Gestures Modeling to Support Creative Design

**Designer Architect**
Creative Institute
University of Design Cognition
+1 234 567 8901
bb@ccc.ddddd

**Computer Scientist**
Cognitive Study
Institute of Embodied Interaction
+1 234 567 8901
bb@ccc.ddddd

**ABSTRACT**

In this paper, we discuss the role of spatial cognition during the architectural design activity, and how it affects design creativity. Recognizing that current CAD applications’ Windows-Icon-Menu-Pointer (WIMP) user interfaces do not fully support creative exploration in modeling geometries for design, we design a prototype system that incorporates a regular PC and a depth camera for gestural input. We present the implementation and use scenario of the system, and discuss why gestural input may advance creativity in the design process.

**Keywords**
Spatial Cognitions, Design Creativity, Architectural Design, Gesture Modeling, Virtual Reality

**CREATIVITY IN DESIGN PROCESS**

Spatial cognitions in architectural design involves the understanding of design forms, space functions and relationships between spaces, is one of the most challenging activities in the early architectural design process. It helps architects to discover problems associated with the contextual environment, which are ill-defined and difficult to specify [1]. Spatial cognition also helps designers solve problems in a creative way.

Usually, a skillful architect can quickly discover the problems from external representations, such as simple sketches or draft models, because they can infer proper mental representations from them. The mental activities for observing these external representations could be as simple as the comparisons of the relationships of two lines, or as difficult as constructing a 3D mental model after inspecting a full set of floor plans. Even in the simple example, the two lines could represent two walls in reality. In this case, the internal geometric representations for an architectural plan could be extremely complicated. Thus, a master architect requires a significant amount of training in building the connections between mental models and external representations during the early years of their design careers [2].

Architects often self-impose constraints in their design process. According to Bonnardel’s study, professional designers add constraints to help filter out impracticable possibilities [3]. In the case of an architectural design process, the common practice is to first obtain spatial information such as the site condition and spatial programs of a building. Then they use these conditions as creative mechanism during the design activities [3].

On the other hand, the spatial cognition of an external representation could also stimulate new ideas designers haven’t expected and as a result, improve their designs. These new ideas usually came from an analogy-making process in which they may compare other cases in their mind to the external representations to improve their designs [4]. Thus, external representations can be at least partially inspired by their previous experiences. The experiences here could be in the form of any sensory inputs, such as visual, auditory, tactile, or even the motion in a space, captured by a Kinetic. Even though the ideas come from various experiences, as soon as the new ideas emerged, designers need to record them as external representations and inspect whether the new representations work in the existing contextual conditions. These are the reasons why architects create 2-D and 3-D representations during the early design process. With careful consideration of the various contextual environments around the site of the proposed building, architects could create their own original designs through both external and internal representations. Design creativities come from the process of making these internal and external representations.

**THE BODY IN OUR MINDS**

Among the senses mentioned before, vision and motion are two senses most closely related to spatial cognitions. Many researchers have discussed the influences of visual/external representations during the design process. Because we have to experience the spaces, inspect drawings, and construct physical models through our eyes, there is no doubt that visual processing plays an important role in spatial cognitions [5]. Therefore, we have seen many discussions on how the external representations affect our decision-making in design process.

An important factor of spatial cognition is the perception of motion in space. Johnson states, “our understanding is our mode of being in the world”, and thus “we are meaningfully situated in our world through our bodily interactions” [6]. In the traditional design activity, architects draw a segment of line with one hand holding a pen and moving for few seconds. When building a physical model, architects rely on moving and modifying pieces of materials in different conditions.
sizes and locations. In a real building space, we may walk through a long corridor in a museum and feel the sunlight shine through the skylight. All these interactions with physical objects or spaces require body awareness.

**CURRENT CAD TOOLS**

We have discussed how designers make their external representation and how the sense of vision and motion affects the process of decision-making. These discussions demonstrate the importance of direct manipulations for external representation. However, current CAD tools for architectural designers only allow for 2-D on screen manipulations. Designers manipulate mouse and keyboard to create geometries. The visual representations on displays cannot be correlated to direct body movements. In addition, there is a distance between designer’s manipulation area (mouse and keyboard) and the representational space (displays). Moreover, no body awareness can be related to the working representation on the screen. In other words, there is no real direct-manipulation of 3-D virtual objects in current Graphical User Interfaces (GUIs) settings. Even though CAD tools could help designers make complex geometry from parametric modeling or the benefits from rapid prototyping, researchers still argue that the current CAD systems could hinder the creative design process due to the lack of support for spatial cognitions [2].

