# Natural User Interfaces in Volume Visualisation Using Microsoft Kinect

Anastassia Angelopoulou[1], José García-Rodríguez[2], Alexandra Psarrou[1], Markos Mentzelopoulos[1], Bharat Reddy[1], Sergio Orts-Escolano[2], Jose Antonio Serra[2], and Andrew Lewis[3]

[1] Dept. of Computer Science and Software Engineering, University of Westminster, UK
{agelopa,psarroa,mentzem,bharat.reddy}@wmin.ac.uk

[2] Dept. of Computing Technology, University of Alicante, Spain
{jgarcia,sorts,jserra}@dtic.ua.es

[3] Dept. of Engineering and Information Technology, University of Griffith, Australia
a.lewis@griffith.edu.au

**Abstract.** This paper presents the integration of human-machine interaction technologies within a virtual reality environment to allow for real-time manipulation of $3D$ objects using different gestures. We demonstrate our approach by developing a fully operational, natural user interface (NUI) system, which provides a front-end framework for back-end applications that use more traditional forms of input, such as wear cable sensors attached to the users. The implementation is a user-friendly system that has immense potential in a number of fields, especially in the medical sciences where it would be possible to increase the productivity of surgeons by providing them with easy access to relevant MRI scans.

**Keywords:** Growing Neural Gas, 3D Sensors, Natural User Interfaces, Volume Visualisation.

## 1 Introduction

Volume visualization is an important form of scientific visualisation, allowing investigation of, for example, medical scanned data such as CT and MRI data, seismic survey data, and computational fluid dynamic (CFD) data [3,10]. To better understand volumetric datasets, people use computer hardware and software to manipulate the data and generate 2D projections for viewing. Much research on volume visualisation has been focused on volume rendering (how to render larger sets of data faster with a higher level of realism) or transfer function generation (how to highlight the regions of interest). To help improve the efficiency and efficacy of volume visualisation, one can integrate virtual reality environments (VEs) and human computer interaction (HCI) technologies in volume visualisation applications. However, these volume visualisation systems require accurate tracking of posture and movement which is provided by wear cable sensors attached to the users. Over the last decades there has been an increasing interest in using neural networks and computer vision techniques to allow users to directly explore and manipulate objects in a more natural and intuitive environment without the use of electromagnetic tracking systems. With the recent rise of motion sensing cameras, most notably Microsofts Kinect, gesture recognition has added an extra dimension to human-machine interaction [9].

This paper presents a virtual reality visualisation system (VirtVis) that has been designed and developed to use various virtual tools that allow users to directly explore and manipulate the volume data in $3D$ space. Many innovations have been integrated into this system, including an optimisation of the hand using the GNG network, an intuitive HCI paradigm tailored for volume visualisation in VEs, and geometric tools that can assist users to fully reveal the internal structure of volumetric datasets. Usability experiments have demonstrated that volume visualisation tasks can be performed significant better in virtual reality viewing conditions, and that using these geometric tools can significantly improve the efficiency and efficacy of the volume visualisation process. The HCI interaction used in VirtVis is based on natural and intuitive hand gestures. To manipulate a virtual object, the user physically reaches to grasp and move it as though it was real.

The remainder of the paper is organised as follows. Section 2 gives a theoretical background over sensor-free systems and our choice of selection. Section 3 discusses the implementation of the proposed system, before we conclude in Section 4.

## 2 Sensor-Free Systems

Human gestures form an integral part in our verbal and non-verbal communication. We use them to reinforce meaning not always conveyed through speech, to describe the shape of objects, to play games, to communicate in noisy environments, and to convey meaning to elderly people and people with special needs. We can use gestures as expressive body motions or to translate non-verbal languages that consist of a set of well defined gestures and hand postures with complete lexical and grammatical specifications as in the case of sign languages.

Hand gestures, which are effectively a $2D$ projection of a $3D$ object, can become very complex for any recognition system. Systems that follow a model-based method [1,13], require an accurate $3D$ model that captures efficiently the hand's high Degrees of Freedom (DOF) articulation and elasticity. The main drawback of this method is that it requires massive calculations which makes it unrealistic for real-time implementation. Since this method is too complicated to implement, the most widespread alternative is the feature-based method [7] where features such as the geometric properties of the hand can be analysed using either Neural Networks (NN) [14,16] or stochastic models such as Hidden Markov Models (HMMs) [4,15].

