Volume 18 (*1999*), *Number 3*

A Translucent Sketchpad for the Virtual Table Exploring Motion-based Gesture Recognition

L. M. Encarnação[1], O. Bimber[2], D. Schmalstieg[3], and S. D. Chandler[1]

[1]Fraunhofer Center for Research in Computer Graphics, Inc., Providence, RI, USA
[2]Fraunhofer Institute for Computer Graphics, External Division Rostock, Rostock, Germany
[3]Vienna University of Technology, Vienna, Austria

Abstract

The Virtual Table presents stereoscopic graphics to a user in a workbench-like setting. For this device, a user interface and new interaction techniques have been developed based on transparent props –a tracked hand-held pen and a pad [12]. These props, particularly the pad, are augmented with 3D graphics from the Virtual Table's display that can serve as a palette for tools and controls as well as a window-like see-through interface, a plane-shaped and through-the-plane tool, supporting a variety of new interaction techniques. This paper reports on an extension of this user-interface design space which uses gestural input to create and control solid geometries for CAD and conceptual design. The application of gestural interfaces is a common method for interacting with virtual environments on a habitual and natural basis. The motion-based gesture recognition presented here uses Fuzzy Logic to support a predictable, flexible, and efficient learning process. This new interaction paradigm greatly increases the Virtual Table's suitability for design tasks. Traditional CAD dialogue can be combined with intuitive rapid sketching of geometry on the pad. Additionally, the resulting events and objects can be associated with scene details below the translucent tablet.

1. Introduction

The application of gestural interfaces is a common method for interacting with virtual environments (VE) on a habitual and natural basis. Gestural interaction is therefore an enrichment of VE interfaces that incorporate true 3D input and output technologies, e.g. six degrees of freedom (6DOF) sensors and stereoscopic displays. In a collaborative research effort, we have developed a system that uses transparent props for two-handed interaction [12] on the Barco BARON [2] Virtual Table (VT), a tabletop VR display based on a workbench metaphor [8]. This system aims to combine well-understood desktop metaphors with a virtual reality interface. The hand-held transparent props are a pen and a pad, and related to earlier research on the Personal Interaction Panel (PIP) [16], an augmented reality interface. While augmented reality systems use semi-transparent or video-based head-mounted displays to overlay computer graphics onto real-world objects, our system overlays transparent physical props onto the back-projected display of the VT to achieve a kind of inverse augmented reality that we call *augmented VR*.

Our system unifies several previously isolated approaches to 3D user-interface design, such as two-handed interaction and the use of multiple coordinate systems, but more importantly it allows for experimentation with the affordances of transparent props that–with the exception of the metaDesk [17]–are generally unexplored. Our interface supports the following important features:

- two-handed interaction
- multi-purpose physical props
- embedding 2D in 3D
- use of multiple coordinate systems (i.e., of the table and the pad)
- transparent tools, especially window-tools and through-the-plane tools.

Each of the listed properties allows the design of distinct forms of interaction.

This paper describes our efforts to explore these possibilities of transparent props for object sketching and control using motion-based gestural interaction. After describing the system setup used for our experiments in Section 2

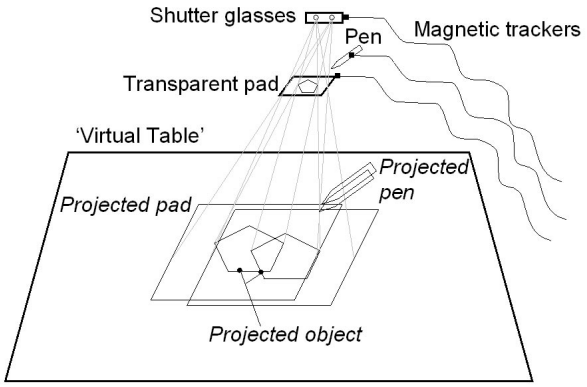

Figure 1: *The Virtual Table's display creates the illusion of graphics aligned with the pen and pad.*

an overview on related work is given in Section 3. We report on the interaction techniques supported by our transparent props in Section 4, and then describe our motion-based gestural interaction in further detail in Section 5. Finally, we present results and observations of the system in practice.

2. System Setup

The employed system uses the Barco Baron Virtual Table as its display device. This device offers a 53"x40" display screen built into a table surface and connects to an SGI Indigo2 Maximum Impact workstation. Together with shutter glasses (Crystal Eyes from StereoGraphics or 60GX from NuVision), a large, very bright, high-contrast stereo display is available. The transparent props we use are an 8"x10" Plexiglas sheet and a large, pen-shaped, plastic tube (Figure 7) which is also fitted with a button. Both props as well as the shutter glasses are equipped with 6DOF trackers (Ascension Flock of Birds) for position and orientation tracking. Using the information from the trackers, the workstation computes stereoscopic off-axis projection images that appear in perspective based on the user's head position (cf. Fig. 1). This property is essential for the use of augmented VR, as the physical props and their virtual counterparts have to appear aligned in 3D.

The material for the pen and pad was also selected for minimal reflectivity so that with dimmed lights –the usual setup for working with the VT– the props become almost invisible. While the props retain their tactile property, in the user's perception they are replaced by the graphics from the VT. Our observations and informal user studies indicate that virtual objects can even appear to float above the Plexiglas surface, but that conflicting depth cues resulting from such scenarios are not perceived as disturbing. Conflicts occur only if virtual objects protrude from the outline of the prop as seen by the user because of the depth discontinuity. The most severe problem is occlusion from the user's hands. Graphical elements on the pad are placed in a way so that such occlusions are minimized, but they can never be completely avoided. The pen was chosen to be relatively large to provide room for graphics displayed inside the pen. In that way, the pen also provides visual feedback, such as showing what the tool is currently associated with. So far, however, we have made only basic use of this capability and have instead focused on the pad as a carrier for the user interface.

