The Mental Canvas: A Tool for Conceptual Architectural Design and Analysis

Julie Dorsey Songhua Xu Gabe Smedresman Holly Rushmeier Leonard McMillan[§]

Department of Computer Science
Yale University

[§]Department of Computer Science
The University of North Carolina at Chapel Hill

Abstract

We describe a computer graphics system that supports conceptual architectural design and analysis. We use as a starting point the traditional sketchbook drawings that architects use to experiment with various views, sections, and details. Rather than interpret or infer 3D structure from drawings, our system is designed to allow the designer to organize concept drawings in 3D, and gradually fuse a series of possibly geometrically-inconsistent sketches into a set of 3D strokes. Our system uses strokes and planar "canvases" as basic primitives; the basic mode of input is traditional 2D drawing. We introduce methods for the user to control stroke visibility and transfer strokes between canvases. We also introduce methods for the user to position and orient the canvases that have infinite extent. We demonstrate the use of the system to analyze existing structures and conceive new designs.

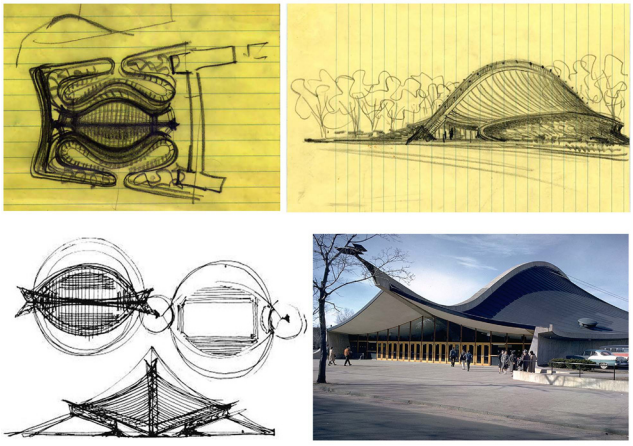


Figure 1. Original sketches for the Yale Hockey Rink (built 1956–58) by architect Eero Saarinen, and a photograph of the completed building. (From the Yale University Archives and Manuscripts.)

1 Introduction

Architects use drawing as an aid to visual thinking in analysis and design. Drawings at various levels of detail allow the designer to both work through how an existing structure fits together, and to collect and refine ideas for new buildings. Existing CAD systems have been extraordinarily successful in the late stages of building design and construction. However, because they require the specification of geometry with great accuracy, they have not proven suitable for analysis and conceptual design. In this paper we present a system that uses the idea of a 3D sketch as the foundation for a representation of a building. Our system closely follows the traditional 2D drawing process in analysis and design, and adds capabilities for the user to fuse the drawn elements into a 3D structure.

In architectural design, the processes of analysis of existing structures and design of new structures are closely linked, as an architect synthesizes experience, knowledge of built work, and other source material in order to imagine new buildings. We take particular interest in the role of analytical drawings in this design process. Analytical drawings use existing structures as their subjects but, unlike documentary photographs or representative sketches, the point of analytical drawings is to understand a building. The sketches prepared in this type of analysis are often a starting point for new designs.

A classic example of a design that would not naturally be roughed out in a CAD-like system, and an inspiration for this project, is the Ingalls Arena at Yale. Early sketches of the arena, by architect Eero Saarinen, and the building, as it looks today, are shown in Fig. 1. In conceptual drawings, ideas are not yet fully formed, and shapes are loosely defined. The goal of this system is to provide the freedom to experiment with similar non-traditional and complex forms.

0-7695-3009-5/07 $25.00 © 2007 IEEE
DOI 10.1109/PG.2007.64

This requires a different approach focusing on how a user can organize ideas in 3D, rather than on many current efforts in sketching research that attempt to interpret simple sketches as 3D shape.

Typically sketches are 2D representations of a 3D idea, but there is no defined "middle ground" or information that "goes between" the sketch and the object. It is often difficult to interpret a 2D sketch whose 3D implications are not clear, and the existing array of computational aids offers little assistance in this task. The incompleteness of a sketch often makes it impossible to resolve ambiguities without further input from the user, and it is even difficult to offer an intuitive way for the user to provide that information. For the most part, these difficulties have been passed over by 3D modeling and visualization programs that instead choose to address the needs of those users who already have a well-defined 3D object in mind.

In our proposed approach, the user begins with a drawing in 2D, without any initial specification of positions or views, or any automatic inference of perspective or primitive shapes. We allow the user to then position these drawings in 3D space, as one would tack up paper sketches on a bulletin board. The user can begin fusing and considering views together by transferring individual drawn strokes onto new planes other than the one on which they were drawn. We assist in this process by giving the user controls over stroke visibility from different views, and giving convenient controls over plane positioning and orientation. The user gradually builds and refines a set of 2D planes, containing strokes, to form a 3D sketch space.

The rest of the paper is organized as follows. We begin with a review of relevant previous work. We then describe the basic features of our proposed system. As a demonstration of the system functions, we show the development of several 3D stroke assemblies. We then go on to demonstrate the primary uses of the system: analysis of existing structures and developing new designs. We conclude with ideas for future directions in this area, based on the experience of architectural students using the system.

2 Previous work

The earliest computer-based sketching system dates back to the pioneering contribution by Sutherland [18], which is widely considered as the first complete graphical user interface. His system, equipped with a light pen and a plotter display, allowed sketching of 2D technical illustrations for design applications. Sachs et al. first introduced 3D drawing to the community of computer graphics [14] in 1991. Research on sketching in computer graphics has since grown to include a variety of topics, including achieving the "look" of traditional sketches [6, 15, 16, 22], sketching abstract concepts [23], and 2D tools for artistic expression. Here, we focus on previous work specifically related to 3D design.

