

# Accessing Learning Objects in Virtual Environment by Hand Gestures and Voice

Qing Chen, ASM Mahfujur Rahman, Ayman El-Sawah  
Xiaojun Shen, Abdulmotaleb El Saddik and Nicolas D. Georganas

DISCOVER, MCR Lab  
School of Information Technology & Engineering  
University of Ottawa, Ontario K1N 6N5, Canada  
E-mail: {qchen, aelsawah, shen, georganas}@discover.uottawa.ca  
{kafi, abed}@mcrlab.uottawa.ca

## Abstract

*This paper presents a human-computer interface based on hand gesture recognition and voice to manipulate the learning objects in a 3D virtual environment. With this interface, the user can search, retrieve and access the learning objects by a set of hand gesture commands and voice controlling the movements of the Avatar car. Two gesture recognition approaches are implemented: the appearance-based approach and the 3D hand model-based approach. The appearance-based approach has the advantage of real-time performance. The 3D hand model-based approach allows more natural dynamic gestures. The learning objects and the virtual environment consider the semantic meaning of the query and apply context-based peer address mapping to search metadata. Based on the experiment results, compared with controlling the Avatar car with the arrow keys on the keyboard, it is more intuitive and interesting for the users to use hand gestures to control the movement of the Avatar car.*

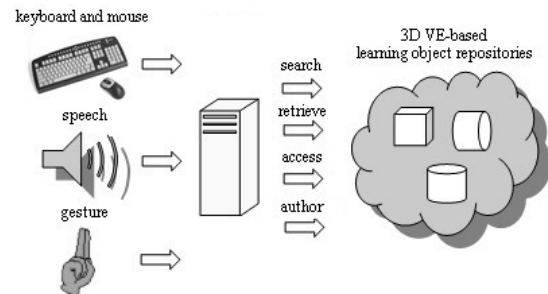
## 1. Introduction

Virtual environments provide a new paradigm for human communication, interaction, learning and training. To interact with the virtual environment, besides traditional human-computer interaction devices such as keyboards and mice, different sensing modalities and technologies can be utilized and integrated with the virtual environment for a more natural user experience [1]. Devices which sense body position and orientation, speech and sound, facial expression, haptic response and other aspects of human behavior or state can be used for the communication between the human and the virtual environment. These devices and tech-

niques make natural and immersive human-computer interfaces (HCI) for applications in 3D virtual environments promising [2, 3, 4].

## 2. Multi Modal-Based Manipulation of Learning Objects in 3D Virtual Environment

The three-dimensional (3D) information provided by virtual environments offers several possibilities such as perceiving more information at a time, displaying meaningful patterns in the data and understanding the relationship



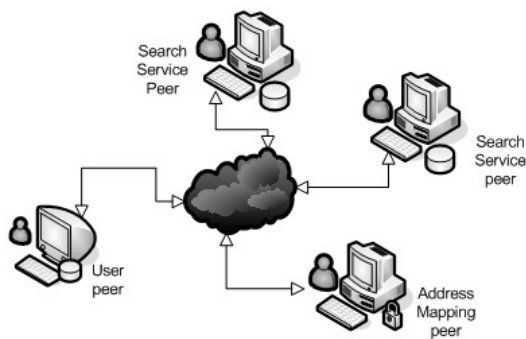
**Figure 1. The architecture of multi modal-based manipulation of learning objects in a 3D virtual environment.**

among different data items [5]. These possibilities may be utilized in different contexts especially in visualizing learning objects as it requires novel and intuitive presentation techniques than what is provided by the traditional 2D approaches [6]. To access the learning objects in a 3D virtual environment, the traditional mouse and keyboard are limited as the mouse itself is a 2D device and the arrow keys on keyboard is not an intuitive approach for human

beings. To overcome these limitations, a multi model-based approaches can be employed to achieve more powerful and natural interaction between the user and the virtual environment. Besides the mouse and keyboard, other modalities can be human voice, hand gestures, *etc.* Figure 1 showed this architecture.

Hand gestures are powerful human to human communication modality. For example, sign languages have been used extensively among speech disabled people. People that can talk and listen also use many kinds of gestures to help the communication in daily life. However, the expressiveness of hand gestures has not been fully explored for the virtual environment applications. Compared with traditional human-computer interaction devices, hand gestures are less intrusive and more convenient to explore the 3D virtual worlds [4]. The speech recognition has been successfully used in many categories. With the application of human voice to interact with the virtual environment, more easiness can be achieved for the user rather than typing the keyboard. To combine the entertaining elements and enhance the user's learning experience while searching and navigating the virtual environment, the computer game concepts can be incorporated with the virtual learning environment for visualizing distributed learning objects. With the hand gestures and the human voice as the input, the user can view, search, access, author and retrieve the learning objects from the distributed learning object repositories.