3. Previous Work

The work presented here is based on the system presented by us in [12]. This approach was originally inspired by the work on the Personal Interaction Panel [16]. This work explored the use of (opaque) pen and pad props in a head-mounted, see-through augmented reality system called Studierstube [11]. Other researchers use pen and pad props, though either in fully immersive or desktop setups: In [10] a system is described for the design of 3D curves and shapes. In [1] the authors report on their use of pen and pad props for embedding traditional 2D GUIs in a 3D immersive system. 3D Palette is described in [4] as a virtual content creation tool using pen and pad props in a fishtank VR setup. With respect to using gestural pen-based 3D input for sketching and conceptual design, Sketch system [20] is probably the most relevant and influential to the presented work. While Sketch would require reaching out on the large surface of the virtual table in order to perform gestures, our approach provides the convenience of a hand-held pad with extended interaction functionality due to its transparent character and its visual presence in the virtual scene.

4. Pen-and-Pad Interaction

The focus of our work is to explore the user-interface and interaction possibilities of the transparent pad as a distinct object. While the two-handed pen-and-pad metaphor is asymmetric [6] and the pad is assigned the more "passive" role (e.g., it is held in the non-dominant hand), it has much more interesting affordances than the pen. The pen and pad have a relationship similar to the mouse pointer and window in a conventional desktop system. However, the contrasts to the desktop are not only that pen and pad operate in 3D, but also that the pad is directly controlled by the user's non-dominant hand and can therefore additionally be used as an active tool.

The pad therefore represents an embedding of 2D in 3D, as already pointed out by [1]. Yet its possibilities extend far beyond that, since it combines several individual metaphors [12] by serving as a tool and object palette (as e.g. in SmartScene [7]), a window tool (as e.g. the Virtual Tricorder [18]), a through-the-plane tool, and a volumetric manipulation tool (as e.g. the World-in-Miniature work [14]). These options co-exist in the design space of our user interface and together form a very powerful and general framework for 3D interaction. Since the physical and geometric properties of the pad are

very basic in nature, it is possible to use all the metaphors mentioned above for application tasks without confusing the user. Our transparent props thus form a two-handed multi-purpose tool for the interaction in virtual worlds.

We believe that these features, particularly the palette and through-the-plane tool characteristics, highlight this system's suitability for gesture-based object creation and manipulation:

- As *tool and object palette* the pad can carry tools and controls, much like a dialog box works in the desktop world. It can also offer collections of 3D objects to choose from. The tool and object palette not only supports the designer with a non-obstructive 2D interface for a VR environment, but also provides the tactile feedback of a planar surface that is prerequisite for sketching.
- As *through-the-plane tool* the pad can be used by the user to orient the "window" defined by the pad, and then manipulate objects as seen through it, i.e. manipulate the 2D projections of objects on the pad. The through-the-plane characteristics allow the user to control and manipulate existing objects in the scene without having to leave the working plane of the physical pad.

Combining these two metaphors, we implemented *context-sensitive dialogue elements* to manipulate the geometry created through sketching. Dependent on the object targeted through the pad with the aid of a cross-hair, different numbers of sliders appear together with the corresponding labeling of the sliders as well as of the geometry type itself (cf. Fig. 6). This metaphor could be easily extended towards object placement for assembly tasks, so that targeting the vertex, edge, or face of a work piece could be used as a snapping destination for another part. Such functionality that requires the support of a more sophisticated modeling system will be investigated in the near future.

4.1. Gesture-based Object Creation

The gestures that are used for object creation were developed to be as intuitive as possible, to ensure easy memorization. In addition, since the user looks through the transparent pad onto the scene that is displayed on the Virtual Table, the gestures have been designed to follow the contours of the top-down projection of the corresponding solid geometries as closely as possible. This differentiates our approach from the one presented in the Sketch system [20], where the user mainly outlines the side-view contours of an object to generate basic solids. Top-view outlines are used in Sketch in a special bird's-eye mode to produce 2D contours to define the silhouettes of 3D geometry created in a subsequent step.

The currently implemented one-motion gestures are conceptually structured in a hierarchical order. Their beginning and end are defined by pressing and releasing a button attached to the transparent pen. Gestures may consist of several pen strokes that are performed close to the pad. These tools are used much like pen and paper, except instead of actually drawing a shape, the computer scans the strokes made on the pad. The strokes' proximity to the pad determines whether or not they contribute to the gesture to be recognized.

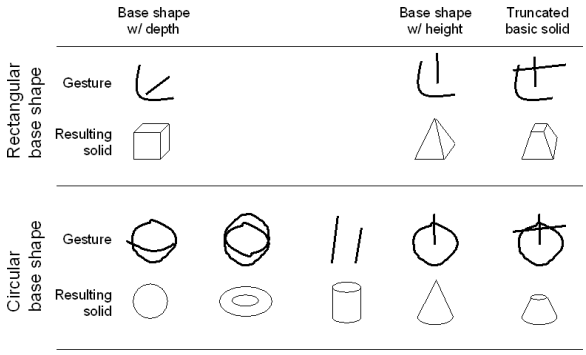

Figure 2: *Sample gestures for object creation.*

The gestures support the generation of 3D objects (cf. Fig. 2) with circular base surfaces, contours (e.g., sphere, cone, truncated cone, cylinder, torus), or rectangular shapes (e.g., cube, pyramid, truncated pyramid). In general, the objects are created by first defining their base surfaces or contours (cf. Fig. 4). Afterwards, a stroke defines either the depth (e.g. for cube) or the height (e.g. for cone). Gestures for truncated solids resemble their non-truncated equivalent in that only the height stroke is extended by a horizontally cutting finishing (cf. Fig. 4). Obviously, special solutions must be developed for cylinder, sphere, and torus generation, since these objects would be created using ambiguous gestures. Therefore, the cylinder gesture makes the exception of employing its side-view contour by being defined through two parallel lines. The torus is defined by two circular gestures. This leaves for the sphere a circular gesture and an arc gesture, which indicates the sphere's curvature in all dimensions. Although apparently several gestures show close correspondences, the recognition rate is generally between 95% and 100%.