Our work is related to prior efforts in digital perspective sketching. Piccolotto [13] introduced perspective sketching on a tilting pen-based table display as an electronic tool for early architectural design and for detecting geometric forms in the input. Cohen et al. [4] explored the interesting idea of 3D curve sketching, which requires the user to specify the image-space projection of the curve and its shadow on a horizontal surface. In the 3D6B editor proposed by Kallio [10], 2D input strokes are projected to the grid surface chosen by the user. However, no post-creation transformation or alteration of strokes is allowed, and a great amount of time is required to produce a 3D sketch. In comparison, our system offers a gradient from 2D to 3D design work: the user chooses to transform the initial 2D sketch drawings to the corresponding 3D forms at any time. In addition, the stroke editing functions supported by our system provide the flexibility to gradually refine the sketches and resolve ambiguities as needed. This matches the cognitive process of conceptual design, as an architect moves from preliminary ideas toward precisely defined models.

Our work is also related to early work in gestural interfaces for modeling, like the SKETCH system [23] and Teddy [8]. SKETCH creates geometry by mapping gestural input to modeling functions. Teddy infers freeform 3D polygonal surfaces from drawn input. Igarashi and Hughes [7] proposed an interface for 3D drawing based on a set of parallel working suggestion engines. A related idea was explored by Tsang et al. [21], who wrote a system to accept input and to suggest similar geometries from a database. In their system, 2D images were introduced as a guide for the sketching process, which can attract, smooth, and resample input curves. This system mainly reuses existing shapes, either captured in the form of images or contained in the geometric database. In our stroke-based system, no geometry is introduced that the user has not explicitly specified.

Tolba et al. introduced a system [19, 20], where user entered strokes are represented as projective strokes: stroke points are projected onto a unit sphere surrounding the viewer. This leads to the capability of reprojecting the strokes to simulate camera motion, thus making the changing of viewpoint possible. As a result, panning, tilting and zooming are effectively supported even though no true 3D geometry model is ever constructed. Our new system moves beyond strokes on the unit sphere to placing strokes freely in 3D space.

There is also previous work in using non-conventional

primitives like strokes or planes as a lightweight modeling framework. Cohen et al. [3] proposed a system called *Harold*, which creates a visually rich virtual world made of strokes using an extended billboard technique. This system is not concerned with the iterative refinement of ambiguous sketches for capturing and stimulating design ideas in the early stages. Ijiri et. al. employed a collection of 2D strokes, which are used as a guide for the 3D modeling of flowers [9]. Bourguignon et al. introduced a system where strokes are visible from one direction and gradually disappear as the camera rotates [2]. Decoret et al. [5] successfully simplified complicated geometric models into a set of representative planes, which they called *billboard clouds*. Their work is inspired by the prior practices of scene modeling through textured clusters [11] and layered depth images [17]. Unlike our work, these efforts focused on efficient scene modeling and rendering, but didn't emphasize the support for a fluid scene design and refinement process.

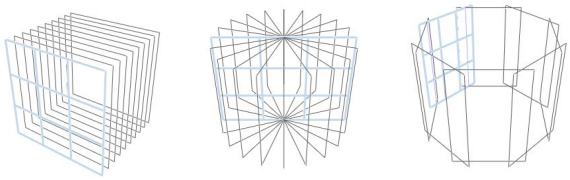
The core idea driving the development of our system is to allow a user to develop coherency in 3D gradually. Theoretically the same process could be carried out in an existing CAD system. However CAD systems are designed to create well-defined 3D models, and then automatically render views of them, rather than letting the user draw a view from scratch. The design software Autodesk AliasStudio tried to bridge this difference. However in the AliasStudio, sketches serve only as transparent overlays to guide 3D model creation by traditional means. Our work focuses on allowing the user to draw 3D shapes as an assembly of gestural strokes, and to easily draw, edit, and refine those strokes.

3 System Description

A 3D sketch in our system is built out of invisible planar 2D surfaces, which we call *canvases*. These canvases can be positioned and rotated in 3D space, using the traditional CAD tools of position, rotate, and scale, on three coordinate axes, using the tablet pen. Though they are theoretically infinite in extent, they are drawn with finite borders so that the user can easily see their orientation onscreen. Only one canvas can be active at any time. A canvas is made active by clicking on the corner of its onscreen border.

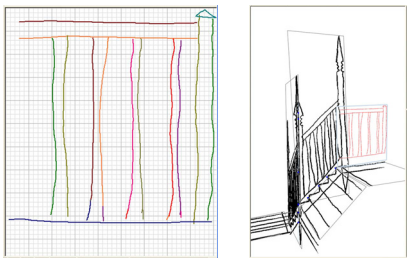
The user draws with the tablet pen, and sets down strokes directly on the active canvas. By rotating the camera view with the arrow keys on the keyboard, the user can draw on a canvas from any angle; the pen position is automatically projected onto the appropriate point on the canvas surface.

For quickly exploring new ideas or changes, a *view canvas* can be created. View canvases act like acetate transparencies locked to the camera's near plane, and provide functionality for quick sketching in 2D, over a view of a 3D object, without defining a volume or worrying about depth. View canvases are bookmarked to a certain view of the 3D object and, once properly oriented, are not meant to be realigned.



(a) (b) (c)

Figure 2. Pre-configured canvas assemblies provided by our system: parallel (a), co-axial (b) and ring (c). The blue planes indicate the currently active canvas.