We decided to use the former for the representation of human gestures since our model should perform at high computational efficiency making it ideal for real time environments, have low quantisation error, and allow for efficient transformation of the objects. More specifically, we have used the GNG model since is superior in terms of computational efficiency, is robust against noise, and can handle complex distributions [2,12,13]. As for a sensor-free hardware platform which will allow the user to move freely and naturally in any environment we decided to use Microsoft Kinect since it combines an RGB camera, infrared depth sensor and multiarray microphone with a proprietary layer of software that allows human body and voice recognition.

## 2.1 Growing Neural Gas (GNG)

GNG [6] is an unsupervised incremental self-organising network independent of the topology of the input distribution or space. It uses a growth mechanism inherited from the Growth Cell Structure [5] together with the Competitive Hebbian Learning (CHL) rule [8] to construct a network of the input date set. In some cases the probability distribution of the input data set is discrete and is given by the characteristic function $\xi_w : \mathbb{R}^q \rightarrow \{0, 1\}$ with $\xi_w$ defined by

$$\xi_w = \begin{cases} 1 \text{ if } \xi \in W \\ 0 \text{ if } \xi \in W^c \end{cases} \tag{1}$$

In the network $\xi_w$ represents the random input signal generated from the set $W \subseteq \mathbb{R}^q$ and $W^c$ is the complement of $W \in \mathbb{R}^q$. The growing process starts with two nodes, and new nodes are incrementally inserted until a predefined conditioned is satisfied, such as the maximum number of nodes or available time. During the learning process local error measures are gathered to determine where to insert new nodes. New nodes are inserted near the node with the highest accumulated error and new connections between the winner node and its topological neighbours are created.

The GNG algorithm consists of the following:

- A set $A$ of cluster centres known as nodes. Each node $c \in N$ has its associated reference vector $\{x_c\}_{c=1}^{N} \in \mathbb{R}^q$. The reference vectors indicate the nodes' position or *receptive field centre* in the input distribution. The nodes move towards the input distribution by adapting their position to the input's geometry using a winner take all mapping.
- Local accumulated error measurements and insertion of nodes. Each node $c \in N$ with its associated reference vector $\{x_c\}_{c=1}^{N} \in \mathbb{R}^q$ has an error variable $E_{x_c}$ which is updated at every iteration according to:

$$\Delta E_{x_\nu} = \|\xi_w - x_\nu\|^2 \tag{2}$$

The local accumulated error is a statistical measure and is used for the insertion and the distribution of new nodes. Nodes with larger errors will cover greater area of the input probability distribution, since their distance from the generated signal is updated by the squared distance. Knowing where the error is large, if the number of the associated reference vectors belonging to the input space is an integer multiple of a parameter $\lambda$, a new node $x_r$ is inserted halfway between the node with the largest local accumulated error $x_q$ and its neighbour $x_f$.

$$x_r = \frac{x_q + x_f}{2} \tag{3}$$

All connections are updated and local errors are decreased by:

$$\Delta E_{x_q} = -\alpha E_{x_q} \tag{4}$$

$$\Delta E_{x_f} = -\alpha E_{x_f} \tag{5}$$

A global decrease according to:

$$\Delta E_{x_c} = -\beta E_{x_c} \tag{6}$$

is performed to all local errors by a constant $\beta$. This is important since new errors will gain greater influence in the network resulting in a better representation of the topology.

– A set $C$ of edges (connections) between pair of nodes. These connections are not weighted and its purpose is to define the topological structure. The edges are determined using the competitive hebbian learning method. The updating rule of the algorithm is expressed as:

$$\Delta x_\nu = \epsilon_x(\xi_w - x_\nu) \tag{7}$$

$$\Delta x_c = \epsilon_n(\xi_w - x_c), \forall c \in N \tag{8}$$

where $\epsilon_x$ and $\epsilon_n$ represent the constant learning rates for the winner node $x_\nu$ and its topological neighbours $x_c$. An *edge aging scheme* is used to remove connections that are invalid due to the activation of the node during the adaptation process.