4.2. Gesture-based Object Control

The defined gestures for object manipulation and control are currently limited to the selection and deletion of objects (cf. Fig. 5). Additional control gestures are available that perform mode changes, thus relieving the user interface apparent on the pad from unnecessary dialogue buttons. Figure 3 shows the different gestures for object control and mode changes. Although again several gestures show close correspondences, the recognition rate is once again between 95% and 100%.

Putting a major focus on intuitivity and the support for easy recollection, objects are selected by circling their pro-

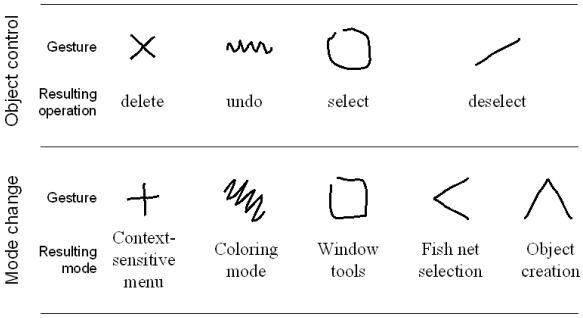

Figure 3: *Sample gestures for object control and mode changes.*

jected images that are viewed through the pad. (*Note:* This functionality was actually already supported by the system described in [12], without using the motion-based gesture recognition presented here. In a similar way, objects are deleted by “crossing them out" on the pad. Undo is represented by a “scribbling" on the pad, thus resembling the erasure of mistakes on common paper.

5. Motion-Based Gesture Recognition

The teaching of gestures to the system and the correction of recognition errors are accomplished using dialogue elements on the pad (cf. Fig. 4,5 along the left side of the pad). Dialogue buttons are associated with each of the following events:

- Loading a set of gestures from a file,
- Saving to a file the currently loaded and recently added gestures,
- Issuing the ‘Learn’ command to the recognizer using the most recently performed gesture,
- Advancing to the next gesture index.

The last event is especially helpful when initially teaching the system each of the gestures one by one. Also when a gesture is incorrectly recognized, this button can be used to add a variation of a gesture to help clarify and hone the process. For instance, the creation of a torus as shown in the bottom row of Figure 4 can be accomplished by penciling two circles onto the pad which are intersecting or not. This allows for much more ‘sloppiness’ in the user’s drawing, i.e. for *true* sketching.

Transforming an expressed gesture into a meaningful statement can be a computationally intensive task and may not be easy to achieve in real-time (i.e. together with rendering, tracking etc.). Usually techniques of artificial intelligence and machine learning are applied to solve classification problems such as the recognition of gesture or speech. Earlier approaches (e.g. based on neural networks[5] or hidden Markov models[9]) usually lack in predictability, flexibility, or performance of their learning process. Since most of them recognize state-based gestures (iconic[13], emblems[3]) we apply Fuzzy Logic[19] in terms of recognizing motion-based gestures.

The method described recognizes previously learned gestures that the system was taught by having a user perform them at runtime. Any kind of 2D, 3D, or 6DOF input device can be used to gather the motion data that carries out the gesture. The reliability of the recognition process can be improved by repeating the same gestures several times and correcting incorrect recognition results. This extends the system’s knowledge. Once learned, it translates the recognized gestures into (for a computer) identifiable objects (numbers, strings, events, etc.) that can be further processed. During a three-stage process, the raw position and orientation data is first used to continuously update the gesture-specific basic information (e.g. bounding box, length, orientation etc.) on the fly. Once scanned, the data serves as basis for calculating a set of fuzzy values that characterizes the gesture in the second stage. Approximated reasoning is dynamically applied to express the trust in each characterization criteria (so-called aspect), depending on the appearance of extreme situations. Combined, each individual set of aspects represent a product rule that describes the degree of membership in a specific class of gestures. These product rules are finally used to find the gesture with the best match among the already learned ones. To do so, we compare the aspect set of the gesture to be identified, with the ones stored in the knowledge-base. The gesture with the smallest total deviation is suggested as the most-likely candidate.

5.1. Basic Information

The basic information consists of gesture-dependent properties that are updated througout the scanning of gestures. This information is the basis for computing the characterization aspects. Because of its similarity, we will only discuss the processing of position data. Orientation information (e.g. azimuth, elevation and roll alignments) are treated in the same way.

If we assume that the position data is represented as follows :

$$G = (x_0, y_0, z_0), (x_1, y_1, z_1), ..., (x_{n-1}, y_{n-1}, z_{n-1})$$

for a gesture G described with n position samples, then the edges of its bounding box can be calculated with :

$$left = min(x_i, left), right = max(x_i, right),$$
$$bottom = min(y_i, bottom), top = max(y_i, top),$$
$$back = min(z_i, back), front = max(z_i, front)$$

The total length of the gesture is :

$$l = \sum_{i=1}^{n-1} \sqrt{\Delta x_i^2 + \Delta y_i^2 + \Delta z_i^2}$$

with $\Delta x_i = x_i - x_{i-1}, \Delta y_i = y_i - y_{i-1}, \Delta z_i = z_i - z_{i-1}$.