### 3. Learning Objects and the Virtual Environment



**Figure 2. The peer to peer searching environment: peers can register in the group's address mapping peer and voluntarily serve as a search service peer.**

Learning objects are entities that are generally suitable in the context of mathematics, engineering, technology, and health science for learning, education and training [7].

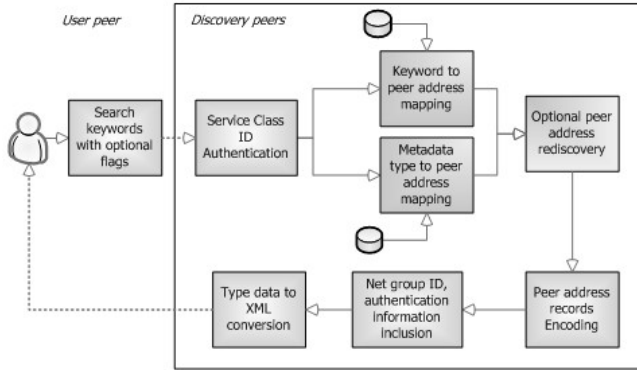
Learning object metadata (LOM) is comprised of some standardized elements for searching, managing and retrieving learning objects. With the advancement in Internet and computing technologies, learning resources are now easy to share and reuse. Learning object repositories can store these learning resources as well as their metadata records [8]. In this paper, a peer-to-peer network architecture is used to tie together all the components of our framework (see Figure 2). With this architecture, the learner's experience can be facilitated by sharing, searching and browsing interesting learning objects. The framework adopts algorithms which can group the searched learning object metadata together and map those in a 3D virtual environment. This framework offers several perspectives of the extracted information and enables a learner to perceive more information from many dimensions at a time. Meanwhile, the strategy of "divide and conquer" are used in the framework so that the overall software system can be decomposed into its individual components, and some of the components may be optional to implement for certain individuals. The goal of mapping the information into 3D metaphor is to allow the user to perceive the information and find related resources in an intuitive and entertaining manner by navigating the car metaphor through the virtual roads of the landscape.

The employed framework allows other institutions to use the provided services. Hence, to promote the "share and reuse" of multimedia learning materials, the peer-to-peer network has been logically categorized into three main types - user peer, address mapping peer, and search service peer. Any peer requesting services is termed as the user peer. Usually it sends the search keywords to the address mapping peer of a particular group  $\hat{G}$ . The address mapping peer applies some procedure and returns the path information. The user peer then uses that information to send the search keywords to the search service peers of  $\hat{G}$ .

A search service peer in the system allows registered users to search and retrieve learning object metadata. The shared information from this peer is used to access the content from standard learning object repositories and to browse the 3D virtual environment. The distributed processes combine their computational processing power to respond to the search queries.

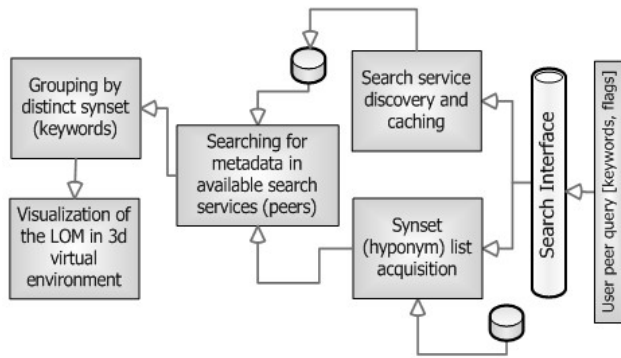
The discovery peers are responsible for producing reliable peer addresses that can serve search services. The search service peers are partitioned according to various requests that they can perform. Whenever a peer wants to provide service to a peer group, it registers itself to the address mapping peer of that group by providing information on how its service will be mapped. As depicted in Figure 3, to access a search service from a group  $\hat{G}$ , the user peer first sends its search keywords to the address mapping peer of  $\hat{G}$ . The discovery system then uses the heuristics to map keywords to the relevant search service peer's address of

$\hat{G}$  and includes the authentication information. By altering the optional flags, the user peer can request address rediscovery so that the search service addresses returned will be validated before sending. Peer address encoding and XML conversion are another two optional services that the peer can request.



**Figure 3. Different functional components of a discovery peer.**

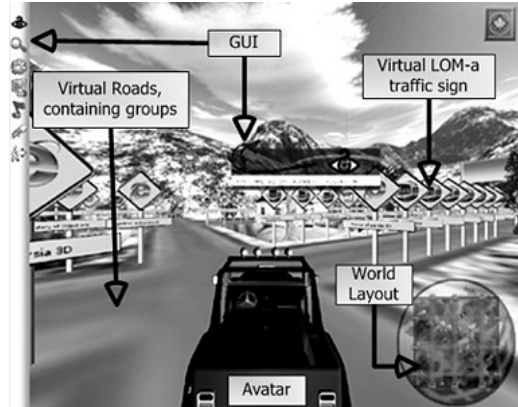
The address mapping peers employ lexical keyword sense mapping to find relevant subject matter on the search keywords. The process is inspired by current psycholinguistic theories of human lexical memory [9]. English nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing one underlying lexical concept. The hypernym relations are used to find out the link in the synonym sets and the primary focus is on nouns and verbs.