A reason that current CAD systems hinder creativities may due to that the mental representations [5] cannot be directly represented on the digital environment. As architects have to discover design problems and find ways to solve them, architectural design in the early stage requires designers to constantly create the mental representations and physical representations at the same time [7]. Since the CAD system can only show projective images instead of real life three dimensional objects, designers require additional mental load to process the 3D design context [6].

Furthermore, because CAD tools adopt the command-execution and Windows-Icon-Menu-Pointer (WIMP) styles of inputs, users require a certain amount of learning periods before they can be fluent with the CAD programs. Even though this style of user interface is effective in recording and producing precise digital models, these CAD tools do not support design creativities for spatial cognitions. Another reason to adopt CAD tools during the design process is to employ the calculation, simulation and automation ability that computers provide. In this aspect, computers could help us to generate the creative forms that were not possible before. For example, augmented with computation, some CAD tools provide the ability to manipulate parametric rules in generative systems. However, as we mentioned before, without the proper design study to set the rules before this step and define the problems, these creative forms may produce chaotic results. In this sense, computers are obviously the experts in computing, recording, and remembering. However, “they are poor at recognition, and the reconciliation of conflicting task” [2].

In this sense, in order for computing tools to really support design process, we need to teach computers to understand what designers are doing, what objects they are creating, and what the design intents are [8]. There are many projects addressing the recognition of 2-D sketches and designers’ behavior in the past twenty years [8, 9]. In the category of 3-D direct manipulations, there are, however, little efforts in computing to recognize human behavior and provide proper reactions to assist spatial cognition in design activity. One possible way for computers to recognize the models that designers are building and manipulating is through the use of Tangible User Interfaces [10, 11]. In this paper, we attempt to resolve these issues by using free-hand direct manipulations with virtual interfaces. Before describing our prototype system, let’s review related gestural input projects in the next section.

**FREE-HAND DIRECT MANIPULATIONS**

Using hands to directly manipulate computer graphics is not new. Thirty years ago, Bolt built the first gesture pointing system with speech recognition for creating computer graphics and modifying the sizes and the locations of each shape [12]. This system had many usages, including the device like a whiteboard for discussing the strategy for the military domain. The early systems like this relied on instrumented gloves to capture locations of the hands and fingers. The hand-pointing direction and the cursor on the screen form a spatial relationship that helps users to directly manipulate the 2-D graphics on the screen. In 1984, Nemeth made the first attempt to utilize hand input for CAD system to support designers with a better spatial cognition with their designs [13]. In order to let users engage more in the modeling environment, Donath et al. adopted virtual reality (VR) with instrumented gloves to create models for design [14]. The system supports designers to generate simple geometries and navigate it with a head-mounted display. These researchers argued that VR gives us the maximum fidelity to design in situ. However, all of these prototype systems require heavy instrumented settings and pre-settings and the physical configurations would not be comfortable for long term manipulations for designers.

In order to make friendlier configurations for designers, Gesture Modeling reduces the size of the work space to only a desk. The system employs both gesture languages and VR for inputs and outputs [15]. The project employs a camera and gloves to trigger the commands and edit the 3-D shapes without electronic pieces for detecting the hand gestures. The depth data is calculated by the size of the hand from the camera. In this project, Gross et al. define a series of gestures that users could utilize during the modeling process, such as “poke”, “gun”, “pinch”, “thumbs up”, and “five”.

In recent years, due to the wide availability of sensing technologies, many projects adopt new interaction techniques to facilitate more natural ways to interact with the computers. However, Don Norman argues that many
Natural User Interface projects, in which inputs are different to our current GUI conventions, are actually not natural [16]. He provides the early mouse gestures in Xerox Star system as an example to illustrate why a lot of these new interaction methods disappear in the future. The requirements of remembering these new gestures to trigger commands are just not natural. Though, he also believes that these new technologies will find proper places to survive [16]. Many other researchers, who adopt natural styles of interactions, also believe that by using body motions and visual feedbacks in the computer interface, people may be able to walk-up-and-use without having to learn specific interaction techniques [17].