The horizontal, vertical and depth movements are :

$$hm = \sum_{i=1}^{n-1} |\Delta x_i|, vm = \sum_{i=1}^{n-1} |\Delta y_i|, dm = \sum_{i=1}^{n-1} |\Delta z_i|$$

The center of gravity is :

$$cx = \sum_{i=0}^{n-1} x_i, cy = \sum_{i=0}^{n-1} y_i, cz = \sum_{i=0}^{n-1} z_i$$

The angle coefficients (legs of a triangle) to calculate the directions of the gesture's start and end sections :

$$sax = \sum_{i=1}^{n-1} \frac{\Delta x_i}{(l_1^i)^p}, say = \sum_{i=1}^{n-1} \frac{\Delta y_i}{(l_1^i)^p}, saz = \sum_{i=1}^{n-1} \frac{\Delta z_i}{(l_1^i)^p},$$

$$eax = \sum_{i=1}^{n-1} \Delta x_i (l_1^i)^p, eay = \sum_{i=1}^{n-1} \Delta y_i (l_1^i)^p,$$

$$eaz = \sum_{i=1}^{n-1} \Delta z_i (l_1^i)^p$$

where l_1^i is the momentary length of the gesture (from segment 1 to segment i). The angle coefficients are used later to calculate the gesture's start and end angles, and to represent its directions (x,y,z components) at the beginning and the end. Note that with increasing distance from the start position (and decreasing distance to the end position), the weights of the start-angle-coefficients (sax, say, saz) decrease while the weights of the end-angle-coefficients (eax, eay, eaz) increase. This causes a sufficiently realistic weighting of the gesture's start and end sections and prevents the computation from taking the distortion into account that usually appears at the beginning and the end (e.g. introduced by the tracking device or the user). The exponent p is used to control how fast the weights increase or decrease (with respect to the length of the gesture). Experimentally, we have chosen $p = 2$ for our examples.

To achieve a higher reliability, we do not use the original motion information only. We also rotate the original positions around the x, y, and z axes and calculate the basic information (as described above) for the rotated information as well. This makes it possible, for example, to track the diagonal movements. In our example we rotate the original positions by 45° to achieve a higher performance. As a result we can replace the time costly sine and cosine functions by multiplications with the constant factor $c = sin(45^\circ) = cos(45^\circ) \simeq 0.7071067$.

$$xR_i = c(c(x_i + c(y_i + z_i)) - c(y_i - z_i)),$$
$$yR_i = c(c(x_i + c(y_i + z_i)) + c(y_i - z_i)),$$
$$zR_i = c(c(y_i + z_i) - x_i)$$

Furthermore, we do not have to perform all multiplications, because scaling of the gesture does not effect the result.

5.2. Characterization Aspects

After the gesture is scanned and the specific basic information is obtained, we can calculate fuzzy values (aspects) that express the grade of membership to their fuzzy sets (e.g. the set of intricate gestures, the set of flat gestures, etc.). Note, that in the following we will discuss only a subset of the defined aspects to illustrate the principles. For performance reasons, we use an integer representation, thus the aspects are normalized to a range of $[0, 100]$.

A gesture's length is at least as long as the diagonal of its bounding box. We use the ratio :

$$a_0 = \frac{\sqrt{width^2 + height^2 + depth^2}}{l} \cdot 100$$

with $width = right - left, height = top - bottom, depth = front - back$ to describe this relation. Results of around 100 indicate that the gesture describes a straight line (i.e. the smaller the value, the more intricate and complex the gesture).

We use the following equation to describe the bounding box's height/width ratio :

$$a_1 = \frac{height}{height + width} \cdot 100$$

Values of around 50 indicate a square height/width relation. We can derive : The smaller the value, the wider the bounding box; and the larger the value the higher the bounding box. Similar ratios can be calculated to express the height/depth (a_2) and depth/width (a_3) relations; and the relations for the rotated gesture (a_4, a_5, a_6).

To express the relative start and end positions, we can use the following ratios :

$$a_7 = \frac{x_0 - left}{width} \cdot 100, a_8 = \frac{y_0 - bottom}{height} \cdot 100,$$
$$a_9 = \frac{z_0 - back}{depth} \cdot 100$$

Values close to 100 indicate that the gesture does begin close to the specific face (right, top, front). Values of around 50 indicate that the gesture begins at the center, and values close to 0 indicate that it begins at the corresponding opposite sides (left, bottom, back) of the bounding box. Similar ratios can be formulated for the end position (a_{10}, a_{11}, a_{12}).

The sum of all horizontal movements is at least as great as the width of the bounding box. We use the following ratio to form a useful statement :

$$a_{13} = \frac{width}{hm} \cdot 100$$

For example :

We obtain a value of around 33 if the projected gesture describes a 'Z' on the x/y-plane. If it describes an 'M', the result is nearly 100; and if it describes a 'C', we obtain a value of around 50. Similar values can be calculated for the vertical and the depth movements (a_{14}, a_{15}), and for the rotated gesture (a_{16}, a_{17}, a_{18}).

To determine in which direction the gesture starts and in which it ends, we calculate the start and end angles using the

horizontal/vertical and depth angle-coefficients, which have been tracked during the scanning process. These computed angles represent the gesture's direction at the beginning and the end (x,y,z components). The angles are calculated based on the gesture's projection onto the x/y-plane, x/z-plane and z/y-plane. Note, that the angle-coefficients represent a weighted sum of the movements with increasing weights for the end-angle-coefficients (the closer we get to the end) and decreasing weights for the start-angle-coefficients (the further the distance from the beginning). This causes a sufficiently realistic weighting of the gesture's start and end sections and prevents from taking the distortion into account that usually appears at the beginning and the end (e.g. introduced by the tracking device or the user).
We calculate the angles at the beginning as follows :

$$a_{19} = angle(sax, say), a_{20} = angle(sax, saz),$$
$$a_{21} = angle(saz, say)$$

