CONTINUOUS 6DOF GESTURE RECOGNITION : A FUZZY LOGIC APPROACH

Oliver Bimber

Fraunhofer Institute for Computer Graphics
Joachim-Jungius-Strasse 11
18059 Rostock
Germany
obimber@egd.igd.fhg.de

ABSTRACT

The method described in this paper recognizes previously learned gestures which the system was taught by performing them.
Any kind of 6DOF input device (e.g. data glove, visual systems, space mouse etc.) 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 wrong recognition results. This extends the system's knowledge. Once learned, the system translates the recognized gestures into (for a computer) identifiable objects (numbers, strings, events, etc.) that can be further processed.
The system's advantages are its usability (e.g. with 2D or 3D input devices or in combination with finger status information, etc.), the minimum of information needed to recognize a gesture, and, consequently, the high speed of its scanning and comparison process.

Keywords : continuous gesture recognition, fuzzy logic, knowledge based, training, motion

INTRODUCTION

The application of gestural interfaces is a common method to interact with virtual environments. Human gestures can be used to navigate through the environment, interact with objects, and input commands that are further processed. Most systems use data gloves *[1][6]* or analyzed optical information of the human hand *[2][8]* to obtain the gestural signals expressed by the user. The evaluated data usually describes finger and hand status *[6][7]* (e.g. thumb up in combination with a rotation of the fist means rotating an object). Other body parts can also describe meaningful gestures (e.g. the arms, the head or even the whole body *[3]*).
To transform 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.). – Fuzzy Logic *[4]* can be a solution.

Fuzzy Logic is basically a multivalued logic that allows intermediate values to be defined between conventional evaluations like *yes/no*, *true/false*, *0/1*, etc. Notions like *rather straight*, *pretty high* or *relatively far left* can be formulated mathematically and processed by computers. In this way an attempt is made to apply a more human-like way of thinking in the programming of computers, and complex structures can easily be described.
Fuzzy Logic was initiated in 1965 by **Lotfi A. Zadeh** *[5]*, professor for computer science at the University of California in Berkley.
The method described in this paper recognizes previously learned gestures which the system was taught by performing them.
Any kind of 6DOF input device (e.g. data glove, visual systems, space mouse etc.) 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 wrong recognition results. This

extends the system's knowledge. Once learned, the system translates the recognized gestures into (for a computer) identifiable objects (numbers, strings, events, etc.) that can be further processed.
The system's advantages are its usability (e.g. with 2D or 3D input devices or in combination with finger status information, etc.), the minimum of information needed to recognize a gesture, and, consequently, the high speed of its scanning and comparison process.

Three processes are involved in the recognition of gestures :

A **scanning process** tracks the motion of the 6DOF input device and updates basic information (e.g. bounding box, length, orientation etc.) on the fly.
The **comparison process** calculates gesture specific relations that describe the gesture's behavior in a Fuzzy Logic common way. These relations are used as aspects to compare the new input with the already learned gestures and to recommend one that matches best.
New gestures can be learned by the **learning process** to extend the system's knowledge.

Because of their similarity, we will only discuss the concepts based on the gesture's position data. The system also takes the gesture's orientation information (e.g. rotation of the hand) into account. An exact explanation of how to process this information would be redundant, because they are treated exactly as the position data. We will discuss the required extensions in sections below.

THE SCANNING PROCESS

Input for the scanning process is the motion data of the 6DOF input device that carries out the gesture and can be represented as a set of position/orientation samples within the three-dimensional space. Below, we will discuss the processing of the position data only :

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

for a gesture G described with n position samples.
Receiving these positions from the input device, the process updates the before initialized basic information :

- *The gesture's bounding box :*

$$\begin{aligned} &left = \min(x_i, left), right = \max(x_i, right), \\ &bottom = \min(y_i, bottom), top = \max(y_i, top), \\ &back = \min(z_i, back), front = \max(z_i, front) \end{aligned}$$

- *The total length of the gesture :*

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

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 :*

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

- *The sum of all points (x,y,z components) to calculate the center of gravity :*

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

- *The angle-coefficient (legs of a triangle) to calculate the start and end angles of the gesture :*

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

were 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 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 from taking the distortion into account that usually appears at the beginning and the end. The exponent p is used to control how fast the weights increase or decrease (with respect to the length of the gesture). We use $p = 2$ in our examples.

