THE EXECUTED AND THE OBSERVED IN SKETCHES: VISUAL AND COMPUTATIONAL PROCESSING FOR EXPLORATIVE DRAWINGS

Onur Y. GÜN
Massachusetts Institute of Technology, Cambridge, MA, United States
oyucegun@mit.edu

Abstract. Drawing is expressing. The mind’s eye works with the drawing to materialize ideas via transforming them into visual abstractions. The genuine supremacy of drawing emerges from its potential to evoke, not from its ability to represent. Computers are harbingers of unprecedented and enriching drawing environments. Yet they also introduce ambivalences because they suppress drafter’s bodily and perceptual engagement with drawings. This paper aims to delineate the similarities and differences between hand drawing and (via-computer) algorithmic drawing for design. The goal is to discuss the altering role of eyes and hands in long-contrasted virtual and material environments of drawing. The outlined comparisons of algorithmic and hand sketching should encourage research for blending digital and analogue modes of sketching.

Keywords. Drawing; computation; sketch; algorithmic; design; studio; visual; calculation.

1. Story of “Two Squares”, Four and a Half Centuries Apart

The oldest preserved architectural drawings date back to the Gothic Era. Most of the drawings are generated for representative reasons, and it is hard to determine whether they were produced prior to or after construction. These drawings are not to scale, they don’t feature shade and shadow, and a sense of depth is hardly perceivable. Gothic drawings highlight the vertical linearity, and are flat in nature (Branner, 1997). In some drawings buildings almost disappear under the dominantly drawn ornamental elements. However, a limited number of instructional set of drawings such as the ones included in Roriczer’s *Booklet Concerning Pinnacle Correctitude*, represent different characteristics.

In the second half of the 15th century, Mathes Roriczer (aka Matthäus Roritzer), a German master builder of cathedrals, prepared a design booklet to elaborate on what he called the “proper way of designing and building pinnacles” for Gothic
churches (Shelby, 1977). He claimed his book would help builders grasp the “art
[architecture] they are dealing with” to overcome the flaws and imperfections of
improper geometric constructions.

Throughout Roriczer’s instructions for erecting a pinnacle from a base plan
(Figure 1), lines are used for two different purposes: at the beginning they are uti-
lized as guides for the descriptive geometry, such as the “base square” or the
“centre line.” Later they depict the outline of the (pre-conceived) pinnacle form.
Regardless, Roriczer instructs the observers about what to see in his drawings.
Lines and shapes, for him, are immutable entities, and their purpose is to convey
a specific form: “then when you eliminate the remaining lines that are not needed
for the setting out, there remains such as form as shown below.”

The visual similarity between George Stiny’s two squares (Figure 2) and
Roriczer’s pinnacle base is striking. What is more striking is the fundamental
difference between the ways these two authors look at and see things. They both
draw, they both look at what they draw, and although the shapes appear to be the
same, they see different things. Roriczer’s shape is a static entity that is not open

Figure 1. Roriczer’s instructional drawings for development of the pinnacle base.

Figure 2. George Stiny’s 2 Squares and their non-hierarchical network:
You can see almost anything.
to interpretation, so it cannot be changed. In contrast, Stiny (2011) looks into the two squares and sees four triangles, pentagons, K’s or k’s, or a network of non-hierarchical shapes.

The goal here is not to compare and contrast the two sets of drawings that were produced roughly four and a half centuries apart; it is rather to emphasize the configuring power of the observer’s eye in defining or deciding what a shape actually means.

2. Putting Parts Together: Hand Sketching vs. Algorithmic Sketching

While designing, the components of drawing fall into place to reveal design intentions and forms. Sketching comes into play, as the way to quickly explore, if not to document ideas. It usually employs traces that are deprived of precise geometric constructions. For R. Evans (2000): “…the sketch is without obvious geometry.”

Drawing is not instant. It happens over time, and it is a product of aggregation, accumulation and omission of traces. The bits one draws merge, dissolve, overlap towards something greater than themselves. Renaissance artists are known for cultivating their drawings this way (Brothers, 2008) (Figure 3). Not the pre-conceived idea, but the visual clues discovered in the course of drawing embody the outcome. The parts drawn fuse together to evolve into a design-communication device: the drawing.

Visual clues can be explored computationally in the absence of the pen and paper. Once considered as a systematic whole of relational parts, a design could be explored via algorithms (Terzidis, 2006). Algorithmic thinking does not necessitate
use of computers; however, they can be considered as productive collaborators because of their high capability to handle complexity.

The algorithmic study *spherical networks* (Figure 4) can represent how nodes and lines could be used to produce conceptual drawings of network connectivity. The underlying GC-Script algorithm, developed in 2005, helps visualize the rule-based relational system of nodes (points) and connecting bridges (lines). Each node in the network connects up to a certain number of neighbouring points depending on a proximal threshold. User defined parameters, such as threshold distances, can be iterated to explore various visualizations. One can start developing a sense of potential formal composition of the system only after sequential executions of the algorithm.