where $angle(x,y)$ calculates the angle by normalizing the result of $a = acos(\frac{x}{\sqrt{x^2+y^2}})$ to the range of $[0,100]$ with :

$$b = \begin{cases} 2\pi - a & : \quad y < 0 \\ a & : \quad y \geq 0 \end{cases}$$

and consequently $angle(x,y) = \frac{b}{\pi} \cdot 50$. A similar process is used for the angles at the end (a_{22}, a_{23}, a_{24}).
As done for $a_7..a_{12}$, we can express the relative position of the gesture's center of gravity with :

$$a_{25} = \frac{cgx-left}{width} \cdot 100, a_{26} = \frac{cgy-bottom}{height} \cdot 100,$$
$$a_{27} = \frac{cgz-back}{depth} \cdot 100$$

with $cgx = \frac{cx}{n}, cgy = \frac{cy}{n}, cgz = \frac{cz}{n}$. The additional processing of the orientation data increases the number of aspects from 28 to 56.

5.3. Search For Matching Gestures

Combined, the aspects form a product rule that classifies the gesture. For example :

IF
'is absolutely not straight' AND
'is more or less flat' AND
.....
THEN
'it might be a circle-like gesture'

An important feature of our fuzzy system (in contrast to others) is that we don't want the user to define rules that characterize motion-based gestures, but that we want the system to learn these rules in terms of differentiating between gestures and recognizing them. Thus the system must be able to automatically generate a new product rule every time it is taught a new gesture (or a new representation of an already learned gesture). Note that the gestures that must be recognized are not known in advance, thus a manually specification of product rules that describe them is not possible. It is important that the system evaluates these rules in a way that allow it to draw the right inference, depending on a possibly large number of generated rules. A rule is represented by a set of fuzzy values (i.e. the individual aspects that characterize a gesture) and is specified by a single representation of a gesture. Note that it is likely that more than one representation of the same gesture exists, in terms of being able to recognize it. To identify the rule that matches best, and so draw the right inference (i.e. recognize the gesture), we compare each fuzzy value (i.e. each aspect) of the current rule with the corresponding value of the already learned rules. The rule with the lowest deviation is the one with the highest probability for drawing the right inference.

Some fuzzy systems allow the user to specify the degree of faith in these rules. This is called approximate reasoning and is usually implemented by multiplying the inferred fuzzy values by predefined weights that represent the user's faith in particular rules (e.g. 0.8 represents high, 0.2 represents low). We apply approximate reasoning by weighting each aspect deviation to indicate its importance in terms of inferring the correct conclusion. Some of these weights can be set up manually by the user to indicate the general importance of the aspects, but most of the weights are calculated dynamically (e.g. depending on the appearance of extreme cases), because the rules (as well as an indication of the faith in these rules) are not known in advance, but are learned by the system.

After all aspects of the gesture to be identified have been calculated, we can compare them with the aspects of the already learned gestures to find the closest match. We use two simple methods to identify deviations or correspondences. One method can be chosen for each specific aspect. The first method simply calculates the absolute difference of two corresponding aspects, while the second one calculates the ratio of the smaller value to the larger value. We prefer the second method to compare aspects that express distance relations (such as in $a_0, a_{13}, a_{14}, a_{15}$, etc.), and the first method to compare aspects that express distances (e.g. the relative distance of two points to a face of their bounding box, i.e. a_7, a_8, a_9, etc.). Note that the results are normalized again to the range of $[0,100]$. A value of 0 indicates identity and a value of 100 indicates maximum deviation.

$$C_1(a_i, b_i) = |a_i - b_i|,$$
$$C_2(a_i, b_i) = 100 - \frac{min(a_i, b_i)}{max(a_i, b_i)} \cdot 100$$

The total deviation of two gestures can be represented as the weighted sum of all individual deviations :

$$td = \sum_{i=0}^{m-1} (w_i C_i(a_i, b_i))^2$$

where a_i, b_i are aspects that describe the same characteristic, w_i is the aspect specific weight that is used to implement approximate reasoning, and m the number of aspects of a product rule.

The recognized gesture is the one with the smallest total deviation among the already learned gestures. Note that the terms are squared to emphasize the effect of larger deviations and to reduce the effect of smaller deviations. If we normalize the total deviation to the range of $[0,100]$, we can compare it to a threshold value; thus we can derive that no gesture was recognized if the smallest total deviation is too high (i.e. above the threshold). The normalization can be done as follows :

$$td' = \sqrt{\sum_{i=0}^{m-1} \left(\frac{w_i^2}{\sum_{i=0}^{m-1} w_i^2} C_i(a_i,b_i)^2\right)}, 0 \leq td' \leq 100$$

Note, that normalization is not necessary to find the smallest total deviation and should not be done if it is not compared with a threshold. This increases the efficiency of the comparison process. Not only can the method of comparison be chosen for every individual aspect, the specific weights can also be set up in advance to indicate the aspect's importance and strength (the larger the weights the higher the strength of the aspect). For example, a_0 would have a large weight, because the peculiarity 'whether the gesture is straight or intricate' is an important information.

A major problem arises if extreme edge relations of the bounding box appear (e.g. an absolutely flat bounding box, etc.). If this happens, we cannot trust the significance of some aspects, thus we must apply approximate reasoning dynamically. To overcome this problem, we modify the weights of such critical aspects (e.g. a_8, a_9, a_{11}, a_{12}, etc.) during runtime, thus reducing their effect on the total result in such extreme cases.