(a) (b)

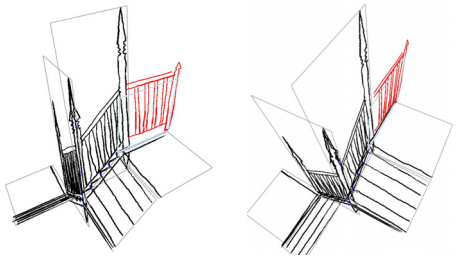
Figure 3. The user can draw strokes in an orthographic 2D window and the system will map the drawing onto the selected canvas.

Strokes can be pushed from a view canvas onto a 3D canvas using *perspective projection*. The user enters *selection mode*, draws a box around the strokes to be pushed, then selects a target canvas. These strokes are then pushed onto the target canvas in such a way that they appear no different from the view canvas viewpoint, but lie flat on the target canvas surface. The results are exactly the same as they would be if the user had drawn the strokes directly on the target canvas.

3.1 Core Features

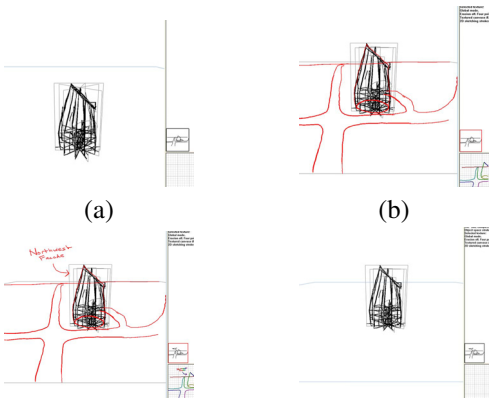
Our system is based on managing canvases, entering strokes and manipulating existing strokes. Each of these operations required the creation of new user interaction concepts.

Managing canvases. To help the user get started quickly, the system provides a few built-in 3D assemblies of canvases in common arrangements: axial cross-sections, paral-



(a) (b)

Figure 4. The user can also draw directly on a canvas through the 3D viewport (a), and the pen position will be automatically projected onto the canvas surface. If the canvas is rotated or moved afterwards, the strokes will remain fixed to their relative positions on the canvas (b).



(a) (b) (c) (d)

Figure 5. Using a view canvas. Figure (a) illustrates a 3D scene. By clicking on the canvas icon on the right, the user is brought into the bookmarked view, with the associated annotations (b). Annotations can be added (c), but as soon as the user leaves that view, they disappear (d).

lel stacks, and a circumferential ring. (See Fig. 2.) These are meant as initial layouts; users can manipulate the position and rotation of each canvas with a full suite of tools very similar to traditional CAD programs. For example, the user can move canvases along their local coordinate axes, optionally using a single-axis constraint along any of the three coordinate axes. To rotate an individual canvas, the user uses the up/down/left/right arrow keys.

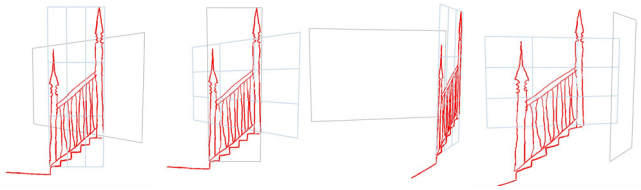
A problem in managing large numbers of overlapping 3D canvases on a 2D screen is selecting the canvas that should be currently active. Conventional drawing systems use a system of cycling through all possible 3D objects corresponding to a 2D screen location clicked by the user. This would be impractical and frustrating in our system. Instead, we use the novel technique of selecting an active canvas by clicking near one of the corners of the rectangular icon that represents the position and orientation of the infinite canvas.

Entering strokes. The user can draw strokes in three ways: on a canvas through a gridded 2D interface (Fig. 3), on a canvas in 3D (Fig. 4), and on a *view canvas* in 2D (Fig. 5). The 2D interface is best suited for orthogonal drawing or tracing images: the user draws lines in a gridded window, and strokes appear in the corresponding point on the 3D canvas. Alternatively, the user can draw directly on the 3D canvas, and the transformation to 2D is computed internally. The user can use existing strokes as context, and use his natural sense of perspective to draw how he thinks the object should look. These lines exist in 3D and, once drawn, can be rotated and repositioned relative to the other canvases. If strokes are drawn outside the displayed boundaries of a canvas, the canvas icon can be expanded to include the new strokes.

Thirdly, the user may draw strokes on a view canvas. The view canvas is a novel representation that allows the user to orient a drawing with respect to a scene without designating a specific 3D position for the canvas. The user can create a view canvas from any viewpoint in the scene; from that point on, it acts as a bookmark to that specific view. Once a view canvas is created, the user can draw freely on the screen, and the strokes will be drawn directly on the camera's near plane. They are tied to this particular bookmarked view, relative to the rest of the scene.

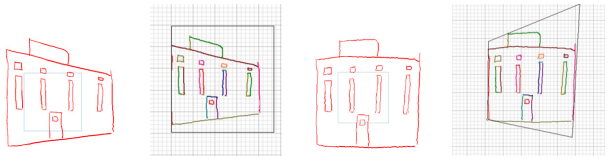
When the user wants to transfer strokes from a view canvas into 3D, the perspective projection command pushes these 2D strokes into the 3D scene (Fig. 6). The user selects strokes on the view canvas and a target canvas in 3D space. The system then pushes these strokes backwards, as if through the screen, until they hit the target canvas. From the position of the view canvas, the pushed strokes will look exactly the same, but these strokes can now be moved and rotated in space. Strokes can also be pushed from one 3D canvas to another, in which case they will appear the same from the point of the view at the time of the push.