To realize a higher reliability we do not use the original motion information only. We also rotate the original positions around the x,y and z axis 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.

We then obtain $hmR, vmR, dmR, saxR, sayR,$ $sazR, eaxR, eayR, eazR$, and a new bounding box ($leftR, rightR, bottomR, topR, backR, frontR$) for the rotated values. 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 $c = \sin(45°) = \cos(45°) = 0.7071067$. The rotated positions of the original gesture can then be calculated as follows :

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

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

THE COMPARISON PROCESS

After the scanning process has collected all the basic information about the expressed gesture, the comparison process calculates the gesture specific information that describes its behavior. These properties (calculated based on the basic information) are used as aspects for the comparison process to find the gesture that matches best out of the already learned set.

Fuzzy Logic can help to find (and evaluate) strict aspects which describe a gesture's behavior good enough to identify it.

Statements like : Is the gesture more straight or more intricate ?, In which direction does it start ?, Where does it end ?, What are the gestures dimensions ? etc., can be formulated mathematically, using Fuzzy Logic.

In this example we use 28 (later 56) different aspects to describe the gesture's behavior. Note, that we will discuss the processing of the position information only :

A[0]: *Is the gesture more straight or more intricate ?*

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. Note, that as we work with integer numbers (to reach a higher performance) we multiply the result by 100 to ensure a higher precision. We continue doing this for all other aspects.

A result of around 100 indicates that the gesture describes a straight line (the smaller the value, the more intricate and complex the gesture).

A[1], A[2], A[3]: *Dimensions of the bounding box.*

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

$$A[1] = \frac{height}{height + witdh} \cdot 100$$

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.

We calculate similar ratios to express the height/depth ratio and the depth/width ratio :

$$A[2] = \frac{height}{height + depth} \cdot 100, A[3] = \frac{depth}{depth + witdh} \cdot 100$$

A[4], A[5], A[6]: *Bounding box dimensions of the rotated gesture.*

We apply the idea mentioned above again to derive similar ratios for the bounding box dimensions of the rotated gesture :

$$A[4] = \frac{heightR}{heightR + witdhR} \cdot 100,$$
$$A[5] = \frac{heightR}{heightR + depthR} \cdot 100,$$
$$A[6] = \frac{depthR}{depthR + witdhR} \cdot 100$$

with

$$widthR = rightR - leftR, heightR = topR - bottomR, depthR = frontR - backR$$

A[7], A[8], A[9], A[10], A[11], A[12]: *Relative start and end positions.*

These aspects express, that the gesture begins relatively close (or relatively far away from) to the right (top, front) lateral face of its bounding box :

$$A[7] = \frac{x_0 - left}{width} \cdot 100,$$
$$A[8] = \frac{y_0 - bottom}{height} \cdot 100,$$
$$A[9] = \frac{z_0 - back}{depth} \cdot 100$$

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 the gesture begins at the corresponding opposite sides (left, bottom, back) of the bounding box.
Similar ratios can be formulated for the end position :

$$A[10] = \frac{x_{n-1} - left}{width} \cdot 100,$$
$$A[11] = \frac{y_{n-1} - bottom}{height} \cdot 100,$$
$$A[12] = \frac{z_{n-1} - back}{depth} \cdot 100$$

A[13], A[14], A[15]: *Movements.*

The sum of all horizontal movements is at least as big 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 vertical and the depth movements :

$$A[14] = \frac{height}{vm} \cdot 100, A[15] = \frac{depth}{dm} \cdot 100$$

Using these aspects helps to determine the course of the gesture.