If we split the weights w_i into its two components (dividend and divisor), we can update the corresponding dividend (while retaining its divisor) depending on the appearance of extreme cases. For normal cases, we assign larger values to the dividend and the corresponding aspect can be higher weighted. We decrease the dividend the more extreme the case becomes, in order to decrease the effect of the aspect on the total result. The information that indicates whether a case is extreme or normal (or something in between) can be derived from the bounding box information stored in the aspects a_1, a_2, a_3 (and a_4, a_5, a_6 for the rotated case). The basic idea is that the dividends (Y) can be calculated with a linear equation of the form :

$$Y = max(0, min(100, b + g(X - a)))$$

where g is the equation's gradient and $[a,b], (0 \leq a \leq 100, 0 \leq b \leq 100)$ the center of rotation (for varying gradients). These three parameters let us describe the appearance (or better, the degree) of extreme cases and how to weight derived aspects. Note that we must apply the original equation, or its reflection at $X = 50$, depending on the behavior of the bounding box aspects (X).
The original's reflection at $(X = 50)$ then is :

$$Y = max(0, min(100, b + g((100 - a) - X)))$$

with its center of rotation at $[(100 - a), b]$ (for varying gradients).

Experimentally, we choose $[a,b] = [20,20]$ with a gradient of $g = 3$. Thus, we can dynamically calculate new dividends for the critical aspects $a_7 - a_{18}$ and $a_{25} - a_{27}$ while the weights for all the other aspects remain constant :

$$\begin{aligned} Dividend_{8,11,14,26} &= max(0, min(100, 20 + 3(a_2 - 20))), \\ Dividend_{7,10,13,25} &= max(0, min(100, 20 + 3(80 - a_1))), \\ Dividend_{9,12,15,27} &= max(0, min(100, 20 + 3(a_3 - 20))) \end{aligned}$$

And similarly, for the rotated gesture. Note that in our example, the divisors of the critical aspects are 20 and that all weights range from 0 (no effect) to 5 (very strong aspect).

5.4. Teaching and Learning of Gestures

Learning new gestures is achieved by simply adding a new product rule (i.e. the specific set of aspects) to the knowledge-base. From then on, it is compared with other, new gestures. The same gesture can be stored several times (in different representations) to increase the reliability. The system can, for example, be trained by correcting it each time it fails to recognize a gesture. In this case, the aspects of the failed gesture can be added to the set again, and so extend the system's knowledge of different representations of the same gesture. The failure rate of the system decreases as it becomes more knowledgeable about the gestures (i.e. if many different representations of same gestures are stored in the knowledge base). Similar gestures (gestures that have similar aspect values) should be taught to the system in a similar way (i.e. same dimensions, etc.) to emphasize details. Because the claim of memory to represent the knowledge base is minimal, a scanned gesture can be recognized very fast (i.e. the comparison process is very effective), even if the knowledge base contains many different representations of same gestures. Each aspect ranges from 0 to 100, thus a 7-bit representation is sufficient. Multiplied by 56 (for 56 position/orientation aspects), a gesture representation has the uses less than 50 bytes (no matter how complex it is). Even smaller representations can be used, if smaller aspects are sufficient enough (e.g. for 2D or 3D recognition).

The recognized gestural information can–as in the presented work–be used to interact with virtual environments on an intuitive and natural basis (for a different application see Fig. 8). Context knowledge and other modalities, such as speech, might be used in addition to improve the user's interaction precision and articulation possibilities.

6. Conclusions and Future Work

We have presented gesture-based interaction for a system that uses transparent props–the pen and pad–for two-handed interaction with the Virtual Table, a desktop VR system. The system exploits the fact that the VT can display 3D graphics aligned with the props, thus turning them into multi-purpose tools. In this sense, transparent props seem to be a tool for the guiding person in a Surround-Screen Projection-Based Virtual Reality System (SSVR), whose viewpoint is tracked, and therefore in correct stereoscopic relation to the interface on the panel's surface. We consider such a configuration an interesting next step for our research.

The motion-based gesture recognizer we have developed employs Fuzzy Logic to achieve a predictable and flexible learning process with the high performance needed in real-time Virtual Environments.

Our system was informally tested with several users, most of which had computer (desktop) experience, but little experience with VR systems. They generally found our design very appealing and were able to perform simple design tasks after a few minutes of initial instruction. We did not observe any difficulties in understanding the tools. Complaints mainly addressed technical inadequacies like tracker error, lag, or frame rate.

Currently the object-creation gestures trigger the import of 'precanned' VRML objects into the OpenInventor-based scene. In order for the system to be used in a CAD environment realistically, its integration with ARCADE [15] is already being pursued. For the same reason, we must obviously expand the gesture space, i.e. the set of recognized gestures for design tasks.

A promising area of future work encompasses the combination of the presented gestures with speech input. Such multimodal interaction would not only increase the interface's intuitivity, but also extend the number of possible gestures to be unambiguously employed, and reduce the number of strokes needed to perform a certain gesture.

Acknowledgments

This work heavily employs the Studierstube/VT system [12] sponsored by the Fraunhofer CRCG Student and Scholar Exchange Program (SSEP) and the Austrian Science Foundation (FWF) under contract number P-12074-MAT. Special thanks to the Fraunhofer INI-GraphicsNet Virtual Table consortium and the Computer Graphics group at Vienna University of Technology for supporting the way to this collaborative research.

