A Tunnel Window and Its Variations: Seamless Teleportation Techniques in a Virtual Environment

Kiyoshi Kiyokawa *Haruo Takemura*

Cybermedia Center, Osaka University
1-32 Machikaneyama, Toyonaka, Osaka 560-0043, Japan
{kiyo, takemura}@ime.cmc.osaka-u.ac.jp

Abstract

This paper proposes a *tunnel window*, a versatile three dimensional interaction technique, for both navigation and remote object manipulation in a large scale virtual environment. A tunnel window, which is created at an arbitrary position in space using a set of hand gesture commands, provides a secondary view whose viewpoint can be controlled independently from that of the primary view. A tunnel window allows various kinds of seamless teleportation operations including teleportation of the user's viewpoint and that of a virtual object from one position to another by simply passing through the window frame. To extend the interaction scheme, a variety of transformation combinations of the relevant coordinate systems, i.e. those of a user, a window frame, and primary and secondary scenes, have been considered. This paper describes basic concepts and design of a tunnel window interface, its implementation, and variations of the tunnel window scheme.

1 Introduction

Our research goal is to provide a set of interaction techniques that can be used for both object manipulation and user navigation in an identical manner. Remote object manipulation and user navigation are both useful and usually necessary operations in a large-scale virtual environment (Mine, Brooks & Sequin, 1997; Bowman, Koller & Hodges, 1998). Direct manipulation (grabbing and moving an object in three-dimensional space just like in the real world) is a common approach for manipulating an object. However, virtual objects are not always within one's reach, especially when a virtual environment is large. So, a VR system often needs to provide a way to manipulate a distant object without the need for changing the user's actual position. Several approaches for remote object manipulation have been proposed. Mine et al. (1997) propose an approach which scales down the entire scene temporarily so that a target object comes close enough to reach. Poupyrev et al. propose another approach which non-linearly extends a virtual arm to the distant target (Poupyrev, Billinghurst, Weghorst & Ichikawa, 1996).

On the other hand, navigation is another essential task in a large-scale virtual environment. One main advantage of using a VR system is that a user can see and examine three-dimensional data from an arbitrary viewpoint. Most VR systems allow a user to change the viewpoint and viewing direction in real-time. When the virtual environment is large, the system needs to provide a navigation function such as *walk-through* and *fly-through* to move around the scene. Ware & Osborne (1990) and Stoev & Schmalstieg (2002) propose navigation techniques such as World-in-Hand and Eyeball-in-Hand that use the user's hand. Schmalstieg & Schaufler (1999) propose an approach for teleportation (instant movement to a remote place) by going through a virtual window frame in which a secondary view is shown.

Most interaction techniques for user navigation and object manipulation are independently developed. Many VR systems support the two types of operations in completely different manners. However, they are similar to each other in a sense that a user changes relative position and orientation between the user and objects. If they have the same interaction methodology in a reasonable way, users will have less difficulty in memorizing and performing commands.

Worlds-In-Miniature (WIM) (Stoakley, Conway & Pausch, 1995) is an intuitive approach to deal with both object manipulation and user navigation in an identical manner. With WIM, a miniature of a surrounding virtual space appears in front of a user in order to provide an overview of the scene, and to support direct manipulation of a remote object or the user’s viewpoint, by moving their miniature counterparts shown in the WIM. For example, a user can change his or her viewpoint by moving a doll representing the user in the WIM, just like moving a chess piece. Although WIM is an easy and natural interaction technique, it has a few drawbacks. For example, WIM consumes a certain screen area, occupies a versatile working volume near body which is desirable to remain empty for other kinds of direct manipulations, and requires a user to understand the relationships between the actual objects in the scene and their miniature counterparts in the WIM.

In this paper, we propose a *tunnel-window*, a versatile three dimensional interaction technique, for both user navigation and remote object manipulation in a large scale virtual environment. A tunnel window, which is created at an arbitrary position in space using a set of hand gesture commands, provides a secondary view whose viewpoint can be controlled independently from that of the primary view. A tunnel window allows various kinds of seamless teleportation operations including teleportation of the user’s viewpoint and that of a virtual object from one position to another by simply passing through the window frame.