The project “Imaginary Interfaces” [18] gives us an interesting demonstration of how the hand spatial interaction with computers could be intuitive with a reference area without any visual cue. In this study, the researchers adopted one hand as a reference plan, and the other hand as the drawing and pointing interface. The study shows that there is a strong correlation between the pointing locations with reference to both visual memory and the memory of hand movements [18]. This gives us the confidence that the 2-D direct manipulations in space could be actually viable while not overloading mental memory. With the benefits of using motion inputs, we believe that a proper set of gestures and modeling constraints with motions could improve our 3-D spatial cognitions in the modeling process. In the next section, we will discuss our prototype system, the gestures and motions implemented for the modeling task.

**SYSTEM DESCRIPTION**

Even though CAD systems are useful for design, there still are many usability issues. For example, if a user wants to build a surface, a program needs to know the normal vector of it. Otherwise, a program cannot decide whether the vertices of the surface are actually for composing a hole or not. Past decades have seen many different approaches of describing computer geometries, such as spline for 2-D curves, solid models for 3-D operations, mesh surfaces, and nurbs. These modeling methods help designers explore creative design with parametric modeling. Meanwhile, we are interested in exploring different ways to interact with the computers to support creative design.

In our first implementation, we employ SketchUp as the host program with a plug-in for controlling geometries and navigating the cameras because SketchUp adopts many metaphors from our physical environments, such as push/pull, and drawing right on surface. Moreover, it helps us to avoid some of the restrictions mentioned earlier. In contrast to the mouse control in the native SketchUp modeling environment, we support users to use both hands to directly manipulate the geometries. As show in Figure 2, the two hand cursors in the display represent the location of the user’s two hands. When a hand “touches” an existing shape, the system will give visual feedbacks according to the gestures.

In the hand gesture part, each hand plays a different role. For example, the left hand serves as a 2-D reference plane in the 3-D space to let the right hand draw shapes on it. The drawing gesture for right hand is “Pointing” (frame C of Figure 1). This gesture allows users to draw any shape. After the camera detect the points of the right hand gesture, the host computer will recognize them as basic geometric shapes, such as rectangles, triangles, circles, and polygon shapes. These basic shapes could become the profile shapes for 3-D manipulations. In this version of the prototype, we implemented several fundamental 3-D modeling behaviors, such as extrude, scale, and Boolean operations.

![Figure 1. Basic Gestures in the Prototype System](image1)

![Figure 2. A screen shot of the 3-D modeling environment of the system](image2)

Since the concepts of these modeling commands are actually based on our physical environment, we could let users perform some basic gestures, such as pulling for extruding in frame F of Figure 1, pinching for scaling by combining frame B and frame E of Figure 1, and grabbing for moving geometry by using right hand shown in frame D of Figure 1. Instead of only executing the commands by showing these gestures, the spatial information for these gestures is also detected by depth camera so that the system can know how far the user is pulling in scale, and toward which direction.

**IMPLEMENTATION**

In order to build the prototype, we employ a depth camera, Microsoft Kinect sensor for recognizing hand gestures and
3-D locations, and a host computer with a screen for computer vision analysis and visual feedback generations. There are two software components: (1) analyzing image stream from the device and (2) generating the geometry representations. In the first part, we adopt OpenNI [19] as the middleware for natural interaction library, and OpenCV [20] for analyzing each image frame. The former software library helps us to detect hand locations and managing image streams from the device. As shown in Figure 3, the depth camera can generate the regular RGB frames as well as depth raw data in 30 frames per second. The depth raw data not only gives the distance from the camera to objects in each pixel, but can also produce grey-scale images. These grey scale images are used for recognizing hands and fingers to produce the meaningful gestures in our prototype system. When the software in the first part recognizes the movements and gestures, the program in the second part will execute certain actions, such as generating a 3-D shape, or moving the scene or a shape. In this version of the prototype, we use SketchUp as our host program with customized Ruby plug-in to meet the requirements.

Figure 3. Depth Image in Grey Scale and RGB Regular Image from A Depth Camera

**Recognizing Gestures**

Recognizing a gesture from a regular camera requires heavy computer processing powers to filter out unnecessary pixels. In our prototype, the depth frames from the depth camera helps us to easily filter out those unnecessary pixels so that we can focus on analyzing where the fingers are. Once the program receives the depth frame from the middleware, it tries to detect the hand center locations and the depth through the routine call of the middleware. If there are two hands in that frame, the program will go into the finger detecting process. The process simply takes the pixels where the depth distances are only 5 cm to the depth of hands that the middleware gives. The rest of pixels are set to black so that the finger detecting process will not need to compute.