References

1. I. Angus and H. Sowizral. Embedding the 2D Interaction Metaphor in a Real 3D Virtual Environment. In *Proc. SPIE*, volume 2409, pages 282–293, 1995.
2. Barco, Inc. BARON. URL: http://www.barco.com/-projecti/products/bsp/baron.htm, 1997.
3. K. Böhm, W. Hübner, and K. Väähänen. GIVEN: Gesture Driven Interactions in Virtual Environments - A Toolkit Approach to 3D Interactions. *Interface to Real and Virtual Worlds-Conference,* Montpellier, 1992.
4. M. Billinghurst, S. Baldis, and M. Philips. 3D Palette: A Virtual Reality Content Creation Tool. In *Proc. ACM VRST'97,* pages 155–156, 1997.
5. S.S. Fels and G.E. Hinton. Glove-talk: A neuronal network interface between a data-glove and a speech synthesizer. *IEEE Trans. on Neuronal Networks,* vol. 4, pp.2-8, 1994.
6. Y. Guiard. Assymetric Division of Labor in Human Skilled Bimanual Action: Kinematic Chain as Model. *Journal of Motor Behaviour*, 16(4):486–517, 1987.
7. Homan. SmartScene: Digital Training - Learn the System by Being Part of the System. Available from URL: http://www.multigen.com, 1997.
8. W. Krüger, C.-A. Bohn, B. Fröhlich, H. Schüth, W. Strauß, and G. Wesche. The Responsive Workbench: A Virtual Work Environment. *IEEE Computer*, 28(7):42–48, July 1995.
9. G. Rigoll, A. Kosmala, and S. Eickeler. High Performance Real-Time Gesture Recognition Using Hidden Markov Models. *Proc. of International Gesture Workshop 1997*, Bielefeld, Springer, 1998.
10. E. Sachs, A. Roberts, and D. Stoops. 3-Draw: A Tool for Designing 3D Shapes. *IEEE Computer Graphics & Applications*, pages 18–26, 1991.
11. D. Schmalstieg, A. Fuhrmann, Zs. Szalavári, and M. Gervautz. "Studierstube" – An Environment for Collaboration in Augmented Reality. In *Proc. of Collaborative Virtual Environments '96, Nottingham, UK, and Virtual Reality Systems – Development and Applications, Vol. 3, No. 1*, pages 37–49, 1996.
12. D. Schmalstieg, L.M. Encarnação, and Zs. Szalavári. Using Transparent Props for Interaction with the Virtual Table. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics (I3DG'99), Atlanta, GA*. ACM SIGGRAPH, May 1999.
13. C.J. Sparell. Coverbal Iconic Gesture in Human-Computer Interaction. S.M. Thesis, *MIT Media Arts and Science,* 1993.
14. R. Stoakley, M. J. Conway, and R. Pausch. Virtual Reality on a WIM: Interactive Worlds in Miniature. In *Proc. Conference on Human Factors in Computing Systems (CHI'95)*, pages 265–272, 1995.
15. A. Stork and B. Anderson. 3D Interfaces in a Dis-

tributed Modelling Environment – 3D Devices, Interaction and Visualization Techniques. In W.D. Fellner, editor, *1995 Int. Workshop on Modeling, Virtual Worlds, and Distributed Graphics (MVD'95)*, pages 83–92. infix, 1995. URL: http://www.igd.fhg.de/~stork/papers/-mvd95/mvd95.html.

16. Zs. Szalavári and M. Gervautz. The Personal Interaction Panel – A Two Handed Interface for Augmented Reality. In *Computer Graphics Forum (Proceedings of EUROGRAPHICS'97)*, volume 16(3), pages 335–346. NCC Blackwell, 1997.

17. B. Ullmer and H. Ishii. The metaDESK: Models and Prototypes for Tangible User Interfaces. In *Proc. UIST'97, Banff, Alberta, Canada*, pages 223–232. ACM, 1997.

18. M. Wloka and E. Greenfield. The Virtual Tricoder: A Uniform Interface for Virtual Reality. In *Proc. ACM UIST'95*, pages 39–40, 1995.

19. L.A. Zadeh. Fuzzy Sets and Applications. John Wiley and Sons Publications, 1987.

20. R.C. Zeleznik, K.P. Herndon, and J.F. Hughes. SKETCH: An interface for sketching 3D scenes. *Computer Graphics*, 30(Annual Conference Series):163–170, 1996.

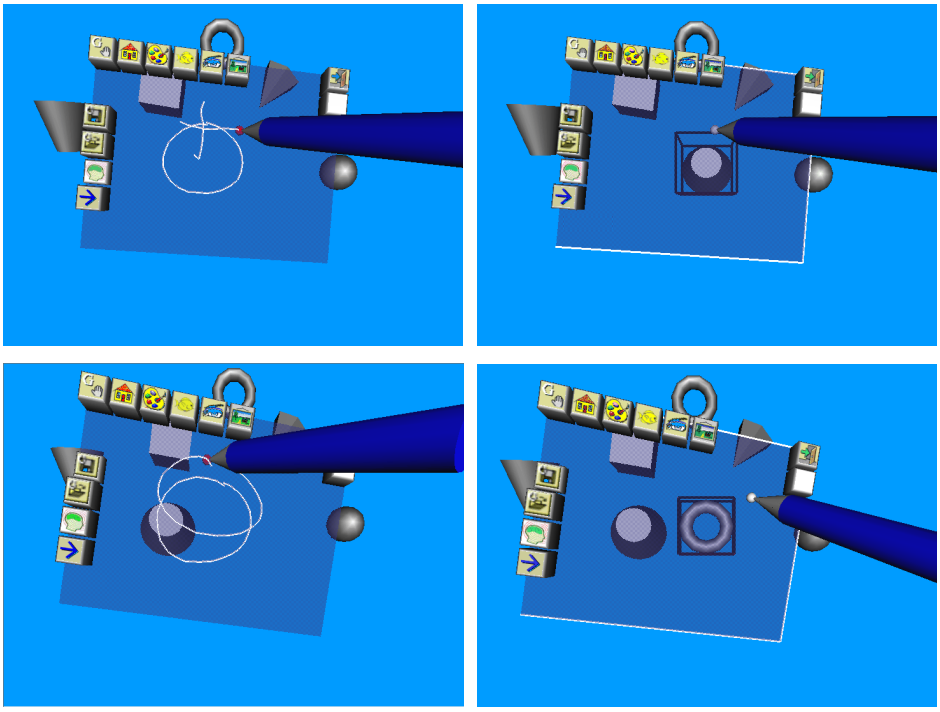

Figure 4: *Creating objects by using a motion-based gesture: a truncated cone (top) and a torus (bottom).*