**Figure 4. The information visualization architecture.**

The graphical user interface in the virtual environment takes the query and feeds it to both the search service discovery module and the keyword sense-mapping module. Figure 4 is the information visualization architecture. A

search is initiated by using all the possible senses of the search keywords and then sends them to the search service peers. In turn, the search service peers return XML learning object metadata. The information obtained is then grouped using an algorithm that considers the keyword senses. The information visualization engine then uses these groups and maps them into the 3D virtual environment.



**Figure 5. The virtual environment model: search results are grouped according to the keywords and are associated with each road in the 3D car metaphor.**

Java 2 platform and Java bindings of the OpenGL library are used for the development of the virtual environment model showed in Figure 5. The learning object metadata can be grouped as a virtual road in the interface, and each metadata is represented as a 3D traffic sign. The text and icon of the traffic sign describes the content of the learning object metadata that can be selected by the user. The user is represented by the Avatar car on the road, and the world layout gives the current position of the user's avatar in the virtual environment.

#### 4. Virtual Environment Navigation with Appearance-Based Hand Gestures

As the user is represented by the Avatar car, we implemented a vision-based hand gesture recognition system to navigate the Avatar car in the virtual environment with a set of different hand gestures. To use human hand as the control device for the virtual environment, the hand gesture recognition system must meet the performance requirements in terms of real-time, accuracy, robustness and scalability.

Vision-based hand gesture recognition techniques can be grouped into two categories: 3D hand model-based approaches and appearance-based approaches [10]. The 3D hand model-based approaches employ an estimation-by-synthesis strategy, and recover the hand parameters by

aligning the appearance projected by the 3D hand model with the observed image features, and minimizing the discrepancy between them [11]. Generally speaking, 3D hand model-based approaches offer a rich description that potentially allow a wide class of hand gestures. However, as the 3D hand models are articulated deformable objects with many degrees of freedom, a very large image database is required to cover all the characteristic shapes under different views. Matching the query image frames from live video with all images in the database is time-consuming and computationally expensive. The appearance-based approaches are based on direct registration of hand gestures with 2D image features. The popular image features to detect human hands and recognize gestures include skin color, hand shape/contour or the combination of these features. Compared with 3D hand model-based approaches, appearance-based approaches have a much more simplified hand model to implement.

Viola and Jones employed a statistical approach for the task of face detection to handle the large variety of instances of human faces [12]. For hand gestures, generally speaking, the reproducibility in real situations is also very poor due to the high degree of freedoms of the human hand as well as the difficulties to duplicate the same working environment including background and lighting condition. Under these situations, a statistical model-based training algorithm can be employed to attack the variation problem. Statistical model-based training algorithms take a set of “positive” samples which contain the object of interest (in our case: human hand) and a set of “negative” samples, *ie.* images do not contain object of interest [13]. During the training process, distinctive features are selected to classify the images containing the object of interest. When the trained classifier misses an object or detects a false object, adjustment can be made easily by adding corresponding positive or negative samples to the training set. Viola and Jones method has been primarily used for face detection which is approximately 15 times faster than any previous approaches while achieving equivalent accuracy to the best published results. Due to the similarity between face detection and gesture recognition, we decide to employ Viola and Jones algorithm for the task of hand tracking and gesture recognition to achieve real-time and accurate performance.

Haar-like features are used in Viola and Jones method. Each Haar-like feature is described by a template which includes 2 or 3 rectangles, its relative coordinates to the origin of the search window and the size of the feature. Figure 6 shows the extended Haar-like feature set by Lienhart [14]. The value of a Haar-like feature is the difference between the sum of the pixels grey level value within the black and white rectangular regions. To detect an object of interest, the image is scanned by a sub-window containing a specific Haar-like feature. Based on each Haar-like feature  $f_j$ , a

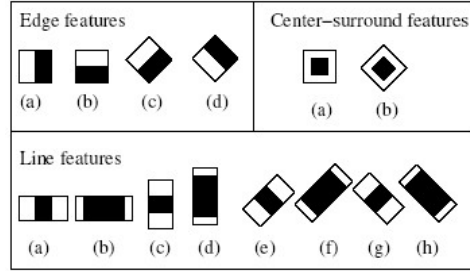


Figure 6. Extended set of Haar-like features.

correspondent weak classifier  $h_j(x)$  is defined by:

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

where  $x$  is a sub-window, and  $\theta$  is a threshold.  $p_j$  indicating the direction of the inequality sign.

There are two motivations for the use of the Haar-like features rather than the raw pixels. The first reason is that the Haar-like features can encode ad-hoc domain knowledge. Compared with raw pixels, the Haar-like features can efficiently reduce/increase the in-class/out-of-class variability and thus making classification easier [14]. The second motivation is that a Haar-like feature-based system can operate much faster than a pixel-based system.