## 2.2 3D Hand Representation with GNG

Figure 1 shows the ability of GNG to preserve the input data topology. Identifying the points of the input data that belong to the objects allows the network to adapt its structure to this input subspace, obtaining an induced Delaunay triangulation of the object. GNG has been adapted using the Point Cloud Library (PCL)[1] for the 3D surface representation. The main difference with the original GNG algorithm is the omission of insertion/deletion actions after the first frame. Since no neurons are added the system keeps the correspondence during the whole sequence, solving intrinsically the problem of correspondence. This adaptive method is also able to face real-time constraints, because the number $\lambda$ of times that the internal loop is performed can be chosen according to the time available between two successive frames that depend on the acquisition rate. The mean time to obtain a GNG on a frame is about 10ms., using the adaptive method. Thus, GNG provides a reduction of the input data, while preserving its structure.

[1] The Point Cloud Library (or PCL) is a large scale, open project [11] for 2D/3D image and point cloud processing.

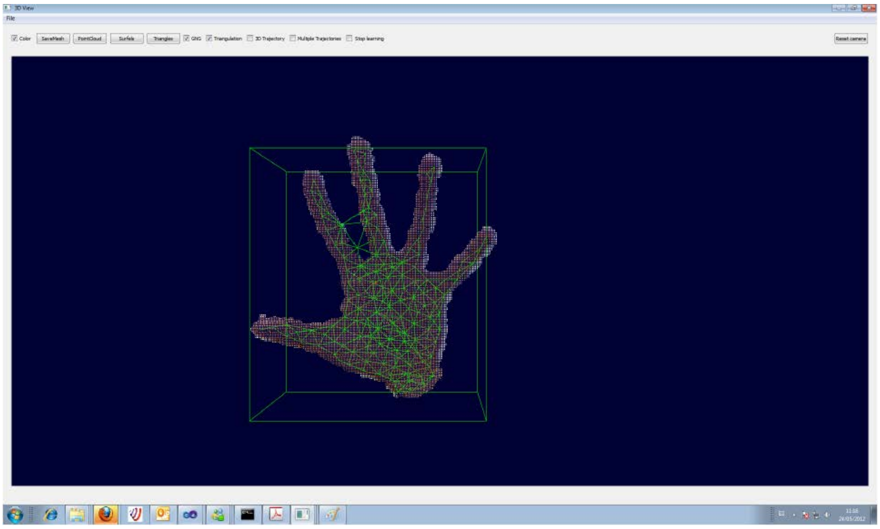


**Fig. 1.** 3D Hand representation with GNG

### 2.3 Microsoft Kinect

The Microsoft Kinect, is an accessory for Microsofts popular Xbox 360 gaming console that removes or reduces the need for a controller by enabling motion tracking in the Xbox 360, and allowing users to use their hands and bodies to play games that can receive this information. All the experimental phase is based on the use of real-world sequences obtained by the Kinect sensor. Such sensors belong to the so-called RGB-D cameras since they provide RGB format images with depth information per pixel. Specifically, Microsoft's Kinect sensor is able to get screenshots of 640x480 pixels and its corresponding depth information, based on an infrared projector combined with a CMOS sensor with a resolution of 320x240 pixels, and can reach rates of up to 30 frames per second. A first processing of sensor data enables obtaining the component in the z axis of coordinates of the points in the three dimensional space.

For the segmentation of the hands from the background, a hybrid technique based on depth information and skin colour has been used. A modification of the Point Cloud Library (PCL) for the 3D surface representation of the objects was used to support the 3D mesh reconstruction of the points based on the GNG algorithm discussed in Section 2. In the past, point clouds have mainly been created using 3D scanners. However, with the advent of technology such as Kinect it is now possible to acquire a point cloud for an object by using the depth sensor functionality. The construction of the VirtVis system is given in Figure 2. The main functionalities of the system as can be seen in the flowchart (User Modifies Object) are a menu system displayed on the screen, where the user can select an option by putting their hand mark over the option and holding for a few seconds (e.g. reset, change object, fly-through), and three indicators that show to the user where their hands and head are in relation to the virtual environment (Track User). This provides constant feedback to the user about the positioning of their hands.