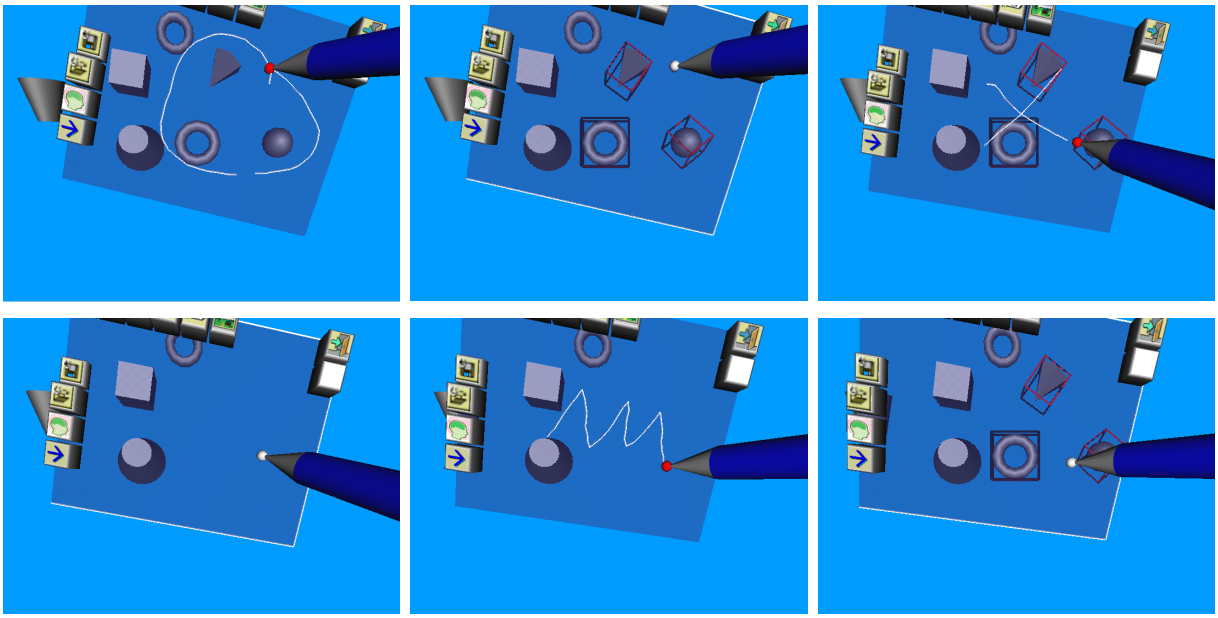

Figure 5: *Object control through motion-based gestures. From left to right: Selection by circling, deletion by crossing-out, and undoing the previous operation by scribbling on the pad.*

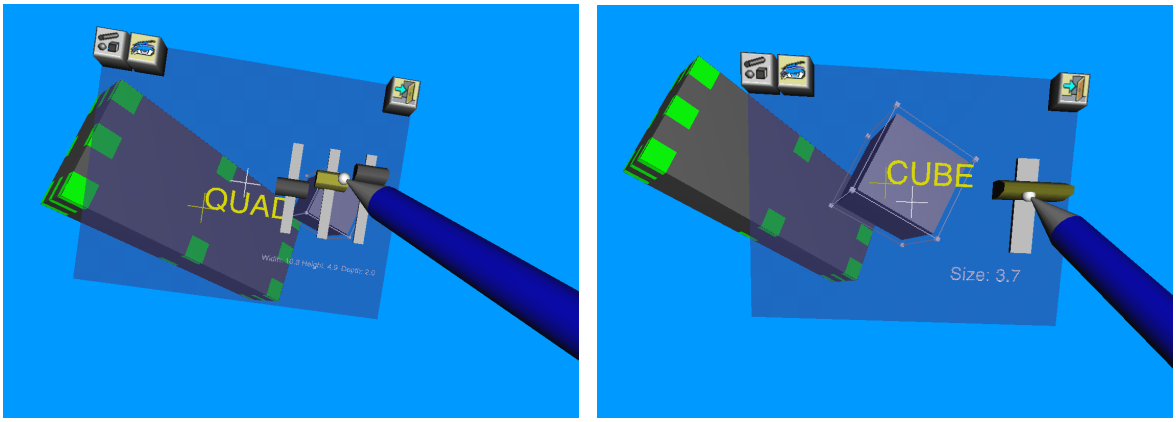

Figure 6: *Context-sensitive dialog elements and labeling on the transparent pad.*

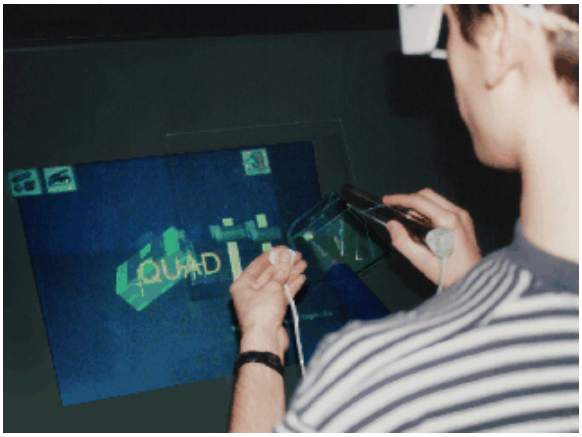

Figure 7: *Transparent pad and pad for interaction with the virtual table.*

Figure 8: *Intuitive motion-based gesture recognition: The players on the basketball field are animated by circular selection and directional strokes.*