah Arendt, *Between Past and Future: Six Exercises in Political Thought* (New York: Viking, 1961) and *Eichmann in Jerusalem: A Report on the Banality of Evil* (New York: Viking, 1963).
16. How exactly this happened is part of a longer story in which – to point out just two factors – corporate modernism and “popular” postmodernism were construed as theoretically shallow and curricula in architecture schools segregated technical and practical courses at a significant distance from courses in history and theory.
21. Michael Mahoney argues that the various devices that have been lumped into the category of “computers” should be disaggregated and approached through the histories of the groups that used them. Michael Mahoney, “The Histories of Computing(s),” *Interdisciplinary Science Reviews* 30, no. 2 (June 2005).
23. More specifically, for the purposes of this paper, I subscribe to Michael Friedman’s rather minimalist “global view of scientific understanding,” in which a theory is a concise, singular mental construct that stands for or “explains” many other mental constructs (observations). In this view, the reduction from multiple unexplained phenomena to coherent figures of knowledge is the hallmark of theory. Michael Friedman, “Explanation and Scientific Understanding,” in *Theories of Explanation*, ed. Joseph Pitt (Oxford: Oxford University Press, 1988).
24. In the following pages I will not fully elaborate the theories I discuss. Rather, I will provide reference to primary sources that expand the theories in further detail and secondary sources that explicate their historical context.


28. The closest are books such as John Lewell’s A Z Guide to Computer Graphics (New York: Mcgraw-Hill, 1985) and Martin Jurgens’s The Digital Print: Identification and Preservation (Los Angeles: Getty Conservation Institute, 2009). What is missing in both are concise accounts of why particular visual techniques might be chosen: what are their connotations and affordances? Generally, it is the field of visual studies that is pursuing this sort of question most vigorously; see, e.g., James Elkins, Visual Practices across the University (Paderborn: Wilhelm Fink, 2007).


30. Though most readers are probably familiar with printers and plotters, CRTs are decidedly obsolete. CRTs or “cathode ray tubes” were the dominant screen technology until the advent of flat-panel screens (LCDs and LED displays).


35. Advertisements in trade magazines in the 1960s and 1970s show that there was a market for innovative techniques to save drafting time using preprinted elements (often furniture and annotations).


39. Chrisman, Charting the Unknown, 142.

40. Ibid., 79.


43. As I will describe in a moment, most of the LCGSA’s early software was designed to use inexpensive chain printers for output, which, the researcher said, “as a research and learning tool […] provided everything I needed to know and was more efficient both time- and money-wise than plotter output.” Allen Bernholtz, “Spatial Allocation in Design and Planning,” *Proceedings of the 9th Design Automation Workshop* (1972).


49. Zabet Patterson has written a book about one of the first microfilm plotters, the S-C 4020, which was used by scientists and artists (and the distinction between the two, she notes, is often difficult to make) at Bells Labs in the 1960s. Some of the first computer art was made using this particular device. Zabet Patterson, *Peripheral Vision: Bell Labs, the S-C 4020, and the Origins of Computer Art* (Cambridge, MA: MIT Press, 2015).


61. Higgens and Kahn, Mainframe Experimentalism.


69. Ibid.