Manipulating strokes. A novel feature of our system is a method for modulating the transparency of lines. The user sees this as erasing or adjusting strokes in a particular drawing, rather than as managing a computer graphics system setting. For some purposes, it may be desirable to see all of the strokes. However, when viewing a complex form, seeing every stroke can sometimes cause visual confusion. To address this, strokes can be assigned varying opacity depending on the angle between the parent canvas and the camera: strokes on a canvas that faces the camera are drawn fully opaque, and strokes facing the sides are nearly transparent. The level of transparency is interpolated using the cosine law. Canvases can optionally be made one-sided,



(a) (b) (c) (d)

Figure 6. The user can move strokes from one canvas onto another through perspective projection. The before (a) and after (b) images from the current camera viewpoint will appear unchanged. However, when the process is observed from the side, the change in stroke position from before (c) to after (d) the operation is clearly seen.



(a) (b) (c) (d)

Figure 7. After strokes are moved, the user may adjust them by dragging the corner points of the bounding box, and the drawing will be deformed in a free-form manner via bilinear interpolation. A sketch drawing on a plane in 3D and its orthographic view are shown before the deformation in (a) and (b), and after deformation (c) and (d).

in which case associated strokes are invisible when viewed from the reverse side.

The user can also can render portions of a canvas opaque using an *occlusion map*, a 2D binary texture map with two possible states for each pixel: transparent or opaque. The user paints on the occlusion map with *brush* and *flood-fill* tools, where the tablet pen position is projected onto the occlusion map just as it would be projected onto a canvas to draw normal strokes. For instance, the user could draw four normal strokes to indicate a square, and then paint the interior of these four strokes opaque on the occlusion map. The opaque regions act like solid white regions, and, as they are rendered after any strokes underneath them, they occlude strokes that would normally appear behind them, as can be seen by comparing Fig. 10 (d) and (e). This functionality becomes important for more detailed arrangements of solid objects, where occlusion is necessary for visual clarity. An occlusion map will never hide strokes that are coplanar with it, only those that lie behind it. The occlusion map expands dynamically to fit the user's input, and texture map resolution is scaled adaptively for performance.

Unlike computer vision systems, we do not require the user to strictly adhere to proper vanishing points. Although accurate input may lead to more realistic results, all strokes are treated the same. However, for those who desire it, the system does offer a freeform deformation tool to assist in properly aligning drawings. The functionality works similarly to the similarly-named operation in Adobe Photoshop: a rectangular box is drawn around the region to be deformed, and the user can independently drag four control points at the corner of the rectangle. The strokes inside the box are deformed to fix the distorted quadrilateral, as shown in Fig. 7.

3.2 Additional Capabilities

Our system has many other capabilities that are not novel, but are included to augment the basic functionality.

Group-based operations. Canvases can be grouped by dragging a selection box across the screen. Individual canvases can be added or removed from the group with the mouse. Once grouped, all canvases are transformed together, and their relative positions are preserved.

Rendering options. Our system offers several rendering options. The user can adjust a global width adjustment to make the entire drawing relatively thicker or thinner. All strokes not on the active canvas can optionally be drawn faintly. Another option allows strokes to get thinner as they recede in the distance, providing a spatial cue for complex models.

Inspired by [12], we provide a flexible texture-based rendering technique to imitate artistic media such as charcoal, pencil, and watercolor. These styles visually reinforce the preliminary and loose nature of the strokes in our system.

Tracing over reference images. Our system allows users to load an existing drawing or a photograph onto a view canvas or a normal canvas in 3D and draw strokes by tracing.

Import and export. All the sketch drawings can be individually exported into a back-end sketch database for reuse in other projects. Importing strokes and canvases from an existing project into a current project is also supported.

3.3 Implementation

Our system runs on an a Windows desktop workstation with a Wacom pressure sensitive input tablet. There are three main parts of the system: an input module, which reads signals from peripheral devices; an operation module, which performs all the user operations including stroke and canvas manipulation; and a display module, which is responsible for visual feedback. The system is implemented in Visual C++ 6.0. The interface uses MFC and displays using