Note that the density in the spherical networks model increases towards the centre of the configuration. This is a result of the shortening corresponding distances between the points distributed on the underlying circumscribed spheres in the system. The structural density of the core, for instance, could be reduced either by decreasing the number of points per subsequent sphere surface towards the centre or by modifying the rules of proximity-dependent connectivity. In each case, the model is re-constructed from scratch.

In a way, the process becomes some sort of sketching action. An abstract formal idea, once translated into an algorithmic system, can produce sketches for visual evaluation. However, these translated algorithmic sketches cannot be altered as rapidly as hand-sketches. For instance, would it be “fair” to break a pure algorithmic system to modify only one individual member out of, say, thousands? If we break all the system to modify that single unit, would the drawing become “truer” (Cook, 2008)? If one does not want to disturb the algorithmic mechanism of the drawing, one could insert new sets of rules into the algorithm: a rule could be employed to modify the system for the system to become flexible enough to accommodate one or several manual (by-hand) modifications. However, in such a scenario the role of the hand would significantly differ from its role in sketching.

Algorithmic constructs necessitate translation of conceptual ideas into rule based systems via syntax, and thus an algorithmically created sketch appears not
simultaneously with, but following mental processes. In hand drawing it is hard to define how thoughts, transformations and traces occur. Both hand and (via-computer) algorithmic sketching could be accepted as feedback mechanisms; however, the differences exceed similarities.

3. What do the Computational Tools for Design do?

3.1. BINARY SYSTEMS, BRUTE FORCE AND DESIGN

Any computer application, due to the binary nature of the machine language, operates via discretization of the world. One has to subdivide things into tiny bits and re-compose (assemble) them using configuration methods to generate designs. However, design concepts that relate to aesthetics cannot necessarily be directly translated into computational means (Fishwick, 2006). Computational tools developed for designers enable computation of Vitruvius’ *commodity* and *firmness* concepts more than the *delight* one.

Parametric systems assist the designer in defining explorative design domains with numeric or geometric dependencies and constraints. Parametric models require a linear bottom-up modelling process and this very nature makes them incompatible for immediate inclusion or exclusion of (geometric) parts. Moreover, because of their deterministic logic, parametric systems produce the same result when same the sets of parameters are used, in contrast to real-life dynamics.

Unlike parametric systems, systems incorporating stochastic searches (such as genetic algorithms) can utilize probability and grant designers a more open-ended design process. However, the valid designs are still determined via constraints within the definition of the problem (Pottmann and Bentley, 2007): the solution to the design problem is somehow already bounded in the domain defined by the pre-conceived fitness criterion.

Computational simulations help analyse probable real-life performances of designs. Although the architectural design industry uses and benefits from such tools, these tools also produce invalid results, and thus the data they produce requires insightful examination for their proper use. Non-expert users might sometimes forget that a simulation incorporates the reconstruction of conditions of reality, not the reality itself.

All the methods covered here benefit from the number-crunching ability of the computers and they mostly suppress the potential contribution of seeing.

3.2. VISUAL ENGAGEMENT AND CODE

Computers enable us to work with constructs of higher complexities which arguably cannot be laid out precisely with pen and paper. The precision required
to work with computers can only be achieved after long skill-building sessions. Yet even after, the unforgiving nature of computers remains there to keep designers’ ambivalent feeling towards them fresh. However, this unforgiving nature might also produce (accidental) unexpected outcomes: optimistically, one may discover something new to look at and see about what one is designing. The time invested to build an algorithmic drawing system is counter balanced with the gain of variety of the visualizations that could be extracted later on.

Ironically, such unexpected discoveries do not lie embedded in the parametric relationships of the computational system. These discoveries rather come into existence as perceptual products of vision. Eye works with the drawing the way it would work in a hand drawing: it looks and sees something beyond the computer’s constructed vision. Arguably, for this very reason, regardless of the advancements in computing, designers keep working with both digital and analogue mediums. Designers mix and match techniques such as digital printing, hand drawing and scanning. Switching between these modes is an effort to translate the products between the two not-so-incompatible digital and analogue worlds.

Maeda’s work represents this bi-modal work process. His algorithmic constructs are products of visual materials explored by the eye and the hand (Figure 5). The code comes into play to outline and then algorithmically refine the image to perfection; technology and art become counterparts (Maeda, 2000).

4. Teaching How to Design and Draw Computationally

Today, students can be accepted as the largest group of early adopters of any computational design tool. Design studios incorporating new techniques and
understandings provide environments to explore unprecedented design forms, and to experiment with new materials and digital prototyping techniques. However, conventional techniques, even at the times the studios prefer them to diminish, keep influencing the processes.