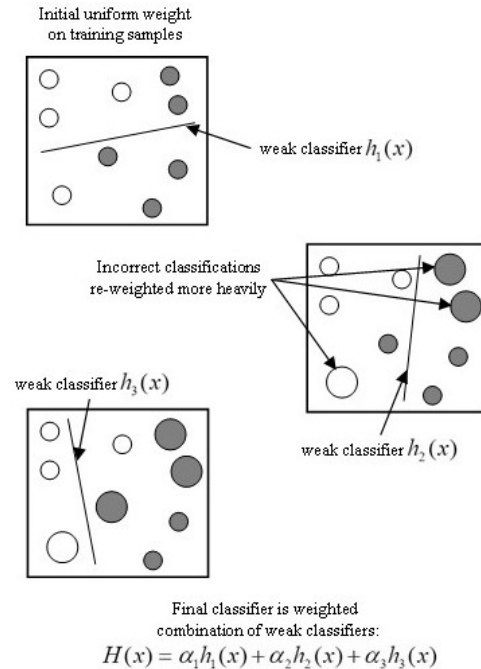


Figure 7. The description of AdaBoost learning algorithm.

The Adaboost learning algorithm is used in Viola and Jones method to select the features and to train the classifiers. The AdaBoost learning algorithm initially maintains a uniform distribution of weights over each training sample (in our case, the hand gesture images) as illustrated in Figure 7. In the first iteration, the algorithm trains a weak classifier using one Haar-like feature which achieves the best recognition performance for the training samples. In the second iteration, the training samples that were misclassified by the first weak classifier receive higher weights so that the newly selected Haar-like feature must focus more computation efforts towards these misclassified samples. The iteration goes on and the final result is a cascade of linear combination of the selected weak classifiers (i.e. a strong classifier which achieves required accuracy).



**Figure 8. Three different hand gestures used to navigate the Avatar car.**

To control the Avatar car, we defined three different hand gestures showed in Figure 8. The “palm” gesture is to activate the car, the “two-finger” gesture turns the car to left, and the “little finger” gesture turns the car to right. The camera used for the video input in our experiment is a low cost Logitech QuickCam web-camera showed in Figure 9. This web-camera provides video capture with maximum resolution of  $640 \times 480$  up to 15 frames-per-second. For the experiment, we set the camera parameters at  $320 \times 240$  with 15 frames-per-second.



**Figure 9. The camera for video input: the Logitech QuickCam web-camera.**

We collected the training samples from a real hand first for the initial experiment. The experiments and testings are implemented in the laboratory with natural fluorescent lighting condition. To variate the illumination, we put an extra incandescent light bulb to create a tungsten lighting

condition. For the “two-finger” gesture, we collected 480 positive samples with different scales. To increase the robustness of the final classifier, we deliberately included a number of positive samples with certain in-plane rotation and out-of-plane rotation. Figure 10 shows some positive samples of the “two-finger” gesture. To simplify the task at the initial stage of the experiment, we keep the white wall as the background for the testing. The scale of the “two-finger” gestures does not affect the training process at this point because we will mark out the hand area in all of the positive samples and crop them to the unified resolution of  $15 \times 30$ .



**Figure 10. Part of the “two-finger” positive samples used in the training.**

We collected 500 negative samples for the training process. These images must not contain the gesture representations. All negative samples are passed through a background description file which is a text file containing the filenames of all negative sample images.

With all of the positive/negative samples and required parameters, we set the required false alarm rate at  $1 \times 10^{-6}$  to terminate the training, which means the accuracy of the classifier will meet the requirement when 1 out of 1 million negative sub-windows are mistakenly detected as a positive sub-window.

A 15-stage cascade is trained for the “two-finger” gesture with the AdaBoost learning algorithm. When the final required false alarm rate  $1 \times 10^{-6}$  is reached, the true-positive detection rate of the final classifier is 97.5% (468 out of 480). For the “palm” gesture, a 10-stage cascade is trained with 412 positive samples. The true-positive detection rate is 98% (404 out of 412) when the required false alarm rate is reached. For the “little finger” gesture, a 14-stage cas-

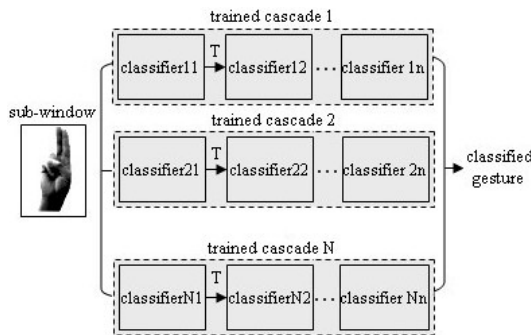
cade is trained with 420 positive samples. The true-positive detection rate is 97.1% (408 out of 420) when the required false alarm rate is reached.