A[16], A[17], A[18]: *Movements of the rotated gesture (diagonal movements).*

We apply the idea mentioned above again to derive similar ratios for horizontal/vertical/depth movements of the rotated gesture (i.e. in diagonal direction) :

$$A[16] = \frac{widthR}{hmR} \cdot 100,$$
$$A[17] = \frac{heightR}{vmR} \cdot 100,$$
$$A[18] = \frac{depthR}{dmR} \cdot 100$$

A[19], A[20], A[21], A[22], A[23], A[24]: *Start and end angle.*

To determine in which direction the gesture starts and in which it ends we calculate the start and end angles using the (during the scanning process tracked) horizontal/vertical and depth angle-coefficients which represent the gesture's direction (x,y,z components) at the beginning and the end. 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 sufficient 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.

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 $a = a\cos\left(\frac{x}{\sqrt{x^2 + y^2}}\right)$ to the range of 0..100 with :

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

and consequently

$$angle(x, y) = \frac{b}{\pi} \cdot 50$$

And the angles at the end :

$$A[22] = angle(eax, eay),$$
$$A[23] = angle(eax, eaz),$$
$$A[24] = angle(eaz, eay)$$

A[25], A[26], A[27]: *Relative positions of the center of gravity.*

Similar to A[7][8][8][10][11][12], these aspects express that the gesture's center of gravity (*cg*) is located relatively close to the right (top, front) lateral face of its bounding box :

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

with $cgx = \frac{cx}{n}, cgy = \frac{cy}{n}, cgz = \frac{cz}{n}$, where n is the number of sample points and *cx,cy cz* are the sums of the point's components, calculated during the scanning process.

Values close to 100 indicate that the gesture's *cg* is located close to the specific face (right, top, front). Values of around 50 indicate that the *cg* is located at the center; and values close to 0 indicate that it is located at the opposite sides (left, bottom, back) of the bounding box.

After all aspects are calculated, we can compare them with the aspects of the already learned gestures to find the one that matches best.

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 method one to compare aspects that express distances itself (e.g. the relative distance of two points to a face of their bounding box, i.e. A[7][8][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.

$$C_1(A_1[i], A_2[i]) = | \quad A_1[i] - A_2[i] \quad |,$$
$$C_2(A_1[i], A_2[i]) = 100 - \frac{\min(A_1[i], A_2[i])}{\max(A_1[i], A_2[i])} \cdot 100$$

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

$$td = \sum_{i=0}^{m-1} (w_i \cdot C_i(A_1[i], A_2[i]))^2,$$

where w_i is the aspect specific weight and m the number of aspects (28 *(56)* in our example).

The recognized gesture is the gesture 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.

Not only the method of comparison (C_1, C_2) can be chosen for every individual aspect; also the specific weights can be set up in advance to indicate the aspect's importance and strength (the larger the weights the higher the strength of the aspect). A[0] would have a large weight, because the peculiarity 'whether the gesture is straight or intricate' is an important property.

The major problems of the comparison process are the extreme relations of the bounding box's lateral faces. An example :

Imagine a gesture that describes an exactly straight, horizontal line on the x/y-plane. The learning process would learn about this gesture and store its aspects within the knowledge base. Because of the dimensions of its bounding box, the aspects A[8][9][11][12] (for example) indicate that the gesture begins and ends at the back/bottom face of the bounding box. Because the bounding box has no depth and no height, it could also begin and end at the front/top (or back/top, front/bottom) face. However, the learning process learns about the gesture this way (back/bottom).

If we then repeated the same gesture, the comparison process probably would not recognize it, because it is very unlikely that the repeated gesture is an exactly straight line again. Start and end point might not be exactly on the same y/z level, but the same gesture is supposed to be represented. The comparison process might find out (for example) that the new gesture begins at the front/top and ends at the back/bottom face of the bounding box. This does not match with the already learned gesture and the comparison process returns (most likely) a wrong result.