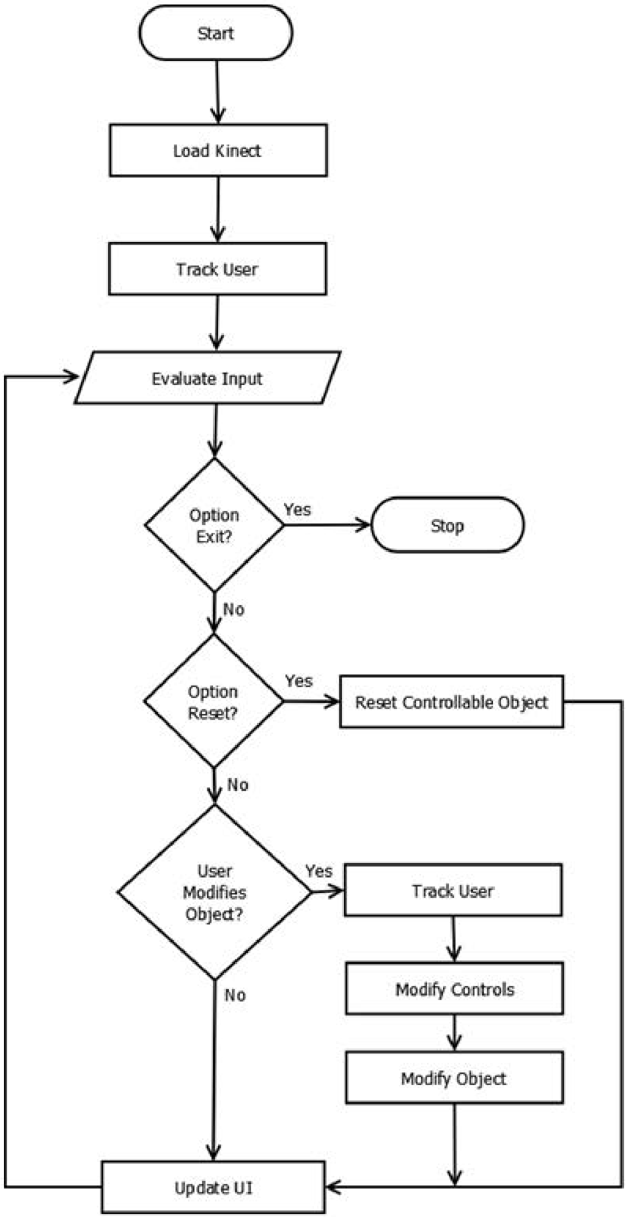


**Fig. 2.** Flowchart of the whole system

## 3 Experiments

Several experiments have been performed for the validation of our system. To carry out all the experiments we used the Kinect SDK which provides a Runtime class in C# that, once initialised, acts as an interface between the Kinect and our system. The Runtime is initialised with the first active Kinect system that the program finds, and is initialised with a number of Runtime options that allow skeletal tracking, video display, and multi-player interfaces. Skeleton Tracking is done via the Kinect Skeleton Engine, which provides, through the SkeletonFrameReady event, positions for up to two skeletons, and various joints on this skeleton. These joints positions are given according to the Kinects field of view, and are converted from our system to screen sized dimensions to accurately represent control of objects on screen.

The depth frame is constantly displayed in the top left hand corner of the screen to provide a reference to the user of what the Kinect is picking up so they can better position themselves for a good user interaction experience. Figure 3 shows the VirtVis main window. The system is based around the users head position, represented as an area that begins at 20% of the distance from the Kinect to the user, and ends at 80% of the distance. If one or both of the users hands are in this box, they are defined as active, and will do different things based on which hands are active. For example, when both hands are active, the user has activated the objects scaling functionality. Object scaling is calculated by taking the vector distance between the left and right hands, as well as the vector distance between the left and right shoulders. Shoulder joints are calculated to make the scaling functionality more intuitive by ensuring that if the hands are inside the shoulders, the object is smaller and when outside the object is bigger.

A modified version of PCL is used to save a reconstructed surface representation of the hand performing a gesture. Once the user has been tracked a timer is started. The user is notified that they have three seconds to get their right hand into position for the point cloud to be taken. After these three seconds the