This paper is organized as follows. Section 2 gives the hardware configuration of our system. Section 3 describes a few existing interaction techniques we have already proposed that are used in combination with the tunnel-window technique. Section 4 describes the design, implementation and examples of the tunnel window technique. Section 5 illustrates a variety of combinations among transformation matrices with regard to the relevant coordinate systems, i.e. those of a user, a window frame, and primary and secondary scenes.

2 System configuration

Our interaction techniques are based on hand gesture commands for three reasons. First, conventional keyboard-and-mouse type interactions are difficult to perform within a typical large screen system such as a CAVE. Second, gesture input can be natural and intuitive when carefully designed. The hand shapes and motions for our gesture commands are chosen from those we perform in daily activities, so we expect that those gesture commands are easy to remember. Third, thanks to the hand’s dexterity, a variety of different commands can be issued in a single step if needed, without any additional devices.

A CyberGlove (Immersion Co.) is used to track hand shape. Currently, one-handed (right hand) operations are used for all interaction techniques. It also provides simple tactile feedback (vibration) when a user ‘touches’ an object. A Polhemus Fastrak (3Space Co.) is used to track 6DOF information of user’s head and hand. A four-screen CAVE (2.3m x 2.3m x 2.3m) is mainly used as a visual display. A prototype system for the CAVE display was developed with OpenGL and SGI’s CAVElib libraries on SGI IRIX operating system. A desktop version was also developed on Windows XP with little modification.

3 Interaction techniques for short and middle range operations

We classified various kinds of operations into three categories depending on typical distance between the user and a target. Short, middle and long range operations deal with a target within the user’s reach, a target out of the user’s reach yet seen in the user’s view, and a target out of the user’s view, respectively. The main purpose of the tunnel window technique is to provide a remote view to the user for long range operations. Once a remote scene is shown in the user’s view, short and middle range operations can now be applied for performing long range operations to the remote scene. In the following, short and middle range operations in our interaction scheme are described.

Introducing an assumption that each object is either *fixed* or *movable* in the scene, our approach allows a user to perform both object manipulation and user navigation in an identical manner. Any operation performed on a fixed object is considered as a navigation command, while the same operation performed on a movable object is considered as a manipulation command. Each object has its default movability value, but a user is also able to change the value from ‘fixed’ to ‘movable’ and vice versa.

For short range operations, a common *direct manipulation* scheme is simply employed. A direct manipulation technique can be applied for user navigation by using World-in-Hand and Eyeball-in-Hand metaphors (Ware, 2000). A movable object follows the user when grabbed. When the grabbed object is fixed, user’s position and orientation move opposite to the hand motion. The latter operation appears as if the user moves the entire scene by hand.

For middle range operations, we have previously proposed a set of unified gesture-based interaction techniques (Tomozoe, Machida, Kiyokawa & Takemura, 2004), including *string-shot* and *skewer*. A *string-shot* operation is used primarily for moving an object or the viewpoint back and forth. Similarly to a common ray-casting technique, a user can shoot a string (line segment) from the tip of the index finger toward any object by a pistol-shaped hand posture (Figure 1a). While keeping the pistol-shaped posture, the direction of the string changes according to that of the index finger. The nearest object intersecting with the string is highlighted to indicate that the object is a candidate for manipulation. When the user spreads the palm (Figure 1b), the other end of the string is anchored (hooked) to the intersecting point.

After this, the user can control the distance between the user and the hooked object until the user makes a posture of scissors. To shorten the distance, the user needs to face his or her palm inward and bend and unbend the fingers repeatedly (Figure 1c) (we call this action *beckoning*). To lengthen the distance, the user needs to face his or her palm outward and do the same (Figure 1d) (*waving away*). During these operations, the line segment expands as an elastic rubber band. By beckoning, for example, the hooked object comes closer when it is movable, while the user moves forward instead when it is fixed. Figures 3j-l and 3e-f show other examples of beckoning and waving away, respectively. If the user makes a posture of scissors (Figure 1e), the string is detached from hand. The line segment, which is still anchored to the hooked object, remains in space for later manipulation.