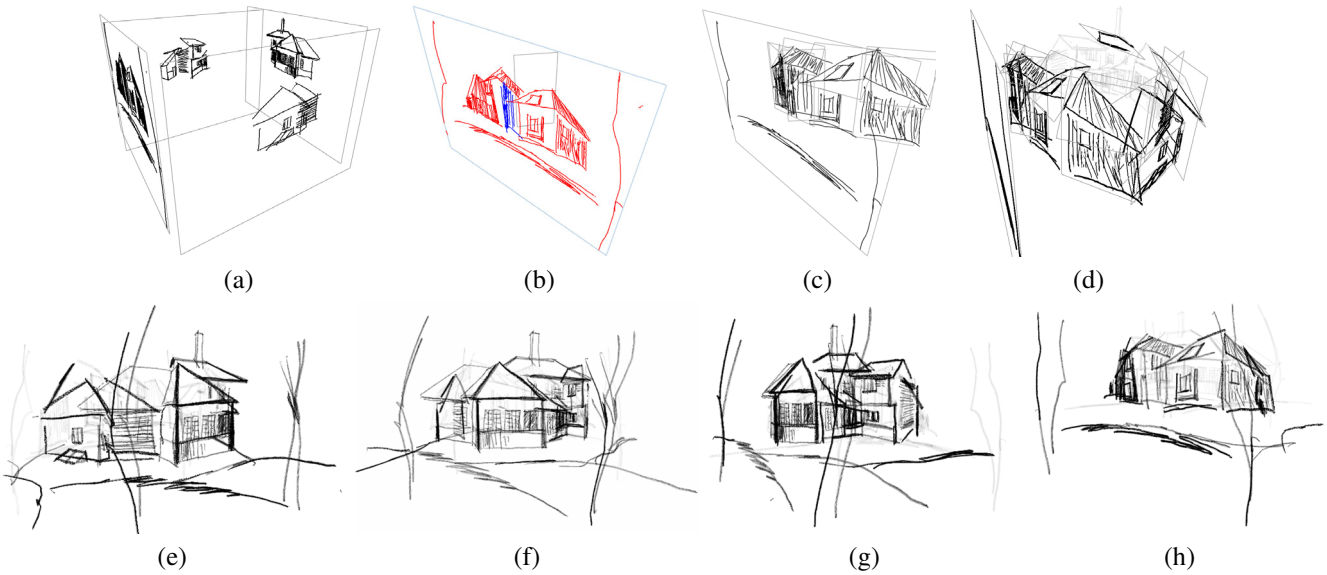


Figure 8. A surburban home. The designer begins with four sketches that he positions in 3D to approximate their orientation with respect to the structure (a). (b) and (c) show the before and after of the perspective stroke push in one sketch. After all four sketch drawings are processed in this way (d), we can view the house from novel views; the designer adds some landscape elements to the presentation at this stage (e)–(h).

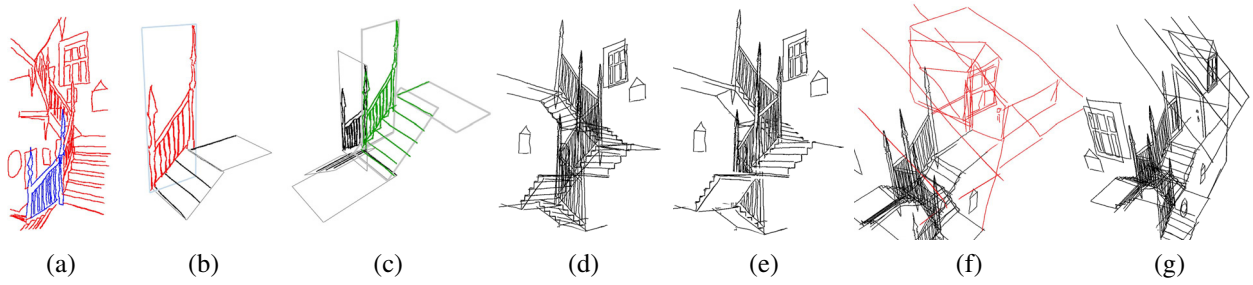


Figure 9. Modeling a stair. Beginning with a single drawing (a), the user selects and projects some strokes into 3D. He also adds a few more strokes using (b) as a contextual guide. Next, he clones a group of canvases together with the strokes in the group and positions the cloned group appropriately in 3D (c). After all the strokes in (a) are pushed, we can view the model from a different view with an optional without (d) or with occlusion strokes (e). Next, the designer continues to add new strokes relative to the strokes already placed in 3D (f). The final result after pushing the newly added strokes into the space is shown in (g).

OpenGL. Strokes are captured from the pen system and represented internally as a list of 3D points with an accompanying list of pressure values if the tablet pen is used. Canvases are represented by their normals, and optional parameters, such as one-sidedness. For performance purposes, the list of 3D points for each stroke is stored internally with the transformation from canvas coordinates to world coordinates already applied.

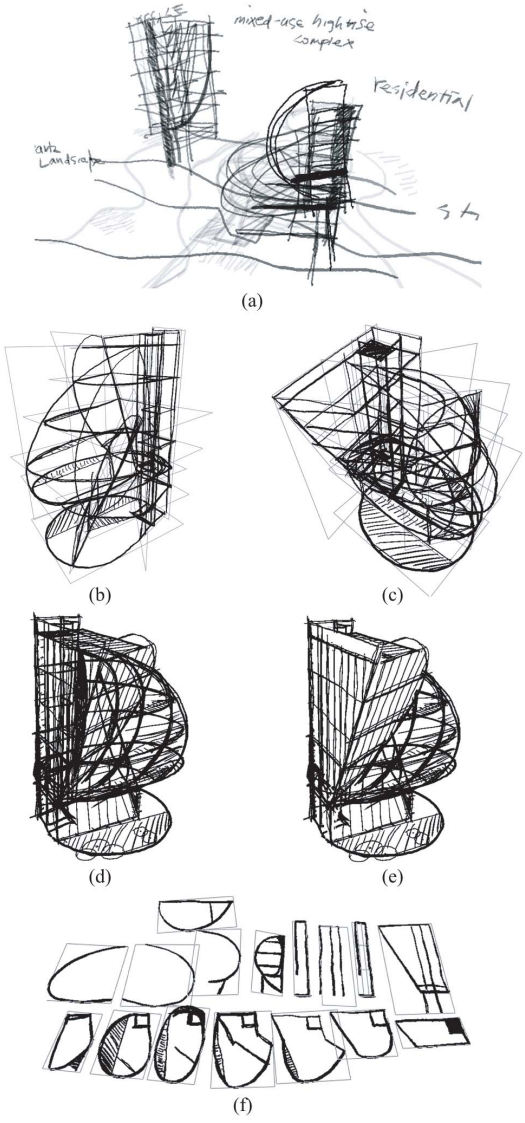


Figure 10. An original design using our system. (a) shows a preliminary sketch of the building and site. (b) and (c) show a detailed draft of the building. (d) is a final version, with occlusion maps in (e). (f) shows all the canvases in (c), laid flat.