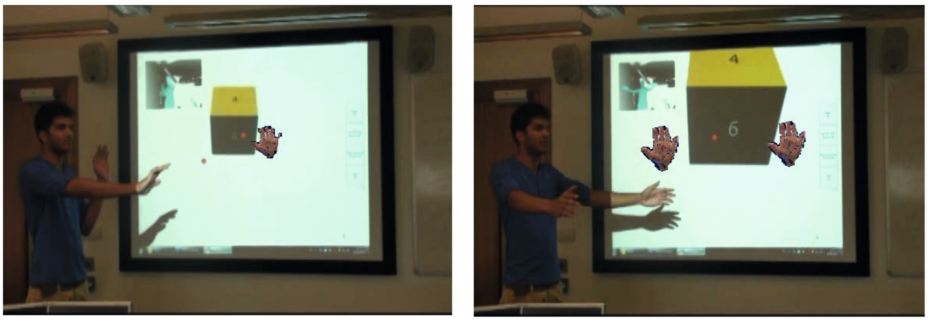
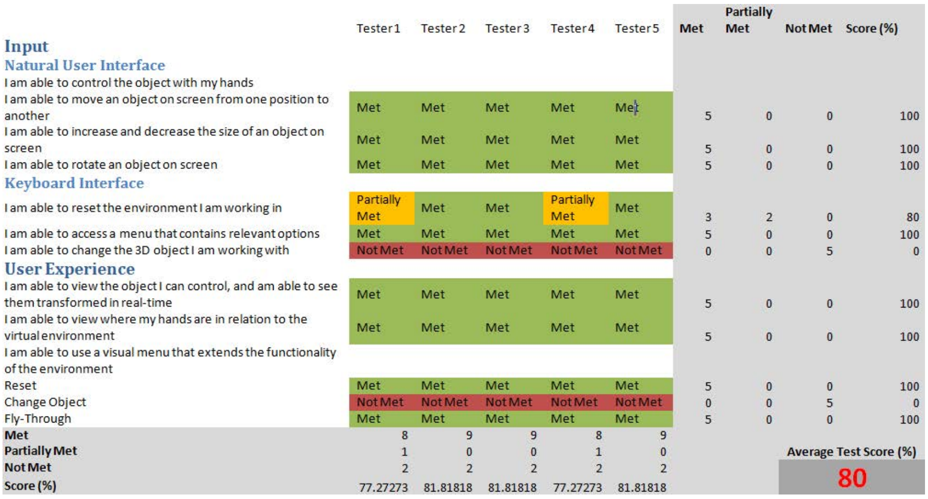


**Fig. 3.** An example of a cube object being transformed using different gestures. Two levels of fly-through functionality have been implemented similar to what is used in the volume visualisation systems.

functionality within PCL takes the point cloud of the right hand and reconstructs the surface using the GNG algorithm discussed in section 2.

Usability testing has been conducted with the User Acceptance Testing (UAT). UAT is common in agile methodology where tests are written as short one-stop requirements where testers are asked to indicate whether a requirement was met, partially met, or not met at all. Figure 4 shows the results of the user experience testing. There was found to be some ambivalence amongst users about whether



| | Tester1 | Tester2 | Tester3 | Tester4 | Tester5 | Met | Partially Met | Not Met | Score (%) |
|---|---|---|---|---|---|---|---|---|---|
| **Input** | | | | | | | | | |
| **Natural User Interface** | | | | | | | | | |
| I am able to control the object with my hands | | | | | | | | | |
| I am able to move an object on screen from one position to another | Met | Met | Met | Met | Met | 5 | 0 | 0 | 100 |
| I am able to increase and decrease the size of an object on screen | Met | Met | Met | Met | Met | 5 | 0 | 0 | 100 |
| I am able to rotate an object on screen | Met | Met | Met | Met | Met | 5 | 0 | 0 | 100 |
| **Keyboard Interface** | | | | | | | | | |
| I am able to reset the environment I am working in | Partially Met | Met | Met | Partially Met | Met | 3 | 2 | 0 | 80 |
| I am able to access a menu that contains relevant options | Met | Met | Met | Met | Met | 5 | 0 | 0 | 100 |
| I am able to change the 3D object I am working with | Not Met | Not Met | Not Met | Not Met | Not Met | 0 | 0 | 5 | 0 |
| **User Experience** | | | | | | | | | |
| I am able to view the object I can control, and am able to see them transformed in real-time | Met | Met | Met | Met | Met | 5 | 0 | 0 | 100 |
| I am able to view where my hands are in relation to the virtual environment | Met | Met | Met | Met | Met | 5 | 0 | 0 | 100 |
| I am able to use a visual menu that extends the functionality of the environment | | | | | | | | | |
| Reset | Met | Met | Met | Met | Met | 5 | 0 | 0 | 100 |
| Change Object | Not Met | Not Met | Not Met | Not Met | Not Met | 0 | 0 | 5 | 0 |
| Fly-Through | Met | Met | Met | Met | Met | 5 | 0 | 0 | 100 |
| **Met** | 8 | 9 | 9 | 8 | 9 | | | | |
| **Partially Met** | 1 | 0 | 0 | 1 | 0 | | | **Average Test Score (%)** | |
| **Not Met** | 2 | 2 | 2 | 2 | 2 | | | | |
| **Score (%)** | 77.27273 | 81.81818 | 81.81818 | 77.27273 | 81.81818 | | | 80 | |