The filtered frame then is sent into finger detecting process. In this process, the program utilize the routine call of OpenCV to get the contour points for each hand, and then compute the convex hull1 of each hand from the contour points. By computing the intersection points of the hand’s contour and convex hull polygons, the fingertips and defects areas between fingers could be retrieved. These fingertips locations on a depth frame image are also used for querying the depth distance in the 3-D scene. Thus the vectors for each finger can be calculated by using the location of each fingertip and the depth point. The red dots of the right figure in Figure 3 are the fingertips, and the yellow dots are the depth locations between fingers. These vectors are used for recognizing the gestures. The location for fingertips could also be the place where the user is pointing.

Once the program recognizes a gesture, the program tries to decide which 3-D location in the physical world the hand is. For example, if the gesture is “Pointing” (frame C of Figure 1), the location will be the fingertip point. If the gesture is “Gun” (frame B and E of Figure 1), the location, however, will be the depth point. When the program decides which gesture is the current one, it will trigger the commands automatically, such as drawing, pulling, pushing, or moving geometry.

Figure 4. The lines made from fingertips and depth point between two fingers

**Recognizing 2-D Shapes**

Among the commands we described in the system description part, the most important one is to draw a 2-D shape on a reference plane. When users’ right hand perform a “Pointing” gesture, the system will wait for the fingertip location to go into the reference plane area, which the left hand created by using “Gun” gesture. As soon as the right hand goes into the area, the points will be transferred to our modeling program – SketchUp. If the points can proximately form a straight line, the program will generate a line for the user. If the points form a corner, the program will create a vertex at the location. When users finish a shape, the program will have a routine call in OpenCV for recognizing whether the shape is one of the basic shapes, such as a rectangle, a triangle, or a circle. If not, the program will adopt the vertices that it originally created.

**Creating a 3-D Geometry**

In this version of implementation, we employ push and pull for creating a 3-D geometry. Users only need to use their right hand to perform the gesture in frame F of Figure 1. When their hands are close enough to a shape created in the previous step, it will trigger push/pull with their hand motions. Once they create a surface or a 3-D geometry, users could create 2-D shapes on them by using the method we described before. In this sense, users can also push/pull the profile on existing surfaces to create more complicated shapes.

---

1 A convex hull is a bounding polygon of a selection of points that all the outer angles are convex.
Visualization Feedback
In this program, we have two separated views for users to receive the visualization feedback. One is a regular view for showing the perspective window, and the other one shows a 2-D top view of the current scene. We believe that these two views give designers enough contextual information so that the designers can imagine the spatial conditions in front of them.

In the navigation part, a 3-D modeling program usually would have rotation, pan view, and zoom for the navigations. For panning and rotating the view, we simply restrict the view camera of the system on a fix height in order to reduce the complexity of gesture recognition. For panning, users simply perform the gesture “Five” on a left hand and move their hand around. If they want to stop panning, they only need to turn their hand to the “Zero” gesture. For rotating the view camera, users need to use their both hands to perform “Five” gestures and make a movement like a circle. The view will rotate according to the center of the circle they perform. For zooming in and out, users have to perform “Gun” gesture on both hands. The scale of zoom is according to the distance between two hands.

Discussion
This prototype system creates a new way of generating computer geometries for designers. Even though designers cannot create precise shapes in this system, they can use their hands freely to create shapes according to their imaginations and spatial perceptions. This prototype system adopts the movement of both hands to control the pushing/pulling, scaling, and panning/rotating view. These movements provide additional perceptions that the traditional modeling system couldn’t provide. We are planning user studies to understand how these movements affect users’ spatial cognition.

FUTURE DIRECTION
Besides conducting a rigorous user study, we would like to add more features into our prototype system. Since spatial cognitions are the most important activity during architectural design process, we believe that adopting augment reality with the system would improve the spatial perception. In the future, while the system includes more features, the menu and visual feedback projecting on the desktop could also benefit users.

SUMMARY
In this paper, we have discussed the roles of spatial cognitions during the design process, and how spatial cognitions affect the design creativities. In the past, researchers have focused on the relationships between visual perceptions and spatial perceptions. We argued that the sense of motion may also play an important role in terms of experiences and imaginations in their mind. These experiences can also assist designers during the design process with visual assistance. Thus, we built a prototype based on the hypothesis.

ACKNOWLEDGMENTS
Keep anonymous for blind review for now.

REFERENCES


16. Norman, D. A. Natural user interfaces are not natural. Interactions (17, 3 2010), 6-10.