4 User Experience and Results

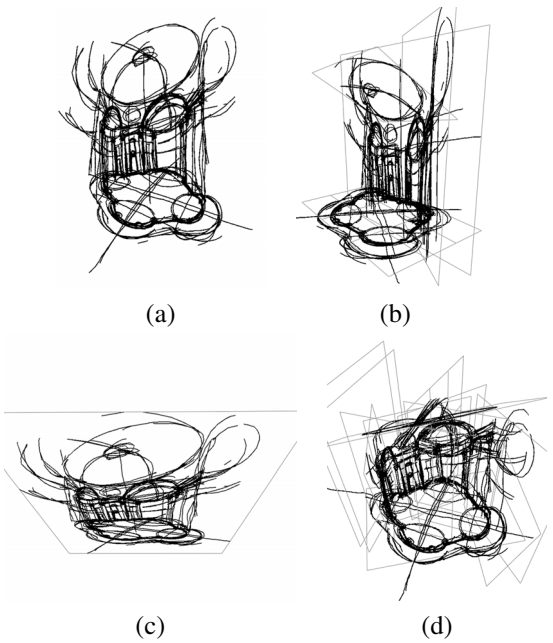


Figure 11. This series documents the conversion of a sketch on a single canvas into a 3D environment. (a) is the original sketch, and (b) shows the results after it this drawing was split up and projected onto several canvases. (c) is a top-down view of the original sketch, and (d) shows the same top-down view of the final result environment.

All of the examples in this paper were drawn by practicing architects and architectural students. The examples were developed after short training sessions. The system was modified to correct bugs and make minor changes in system behavior as the result of interaction with the users.

All of the users were familiar with 3D modeling packages commonly used in architecture. Initially, all users had difficulty breaking out of the mindset of *either* drawing in 2D, *or* using traditional CAD workflow. The most effective way for users to learn to gradually move between 2D and 3D was to start with a given set of 2D views as a first experience. An example that we found successful in introducing the concept is shown in Figure 8. A set of sketches were prepared by drawing over four photographs of a suburban house. Figure 8a shows an initial assembly of canvases. Figures 8b-d show the processes of pushing strokes from one of the views onto other canvases. Repeating the process creates a collage-like assembly viewable from any angle. This example successfully introduced the idea to the user a small number of strokes originating on separate sketches can be modified to capture the look of a complex structure, without an underlying 3D model.

Figure 9 shows the modeling a stair and adding a new feature. This example introduces the user to creating a new design element, rather than just working with pre-existing sketches. Here, the user alternates between replicating and pushing strokes from an initial sketch and adding new strokes using the initial drawings as a contextual guide.

After becoming accustomed to the concept of the system, users were able to develop different types of models. Here we show one example of a new design, and one example of an analysis of an existing building.

Figure 10 shows the iterative development of a design for a mixed-use complex, conceived entirely in our system. An interesting use of the system we hadn't anticipated was that the user took advantage of the stroke input and infinite extent of the canvases to add text annotation to early sketches, as shown in Fig. 10(a). As shown by a comparison of (d) and (e), both the all-strokes-visible and stroke-occlusion features were used.

Architects often use line drawings to better understand a complex three-dimensional space. Many buildings are designed with carefully chosen proportions between their parts. A special type of architectural drawing, called an *analytical drawing*, is used to identify and understand these relationships. Figures 11-13 show an architectural analysis of San Carlo alle Quattro Fontane, a Roman baroque church designed by Borromini. These drawings show the spatial organization behind the church's ornate interior spaces. Figure 11 shows an analysis of the church's interior by means of a single perspective drawing that is then broken apart into logical collections of strokes that are pushed out into space.

Particularly notable is the organization of the courtyard as shown in Fig. 12. This three-dimensional sketch model, produced using the extracted grid as a guide, shows the prismatic volume defined by each vault. This partitioning of the courtyard is not at all obvious from an initial glance, but an analysis of the geometry of the space using the tools of drawing, we can learn about the processes behind its conception.

The system was used again for annotation to record the proportions of the floor plan. The analytical model synthesizes the analytical drawings into a 3D representation. Figure 13 studies the central interior space in the church. The rules governing this space are very similar to those behind the design of the courtyard. The curved architectural forms here are conveyed with only 26 canvases.

The users' experience with the system demonstrated that it can be used effectively as a design and analysis tool. Our initial implementation of the system contained many buttons and menu options for variations in drawing style and modes of navigation that were not used and that cluttered the interface. We are currently streamlining the interface to eliminate these unused features.

5 Discussion and Future Work

Our goal has been to develop a system for analysis and conceptual design that is compatible with the process of visual thinking using traditional pencil and paper. We start with the same traditional process, and add the ability to move strokes into 3D space to merge and refine the structures suggested in the 2D drawings. In so doing, we expand on how sketching is conceived by extending the process of getting from 2D to 3D.

Though many of the individual techniques we utilize in our system are not new, the system as a whole is different from any other computer-aided design system currently available. We have totally eliminated a reliance on precisely defined geometry from the process of conceptual design on the computer, where ideas are too loosely formed to be so specified. This represents a fundamental change in the underlying representation of a form, whereas existing sketch-based tools, in the context of 3D modeling, offer innovative interfaces wrapping a traditional model representation.

There are many results of this new representation that would benefit from exploration. Adjustments to a complex volume can be made by redrawing a few strokes: the human visual system fills in the details. Making these changes is an easy, lightweight, and freeform affair. Designers can draw and redraw lines without being bound by the constraints of a polygonal mesh or the inflexibility of a parametric pipeline. Our system allows easy iterative refinement throughout the development of an idea, without imposing geometric precision before the idea is ready for it.