**Fig. 4.** User Acceptance Testing results

the reset requirement was met. Using the UAT scoring matrix, the requirement has been assigned a test score of 80%. Two out of five testers indicated that this functionality was partially met; indicating to the interviewer that they felt the reset functionality should reset the entire window as opposed to just the object. However, this is not necessary as the main environment is the controllable object and a window reset can be done by restarting the application.

## 4 Conclusions and Future Work

In this paper we have presented an architecture to represent gestures based on neural networks and 3D sensors. The system operates a fully functional natural user interface (NUI) with 3D reconstruction of hands, an intuitive HCI paradigm, and tools for fly-through interactivity as is used in volume visualisation applications. As for future work, we will improve the system performance at all stages to achieve a natural interface that allows us to interact with any object manipulation system.

## References

1. Albrecht, I., Haber, J., Seidel, H.: Construction and animation of anatomically based human hand models. In: Proc. of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 98–109 (2003)
2. Angelopoulou, A., Psarrou, A., García Rodríguez, J.: A Growing Neural Gas Algorithm with Applications in Hand Modelling and Tracking. In: Cabestany, J., Rojas, I., Joya, G. (eds.) IWANN 2011, Part II. LNCS, vol. 6692, pp. 236–243. Springer, Heidelberg (2011)
3. Calhoun, P.S., Kuszyk, B., Heath, D., Carley, J., Fishman, E.: Three-dimensional volume rendering of spiral CT data: Theory and method. RadioGraphics 19(3), 745–764 (1999)
4. Eddy, S.: Hidden Markov Models. Current Opinion in Structural Biology 6(3), 361–365 (1996)
5. Fritzke, B.: Growing Cell Structures - A self-organising network for unsupervised and supervised learning. The Journal of Neural Networks 7(9), 1441–1460 (1994)
6. Fritzke, B.: A growing Neural Gas Network Learns Topologies. In: Advances in Neural Information Processing Systems 7 (NIPS 1994), pp. 625–632 (1995)
7. Koike, H., Sato, Y., Kobayashi, Y.: Integrating Paper and Digital Information on Enhanced Desk: A Method for Real Time Finger Tracking on an Augmented Desk System. ACM Transactions on Computer-Human Interaction 8(4), 307–322 (2001)
8. Martinez, T., Schulten, K.: Topology Representing Networks. The Journal of Neural Networks 7(3), 507–522 (1994)
9. Ravikiran, J., Mahesh, K., Mahishi, S.: R., D., Sudheender, S., Pujari, N.: Finger detection for sign language recognition. In: International MultiConference of Engineers & Computer Scientists, p. 489 (2009)
10. Rosenblum, L.J.: Scientific Visualization: Advances and challenges, vol. 4 (1994)
11. Rusu, R., Cousins, S.: 3D is here: Point Cloud Library, PCL (2011)
12. Stergiopoulou, E., Papamarkos, N.: Hand gesture recognition using a neural network shape fitting technique. Engineering Applications of Artificial Intelligence 22(8), 1141–1158 (2009)
13. Sui, C.: Appearance-based hand gesture identification. Master of Engineering, University of New South Wales (2011)
14. Vamplew, P., Adams, A.: Recognition of Sign Language Gestures using Neural Networks. Australian Journal of Intelligent Information Processing Systems 5(2), 94–102 (1998)
15. Wong, S., Ranganath, S.: Automatic Sign Language Analysis: A Survey and the Future beyond Lexical Meaning. IEEE Transactions on Pattern Analysis and Machine Intelligence, 873–891 (2005)
16. Yang, J., Bang, W., Choi, E., Cho, S., Oh, J., Cho, J., Kim, S., Ki, E., Kim, D.: A 3D Hand-drawn Gesture Input Device Using Fuzzy ARTMAP-based Recognizer. Journal of Systemics, Cybernetics and Informatics 4(3), 1–7 (2009)