Architecture students have to discover ways of playing with geometry and make it their own to develop confidence for design exploration. Computers should be introduced as tools for them with which to build and look at things. A top-down teaching method, feeding the students with certain forms and modelling techniques, would only make them followers of the precedents. Likewise, not supplying enough material to play with would be discouraging. Arguably, what the students need is a safe ground on which they can feel confident enough to discover their own design gestures and their potential of seeing while utilizing computers for design exploration.

*Basic Design Studio* at the Istanbul Bilgi University, Department of Architecture was constituted around such an idea. The studio aimed to discover and teach ways of generative design thinking via *algorithmic constructions* and *visual evaluations*. Transformational patterning exercises proved satisfactory for these two core concentrations.

Students did an empirical exercise by drawing closed polygons and generating arrays of them via applying planar transformations. They were free to draw any closed polygon, and they could apply any type of planar transformation (translation, rotation and scaling). Three operations of polar or rectangular arraying helped them finalize their patterns. At the end of the exercise they had to discover newly emerging polygonal figures and classify them as *main shape(s)*, *secondary shape(s)*, *transition shape(s)* and *connector shape(s)* (Figure 6).

![Figure 6. First year Basic Design studio: Student Beril Binoğul's pattern exercises and her 3D maze.](image-url)
The emerging pattern, after their subsequent modifications, became the medium for students to look into and evaluate. They made their own discoveries through drawing, seeing, evaluating and interpreting. They modified the sequences of operations they picked to manipulate the resulting shapes they produced. These sequences were represented with an algorithm (computational calculation) and the products were evaluated visually (visual calculation).

Students had both computational and perceptual exercises to attribute meanings, functions, and even personalites to the figures. Such attributions helped them define and design unique parts to be composed into their own designs. The final step of the exercise was formulated around an imaginary creature made of water of a certain mass. The aim was to construct sequential voids to accommodate this creature while it navigated through the voids of a 3D maze as it changed physical phases to fit in, flow or evaporate (Figure 6): “…design a 3D maze in which your water creature will cycle through, in and out. Let your creature witness different conditions as it passes through different voids, find entries and exits throughout its life of cycles.”

Practiced by students and extended to more sophisticated assignments, similar techniques enable students to work with forms and configurations of much higher complexities both in 2D and 3D exercises. In the following example, students sequentially transform polygonal units to build an intricate structurally sound adaptive bridge (Figure 7).

![Figure 7. First year undergraduate student Ulufer Çelik’s sequential transformations applied to a polygonal shape (left) andYağız Özkan’s “wall to hollow” bridge (right).](image)

5. Contributions

5.1. ALTERNATIVE WAYS FOR VISUAL AND COMPUTATIONAL PROCESSING OF HAND-DRAWINGS

This paper is about the eye and its potential to discover hints of new designs within drawings. Computation, without a doubt introduces novel methods, mediums and
means to explore (with) design drawings. Yet the impossibility of fathoming the emerging is what is valuable in sketching. Zero dimensional calculation (Stiny, 2006), however, as opposed to the on-going branding about emergence in computational design, cannot fully support the emerging. In a sense, algorithmic sketches remain intangible and impermeable. In other words, an algorithm can produce sketches, yet a potential shift in intention requires a structural modification and re-execution of the code. The code may appear as a hindrance to the immediacy of the sketch. A pencil stroke proves more efficient and tactile, albeit less precise.

The outlined comparisons of algorithmic and hand sketching here do not aim to praise one of these drawing modes above the other. Rather, they aim to explore how these modes could be brought together. These comparisons should be used as a motive to search for alternative methods of sketching that could incorporate feedback mechanisms between hand, eye and the computer.

5.2. WORK IN PROGRESS

Research about incorporation of computational methodologies into drawing processes have been developing around gestural interaction and shape recognition, and are definitely re-shaping the drawing process, yet the investment in the eye and vision might potentially introduce alternative ways for novel discoveries. A hypothetical workflow that incorporates a computational camera that looks at one’s drawing process to produce auxiliary imagery for enriching the visual clues (Figure 8) provokes an alternative way to draw with computation. A hybrid drawing model that

Figure 8. Eye-Camera coupling for hard-drawing. A computational camera can record the development of drawing, extract embedded but not seen drawings and apply computation to produce auxiliary drawings during the process of drawing.
merges digital and analogue methodologies, acknowledging the limitations and potentials of both will prove more tangible, accessible and enriching.

References

Shelby, L. R.: 1977, Gothic design techniques: the fifteenth-century design booklets of Mathes Roriczer and Hanns Schmuttermayer, Southern Illinois University, Carbondale.