Our approach suggests several interesting avenues for future work. A natural extension of the system would allow a designer to fully refine a set of drawings into a 3D model, perhaps exploring multiple alternatives simultaneously. As our system enables a designer to view many concept drawings in a common context, providing the means to manage and navigate through a large number of sketches would allow designers to more readily draw on a body of referential and analytical drawings. Last, storing and representing the temporal evolution of a design, and making different steps in the design accessible to the user in the future, would be an invaluable aid in the design of new structures.

In this work, we have used the architectural design process as a driving application. However, architecture has its own methods, and we don't expect that the system we have developed is directly applicable to other types of visual analysis and design. In general, visual analysis through inter-

active drawing is in a very early stage in graphics research. Much more work remains to be done to understand visual analysis across many creative fields to understand the role that graphics can play.

6 Acknowledgments

We would like to thank Phillip Isola for helpful discussions and Soo-Hyun Kim for experimenting with the system and developing the examples shown in Figures 8, 10, and 11-13. This system builds on preliminary ideas outlined in a PhD thesis authored by John Alex [1], which was supervised by the first author. This work was supported by NSF award CCF-0738472 and a grant from Google.

References

[1] J. P. Alex. *Hybrid Sketching: A New Middle Ground Between 2- and 3-D*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2005.

[2] D. Bourguignon, M.-P. Cani, and G. Drettakis. Drawing for illustration and annotation in 3D. In *Computer Graphics Forum (Proc. of Eurographics)*, volume 20, pages 114–122, 2001.

[3] J. M. Cohen, J. F. Hughes, and R. C. Zeleznik. Harold: a world made of drawings. In *Proc. of the symposium on non-photorealistic animation and rendering (NPAR)*, pages 83–90, 2000.

[4] J. M. Cohen, L. Markosian, R. C. Zeleznik, J. F. Hughes, and R. Barzel. An interface for sketching 3D curves. In *Proc. of the symposium on Interactive 3D graphics (SI3D)*, pages 17–21, 1999.

[5] X. Decoret, F. Durand, F. X. Sillion, and J. Dorsey. Billboard clouds for extreme model simplification. *ACM Trans. Graph.*, 22(3):689–696, 2003.

[6] A. Hertzmann. Tutorial: A survey of stroke-based rendering. *IEEE Comput. Graph. Appl.*, 23(4):70–81, 2003.

[7] T. Igarashi and J. F. Hughes. A suggestive interface for 3D drawing. In *Proc. of the symposium on user interface software and technology (UIST)*, pages 173–181, 2001.

[8] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: a sketching interface for 3D freeform design. In *SIGGRAPH '99*, pages 409–416, 1999.

[9] T. Ijiri, S. Owada, and T. Igarashi. Seamless integration of initial sketching and subsequent detail editing in flower modeling. In *Proc. of Eurographics*, pages 617–624, 2006.

[10] K. Kallio. 3D6B editor: projective 3D sketching with line-based rendering. *Proc. of Eurographics Workshop on Sketch-based Interfaces and Modeling*, pages 73–79, 2005.

[11] P. W. C. Maciel and P. Shirley. Visual navigation of large environments using textured clusters. In *Proc. of the Symposium on Interactive 3D graphics (SI3D)*, pages 95–103, 1995.

[12] J. D. Northrup and L. Markosian. Artistic silhouettes: a hybrid approach. In *Proc. of the symposium on Non-photorealistic animation and rendering (NPAR)*, pages 31–37, 2000.

[13] M. A. Piccolotto. Sketchpad+ architectural modeling through perspective sketching on a pen-based display. Master's thesis, Cornell University, 1998.

[14] E. Sachs, A. Roberts, and D. Stoops. 3-draw: A tool for designing 3D shapes. *IEEE Comput. Graph. Appl.*, 11(6):18–26, 1991.

[15] M. P. Salisbury, S. E. Anderson, R. Barzel, and D. H. Salesin. Interactive pen-and-ink illustration. In *SIGGRAPH '94*, pages 101–108, 1994.

[16] M. P. Salisbury, M. T. Wong, J. F. Hughes, and D. H. Salesin. Orientable textures for image-based pen-and-ink illustration. In *SIGGRAPH '97*, pages 401–406, 1997.

[17] J. Shade, S. Gortler, L. wei He, and R. Szeliski. Layered depth images. In *SIGGRAPH '98*, pages 231–242, 1998.

[18] I. E. Sutherland. *Sketchpad: A Man-Machine Graphical Communication System*. New York: Garland Publishers, 1980.

[19] O. Tolba, J. Dorsey, and L. McMillan. Sketching with projective 2D strokes. In *Proc. of the symposium on user interface software and technology (UIST)*, pages 149–157, 1999.

[20] O. Tolba, J. Dorsey, and L. McMillan. A projective drawing system. In *Proc. of Symposium on Interactive 3D graphics (SI3D)*, pages 25–34, 2001.

[21] S. Tsang, R. Balakrishnan, K. Singh, and A. Ranjan. A suggestive interface for image guided 3D sketching. In *Proc. of the SIGCHI conference on Human factors in computing systems (CHI)*, pages 591–598, 2004.

[22] G. Winkenbach and D. H. Salesin. Computer-generated pen-and-ink illustration. In *SIGGRAPH '94*, pages 91–100, 1994.

[23] R. C. Zeleznik, K. P. Herndon, and J. F. Hughes. Sketch: an interface for sketching 3D scenes. In *SIGGRAPH '96*, pages 163–170, 1996.