In order to evaluate the performance of the trained classifiers, a collection of 100 marked-up images for each gesture (which are not used as positive samples for the training process) is collected separately with similar background but different illumination condition. Table 1 compared the performance of the four trained classifiers and the time spent to detect all 100 testing images.

Gesture	Hits	Missed	False	Time (second)
Two-finger	100	0	29	3.049
Palm	90	10	0	1.869
Little Finger	93	7	2	2.452

**Table 1. The detection results of the trained classifiers.**

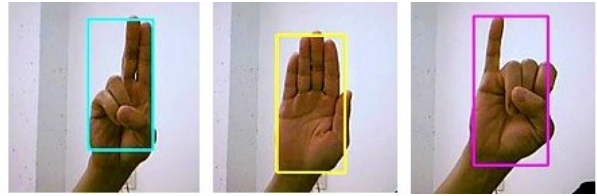
By analyzing the detection results, we found that some of the missed positive pictures is caused by the excessive in-plane rotations. For the false detections, the majority of them only happened in very small areas which have a higher probability containing similar color patterns to be detected by the selected Haar-like features. These small false detection boxes is not a serious problem because they can be easily eliminated by defining a size threshold. The maximum time required to detect 100 testing images is the “two-finger” gesture classifier which cost 3.049 seconds. The time required for the other classifiers are all within 3 seconds. We tested the real-time performance with live input from the Logitech QuickCam web-camera with 15 frames-per-second at the resolution of  $320 \times 240$ , and there is no detectable pause and latency to track and detect the hand gestures with all our trained classifiers.



**Figure 11. The parallel cascades structure for hand gesture classification.**

With the quick detection speed, we implemented a parallel cascades structure to classify different gestures (see Fig-

ure 11). In this structure, multiple cascades are loaded into the system simultaneously, and each cascade is responsible for a single hand gesture. Rectangles of different colors are used by the program to tell which gesture is detected.



**Figure 12. The recognition result with the parallel cascades structure.**

Based on the experiment results, with our VR player turned on, we found the real-time performance of the classification is not impaired when we load the three trained cascade classifiers and the virtual environment at the same time, and there is no confusion detected as illustrated in Figure 12.

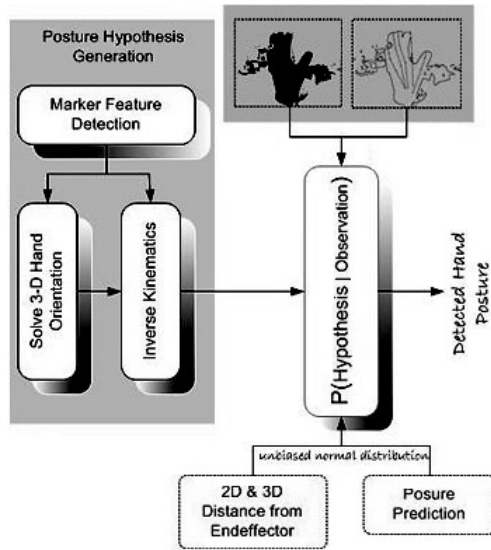
## 5. Virtual Environment Navigation with Marker-Based 3D Hand Model-Based Gestures



**Figure 13. The black glove with colored markers.**

As stated previously, 3D hand model-based approaches offer a rich description that potentially allow a wide class of hand gestures. One of the advantages is that the dynamic properties of the hand gestures can be described more easily with a 3D hand model. To incorporate the advantages of the dynamic hand gestures into our system, we also implemented a marker-based hand gesture recognition system to navigate the 3D virtual environment. The colored markers are attached on a black glove as showed in Figure 13. The major phases involved in this approach include the feature recognition in 2D space, the conversion into 3D space,

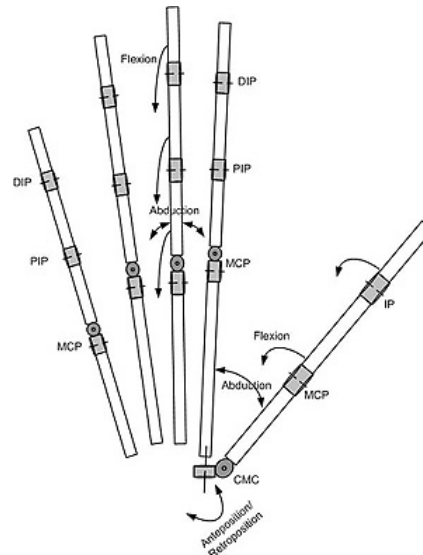
the dynamic gesture recognition using a Dynamic Bayesian Network.