To overcome this problem we modify the weights of such critical aspects (e.g. A[8][9][11][12], etc.) while runtime, and so reduce their effect on the total result in such extreme cases.

If we split the weights 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 gets to decrease the effect of the aspect on the total result. The information 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][2][3] (and A[4][5][6] for the rotated case).
Thus, we can dynamically calculate new dividends for the aspects A[7]-A[18] and A[25]-A[27] while the weights for all the other aspects stay constant :

$$\begin{aligned} Dividend[8][11][14][26] &= \max(0, \min(100, 20 + (A[2] - 20) \cdot 3)), \\ Dividend[7][10][13][25] &= \max(0, \min(100, 20 + (80 - A[1]) \cdot 3)), \\ Dividend[9][12][15][27] &= \max(0, \min(100, 20 + (A[3] - 20) \cdot 3)) \end{aligned}$$

And for the rotated gesture :

$$\begin{aligned} Dividend[17] &= \max(0, \min(100, 20 + (A[5] - 20) \cdot 3)), \\ Dividend[16] &= \max(0, \min(100, 20 + (80 - A[4]) \cdot 3)), \\ Dividend[18] &= \max(0, \min(100, 20 + (A[6] - 20) \cdot 3)) \end{aligned}$$

Note, that in our example the divisors of the critical aspects (A[7]-A[18]) are 20 and that all weights range from 0 (no effect) to 5 (very strong aspect).

ORIENTATION

Because of its similarity, the orientation data (azimuth, elevation, and roll alignments of the input device) can be treated as the position data (as described above). Twice as many basic information has to be tracked during the scanning process, and the number of comparison aspects that are calculated during the comparison process has to be doubled (i.e. 56 instead of 28).
The difference between the orientation and the position data is their value range. While the value range of the positions can be expressed as distances (e.g. mm, inch, etc.), the value range of the orientations can be expressed as angles (e.g. degree or radial). This difference, however, plays no role and can be ignored as we apply the Fuzzy Logic concepts and formulate relative statements. Thus, we are able to treat position and orientation data the same way.
To take a gesture’s orientation information into account (beside the position data), makes it possible to recognize a much wider range of gestures. For example, gestures that are performed by the wrist or the head can be recognized as well.

THE LEARNING PROCESS

Once a gesture is scanned and its aspects are calculated, it can be taught by simply adding it to the knowledge base (set of gesture with their aspects). From then on it will be compared with other, new gestures.
The same gesture can be stored in the knowledge base 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 about different representations of the same gesture. The failure rate of the system decreases the more it knows 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 enough. Multiplied by 56 (for 56 position/orientation aspects), a gesture representation has the size of less than 50 bytes (no matter how complex it is). Even smaller representations can be used, if less aspects are sufficient enough (e.g. for 2D or 3D recognition).

APPLICATIONS AND FUTURE RESEARCH

The method described in this paper can be used in all possible VR scenarios where 2D,3D or 6DOF tracking is possible. Even head tracking can be used to recognize gestures that are expressed by the head (e.g. nodding).
The optimal scenarios for this method make use of large VR environments, such as caves or virtual tables, etc., that offer a wider interaction radius.
The ability to learn gestures and to improve its knowledge by correcting it, makes the system extremely flexible and offers a large range of applications.
The method can also be used with two dimensions only by assigning 0 to the other dimensions. On a virtual table for example, the projected x/y coordinates can be used to recognize gestures that draw symbols or even characters onto the table. Pen-like input devices are imaginable.
Finger gestures can also be used in combination with this approach.