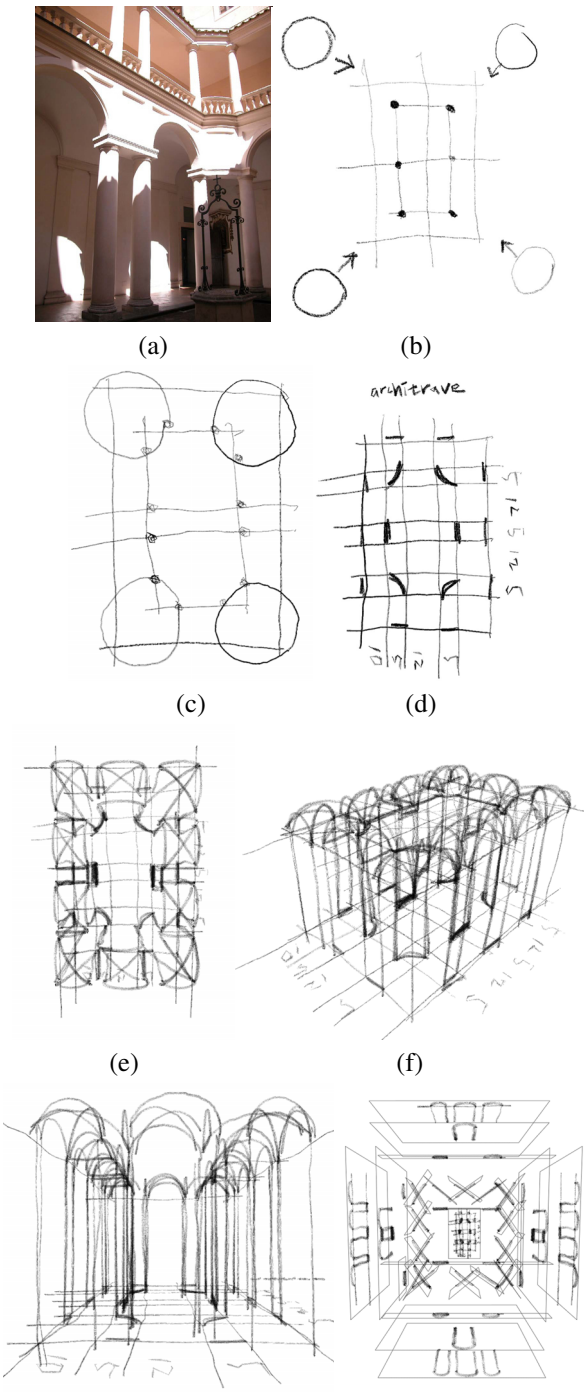


Figure 12. Analysis of the courtyard of the church *San Carlo Alle Quattro Fontane*: a photograph (a); (b) and (c) are analytical diagrams of the plan; (d) shows a side view; (e)–(g) are three additional views; (h) shows all of the constituent sketches employed in this analysis. Sketches are displayed here using a variety of stroke styles.

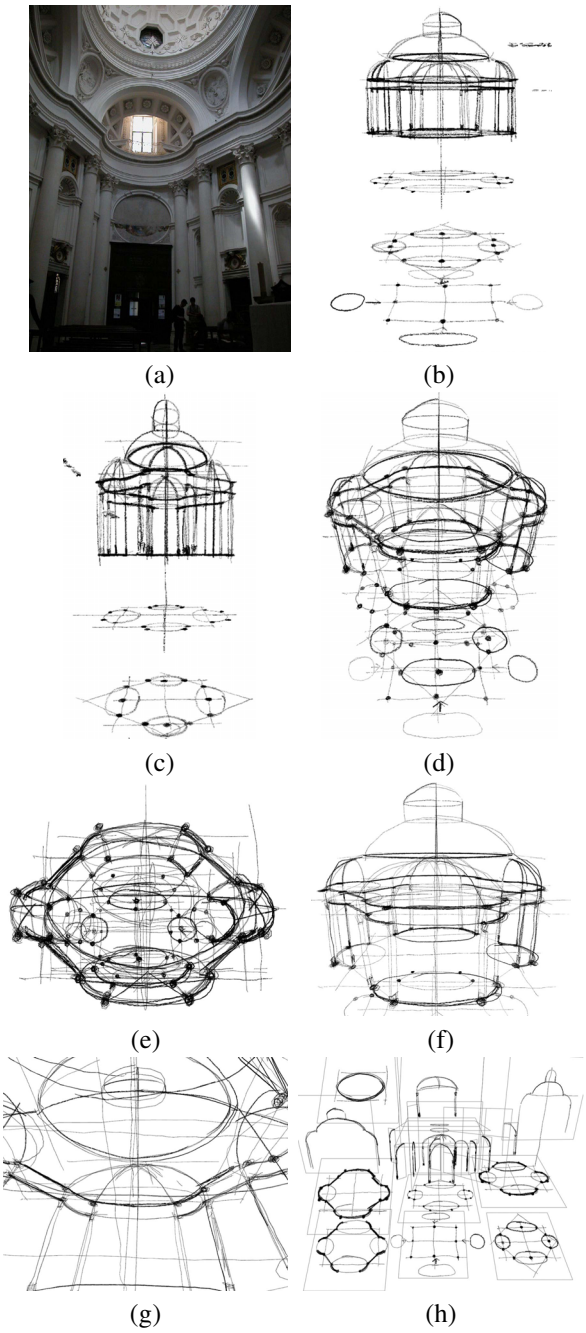


Figure 13. Analysis of the interior of *San Carlo Alle Quattro Fontane*: a photograph (a); the front (b), side (c), 45 degree front top (d) and an overhead view (e); a closer look toward the interior (f); and an interior view (g). (h) shows all of the sketch drawings the architect developed for this analysis.