**Figure 14. The marker-based 3D hand posture estimation framework.**

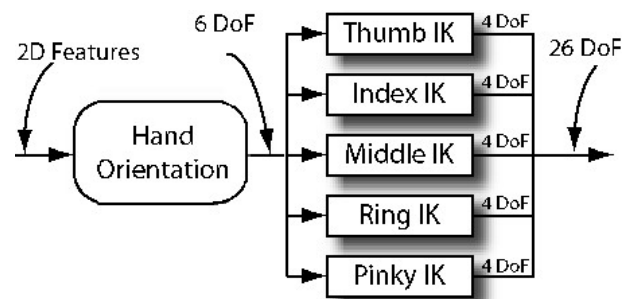
In this section, we define the term “hand posture” as a static hand pose without any movements, and the term “hand gesture” as a sequence of hand postures within a certain time. The 3D hand posture estimation framework is showed in Figure 14. The extracted features are used to generate hand posture hypothesis which are then validated using our observation model. Posture hypothesis are derived by solving the inverse perspective projection of the hand marker corners. It is possible to determine the location and orientation, up to two possible orientations, of the marker in 3D with respect to camera coordinate frames from its corners projection on the image plane. The hand model orientation is then determined using a pre-determined hand marker to hand model transformation. The projection of the finger tip markers are used to generate finger end-effector hypothesis which are solved using inverse kinematics. Feasible postures are further validated using an observation model, which determine the probability of posture hypothesis given the image.

The 3D hand model is used to generate posture hypothesis from image cues and to evaluate the probability of those hypothesis given image cues. Our 3D hand model, showed in Figure 15, consists of a 3-Degree of Freedom (DOF) revolute joint for the root (wrist) and five link chains (fingers) connected to the root. All fingers, except for the thumb, consist of 4 links; the first link is fixed to the root and is part of the palm, the second link is connected to the chain by two 1-DOF revolute joints connected by an abstract zero-



**Figure 15. The 3D hand model.**

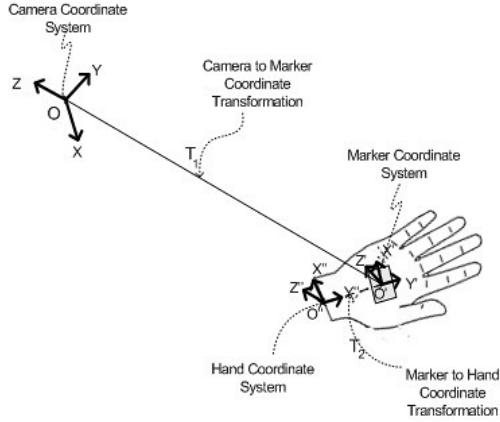
length link. The other two links are connected to the chain through a 1-DOF revolute joint. The thumb has only three links; the first link is connected to the root by two 1-DOF revolute joints, connected to each other by an abstract zero-length link to model the 2-DOF carpometacarpal joint. The finger links are modeled as cylinders, and 1-DOF revolute joints are modeled using a shaded cylinder in the direction of the rotating axis. A 2-DOF revolute joint is simplified as two 1-DOF revolute joints connected by a zero-length and zero-mass (dummy) link. In the hand model there are 26 links including 7 dummy links: 5 for the fingers and 2 for the wrist, and 23-DOF revolute joints - the root’s joints are not shown.



**Figure 16. Solving Model DOF in two stages.**

The 26-DOF hand model are solved, given the detected marker positions, in two stages as illustrated in Figure 16. In the first stage, the 6-DOF associated with the model root (location and wrist orientation) is determined using the perspective geometry of detected 2D features on the hand. In the second stage, finger postures are determined using in-

verse kinematics (IK) in order to solve the angles of the fingers joints.



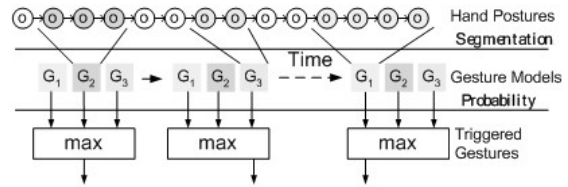
**Figure 17. Camera to Marker Coordinate Transformation.**

Hand orientation in 3D is formally defined by the camera to the hand model coordinate transformation as shown in Figure 17. The corners of the square shaped hand markers are used to calculate the camera to marker transformation  $T_1$  using perspective geometry and the marker’s geometrical constraints. The marker-to-model coordinate transformation  $T_2$  is determined priori and is considered constant throughout tracking. Finger pose detection relies on the fingers IK to calculate the joint angles. Due to loss of depth information, all possible finger postures along the camera’s view trace are determined.

We evaluate the probability of the generated hand postures, given the features extracted from the image. The hand model is projected on the image plane and a 2D observation is performed by sampling the model and evaluating the probability of each point independently. First, each sample point must lie on a pixel with the hand or glove color. A binomial density function is used, with probability  $P$  that the sample lies on a hand color and probability  $Q = 1 - P$ , that the sample will lie on a pixel with another color.  $P$  is chosen much larger than  $Q$ . Second, points on the model silhouette are compared with edges detected from the image. As clutter is expected, we adopt a functional adapted form [15] by the assumption that clutter is a Poisson process, that all edges has the same probability of being the true observation, and that the measured target accuracy is a normal distribution. Third, as we may detect multiple possible targets for the end-effector of a finger, a similar functional used to determine the probability of the end-effector hypothesis, given all the detected targets. The maximum probability posture hypothesis is used as the candidate posture.