A Series of gestures could be recognized by identifying the individual gestures and ”feeding” their IDs as tokens into a parser (e.g. yacc or bison). Grammars could be developed that help to understand the specific sign-language.
To support two handled scenarios (for a more realistic gestural interaction) would be a simple extension.
Primarily, it should be done research to find stricter aspects that might extend the method by finger gestures or to increase its reliability.
Possible application scenarios should be found within the human computer interaction and the VR area; as well as appropriate gestures.
In combination with speech recognition for example, gesturing could be used to explain actions more exact. Such an interface would allow us to use gestures to emphasize important information, to give commands like ”*stop*”, ”*come here*”, ”*turn around*”, or communicate a more exact description of an object’s size or shape.

The method was demonstrated on a virtual table in combination with a 6DOF magnetic tracker. Six gestures that described different semicircles were defined and used to rotate a cube around the x,y,z-axis. Two other gestures were used to make the cube appear and disappear, and a last one to stop rotating the cube. In this case, it should be extremely difficult for the system to interpret the different rotation gestures correctly, because all of them describe semicircles (just in different alignments). Strong distortion of the tracking device (especially further away from the transmitter) made it even more difficult.
Initially, after each gesture was trained exactly once, the system’s failure rate was approximately 15%, even for users who weren’t involved in the training. After a short training phase (a couple of minutes), the system’s failure rate was below 1%.

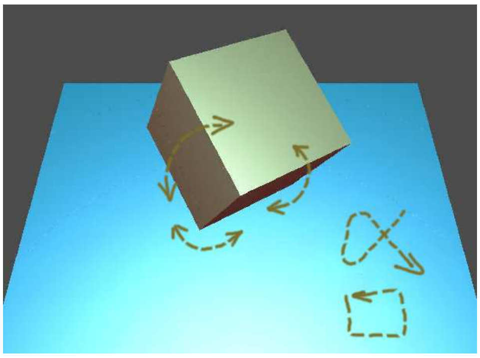

Fig. 1 : Illustration of the learned gesture-set.

The illustrated ‘X’-like gesture is used to make the cube disappear, and the square-like gesture to make it appear. The eight semicircle-like gestures indicate rotation in the specific direction. Note, that the gesture which is used to stop the rotation isn’t illustrated since it was realized as an absolutely motionless gesture (i.e. it would appear as a single point in the space).
Further test scenarios applied gesturebases with approximately 30 trained representations per learned gesture and also achieved high recognition rates.
Of course, the recognition rate strongly depends on the number and type of the learned gestures, the quality of the gesturebase and on the used hardware.

REFERENCES

[1] **Virtual Technologies, Inc.**, http://www.virtex.com/

[2] **Fraunhofer Institute for Computer Graphics, Darmstadt,** http://www.igd.fhg.de/www/igd-a7/Projects/Biosignals/gesture.html

[3] **University of Maryland**, Institute for Advanced Computer Studies, http://www.umiacs.umd.edu/~gavrila/abstract.html

[4] **H.-J. Zimmerman**, *Fuzzy Set Theory – and its Applications*, 2nd edition, Kluwer-Nijhoff Publishing, 1990

[5] **University of California (Berkley)**, Lotfi A. Zadeh, *Fuzzy Sets and Applications*, John Wiley & Sons Publications, 1987

[6] **Fraunhofer Institute for Computer Graphics, Darmstadt,** Kaisa Väähänen, Klaus Böhm, *Gesture Driven Interaction as a Human Factor in Virtual Environments – An Approach with Neural Networks*, Virtual Reality Systems Conf., British Computer Society, 1992, ACADEMIC PRESS 1993

[7] **Fraunhofer Institute for Computer Graphics, Darmstadt,** Klaus Böhm, Wolfgang Hübner, Kaisa Väähänen, *GIVEN: Gesture Driven Interactions in Virtual Environments - A Toolkit Approach to 3D Interactions*, Interface to Real and Virtual Worlds-Conference, Montpellier, France, 1992

[8] **Computer Graphics Center, Darmstadt,** http://www.zgdv.de/www/zgdv-vc/projects/given/