We use a Dynamic Bayesian Network Model (DBNM) for the dynamic gestures. In our model, we observe a se-

quence of hand postures and we assume that it forms a DBN, which means that each observed posture  $o_i$  is dependent on previously observed postures  $o_j$ , where  $j$  is less than  $i$ . We further simplify the DBN by using a first-order Markov Model as its causal model, which means that an observed posture  $o_i$  is solely dependent on its previous posture  $o_{i-1}$ . The DBNM of a dynamic gesture  $g_i$  is trained using gesture sample and is then used to provide a probability measure of the gesture  $g_i$ , given a sequence of hand postures  $\Theta - P(g_i|\Theta)$ . The most probable gesture model (with probability past a predefined threshold) triggers the recognition of its corresponding gesture  $g_i$ . It also provides the segmentation of the recognized gesture from the posture sequence as shown in Figure 18.



**Figure 18. The DBNM Architecture.**

We evaluate the probability of the underlying dynamic gesture models, given an observed sequence of hand postures using Bayes rule (2):

$$P(g_i|\Theta) = P(\Theta|g_i) \cdot P(g_i)/P(\Theta)$$

We evaluate the probability of the gestures over the same data, thus we could ignore the probability of the observed data in our calculation. Moreover, the canonical probability of the gesture models depends on the context. Thus, the gesture dependent factor in above equation is the probability of the observed sequence of postures, given the dynamic gesture model  $P(\Theta|g)$ :

$$\begin{aligned} P(\Theta|g) &= P(o_1, o_2 \dots o_T|g) \\ &= P(o_1|g)P(o_2|o_1, g) \dots P(o_T|o_{1..T-1}, g) \end{aligned}$$

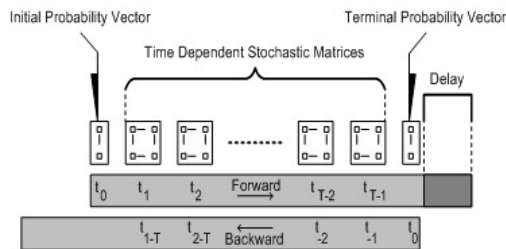
Applying the first order Markov model to the DBN, the probability of the observed posture sequence can be computed as:

$$P(\Theta|g) = P(o_1|g)P(o_2|o_1, g) \dots P(o_T|o_{T-1}, g)$$

Our dynamic gesture model is defined, using above equation, by an initial probability vector specifying  $P(o_1|g)$ , and a sequence of posture transition stochastic matrices specifying  $P(o_i|Po_{i-1}, g)$  acquiring the dynamic aspect of the gesture. The number of matrices included in the model is determined by the sampling frequency and the maximum time period of the gesture. A terminal posture probability vector is included in the model to enable the

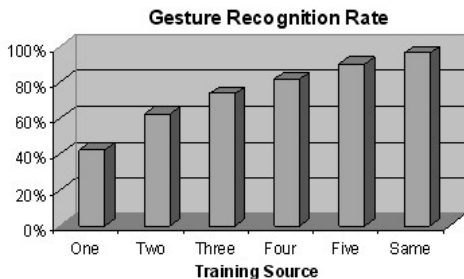
segmentation of the dynamic gesture over an extended time window.

The dynamic gesture model can be trained from just a few samples by randomizing the multi-dimensional posture attribute vectors using an offset and time scaling random variables to accommodate dynamic gesture acuteness and speed variations respectively. The posture attributes are mapped to a discrete set of postures using a multi-threshold posture classifier. The model's probability vectors and stochastic matrices are then evaluated by integrating the offset and time scaling random variables' probability density functions over the ranges over which the resulting postures remain constant. Models created using other training samples can be merged using a merge operator, which is based on a weighted average of the associated probability vectors and stochastic matrices and the weight is proportional to the number of samples used to create the model.



**Figure 19. Forward vs. reverse time traversal.**

The probability of observation, given a gesture model is evaluated efficiently in real time by acquiring posture attributes at sample times, classifying the hand postures, and using the classified postures as indices to elements from the model's initial probability vector and posture transition matrices. The dynamic gesture is segmented by detecting the terminal posture. We save some latency by training the dynamic gesture model in reverse time traversal as shown in Figure 19.



**Figure 20. The DBNM Recognition Rates.**

We tested our dynamic gesture recognition framework using a CyberGlove. To test the recognition success rate of our scheme, we conducted a test using sample sets of 14

gestures, including 7 dynamic gestures and 7 postures. The gesture samples were acquired using a CyberGlove from 6 different operators. The gestures trained on the first operator were used to recognize the gesture samples from the rest of the operators. Then, we progressively trained on the least successful operator and redone the recognition test on the rest of the operators. The results of the experiment are shown in Figure 20.

The success rate progressively increased as we trained on more operators. Moreover, we realized that the recognition rate is improved dramatically if we use the gestures trained on the same operator to recognize his gestures. We think this is due to the variation in the CyberGlove calibration between the operators. We are currently testing the dynamic gesture model with postures acquired from the tracking system. To compensate for the noise produced by the imaging artifacts we are using a fuzzy DBNM in which a number of candidate postures provided by the vision-based posture estimator are used by the DBNM to detect the required gesture.

## 6. Integration of Hand Gestures, Speech and the Virtual Environment

We use Sphinx-4 [16], a speech recognizer written entirely in the Java programming language to provide a speaker independent recognition of pre-defined commands described by a grammar. Speech recognition and speech synthesis free the user hands for other tasks and allows the users to take advantage of their natural voice communication skills. In the e-commerce application, speech recognition can be put alongside the web browser simulation to aid a user to find items of interest within the learning objects repository.

The Java Native Interface (JNI) framework is employed to integrate our gesture recognition component with our Java-based virtual environment. With this framework, the Java native methods can call applications and libraries written in C/C++.

We tested the navigation of the virtual environment with proposed gesture commands. Compared with controlling the Avatar car with the arrow keys on the keyboard, it is more intuitive and interesting for the users to use hand gestures to control the movement of the Avatar car.

## 7. Conclusions

In this paper, a multi model-based interface for manipulating the learning objects in a 3D virtual environment is implemented. The learning objects and the virtual environment consider the semantic meaning of the query and applies context-based peer address mapping to search metadata. With the multi-model-based interface, the user can

access learning objects and navigate the virtual environment with hand gestures as well as voice. The hand gestures are implemented with two approaches: the appearance-based approach and the marker-based 3D hand model-based approach. The appearance-based approach provides accurate and real-time control for the movements of the Avatar car. The marker-based 3D hand model approach can integrate the dynamic properties into the system. Compared with traditional human-computer interaction devices such as the keyboard and the mouse, this multi-model-based interface provides more naturalness and entertainment to the user.

## References

- [1] M. Turk. *Gesture Recognition in Handbook of Virtual Environment Technology*. Lawrence Erlbaum Associates, Inc., 2001.
- [2] V. Pavlovic, R. Sharma, and T. Huang. Gestural interface to a visual computing environment for molecular biologists. *Proc. Second International Conference on Automatic Face and Gesture Recognition*, pages 30–35, 1996.
- [3] T. Kirishima, K. Sato, and K. Chihara. Real-time gesture recognition by learning and selective control of visual interest points. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(3):351–364, 2005.
- [4] Y. Wu and T. S. Huang. Hand modeling analysis and recognition for vision-based human computer interaction. *IEEE Signal Processing Magazine, Special Issue on Immersive Interactive Technology*, 18(3):51–60, 2001.
- [5] S. K. Card, J. D. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers, New York, 1999.
- [6] J. Klerkx, E. Duval, and M. Meire. Using information visualization for accessing learning object repositories. *Proc. 8th Int'l Conf. Information Visualization (IV'04)*, 14(5):465–470, 2004.
- [7] IEEE Learning Technology Standards Committee. *IEEE Learning Object Metadata, final draft standard (IEEE1484.12.1)*, 2002.
- [8] F. Neven and E. Duval. Reusable learning objects: a survey of lom-based repositories. *Proc. 10th ACM Int'l Conf. Multimedia*, pages 291–294, 2002.
- [9] Cognitive Science Laboratory, Princeton University. *Wordnet, a lexical database for the English language*.
- [10] H. Zhou and T. S. Huang. Tracking articulated hand motion with Eigen dynamics analysis. *Proc. of International Conference on Computer Vision*, 2:1102–1109, 2003.
- [11] A. Imai, N. Shimada, and Y. Shirai. 3-d hand posture recognition by training contour variation. *Proc. 6th IEEE International Conference on Automatic Face and Gesture Recognition*, pages 895–900, 2004.
- [12] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Proc. IEEE Conference on Computer Vision and Pattern Recognition CVPR2001*, pages 511–518, 2001.
- [13] G. Bradski, A. Kaehler, and V. Pissaris. Learning-based computer vision with Intel's open source computer vision library. *Intel Technology Journal*, 9(2):119–130, 2005.
- [14] R. Lienhart and J. Maydt. An extended set of Haar-like features for rapid object detection. *Proc. IEEE International Conference on Image Processing ICIP 2002*, 1:900–903, 2002.
- [15] A. Blake and M. Isard. *Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion*. Springer publisher, 2000.
- [16] Sphinx-4, A Speech Recognizer Written Entirely in the Java Programming Language. <http://cmusphinx.sourceforge.net/sphinx4/>.