After a string-shot operation, the line segment now becomes a rigid handle of a *skewer* operation, which is used primarily for rotating an object or the viewpoint. The user can simply grab (close the palm near) the line segment (Figure 1f) and swing it with the hooked object. The hooked object moves and rotates around the user when it is movable, while the user moves and rotates around the hooked object when it is fixed. The latter operation appears as if the user moves and rotates the entire scene around the user. When a user releases the skewer (spreads the palm) (Figure 1g), followed by making a posture of scissors (Figure 1h), the skewer finally disappears. Figure 3g-h show an example of a skewer operation for a movable object (cone), and Figure 3i shows another operation for a fixed object (wall). By using string-shot and skewer operations, users can perform 6DOF operations for a remote target.

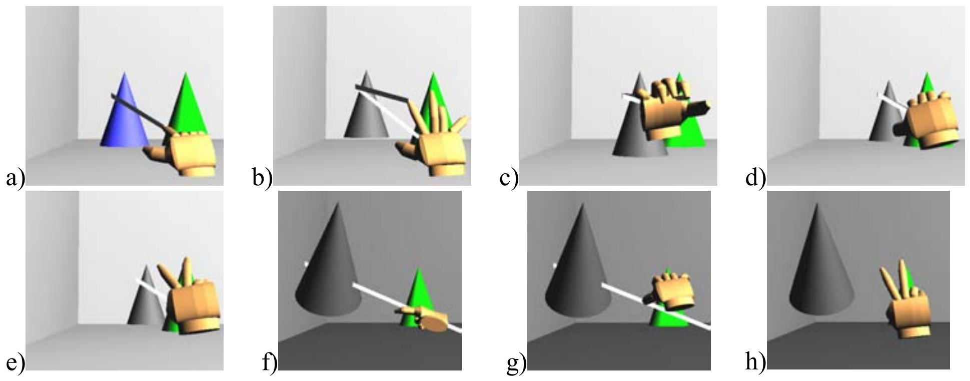

Figure 1: Middle range operations

4 Tunnel-window

4.1 Basic concept

We propose a *tunnel-window*, a novel versatile interaction technique for a variety of long range operations. A tunnel window is a virtual window frame through which a user can observe a secondary scene and perform various operations to the remote place in the view. Technically speaking, rendering algorithms and the basic concept of a tunnel window are quite similar to those of SEAMS (Schmalstieg & Schaufler, 1999). The main idea is to have different views in nested viewports to 'connect' multiple scenes. While SEAMS aims primarily at providing a hyperlink mechanism in a virtual environment, our techniques focus more specifically on user interaction techniques for remote object manipulation and user navigation. In this sense, a tunnel window technique is more similar to the Through-The-Lens (TTL) technique (Stoev & Schmalstieg, 2002), which offers a secondary view for object manipulation and navigation techniques such as WIM and World-in-Hand. Unlike the TTL technique, our technique allows a user to create an arbitrary number of viewing windows at arbitrary positions on demand.

4.2 Gesture-based interaction

Figure 2 shows a state transition diagram of a tunnel-window. Figures 3 and 4 show screenshots of a tunnel window operation in desktop and CAVE setups, respectively. Note that these operations in Figures 3 and 4 are just for examples and do not illustrate practical situations. Following four states exist in a tunnel-window operation.

- **Positioning:** a state in which a user is positioning a corner of a tunnel-window.
- **Framing:** a state in which a user is determining the other end of the window frame.
- **Waiting:** a state in which a tunnel window has been created and is ready for manipulation.
- **Holding:** a state in which a user is moving a tunnel window.

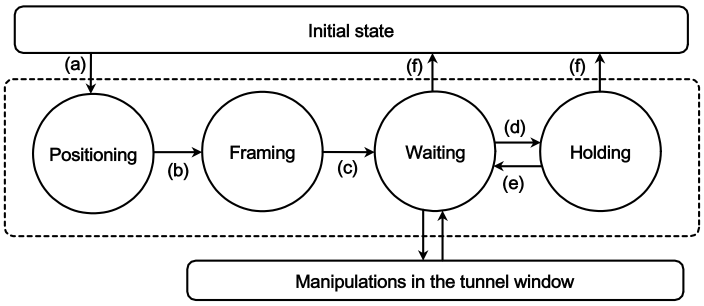

Figure 2: State transition diagram of a tunnel window operation

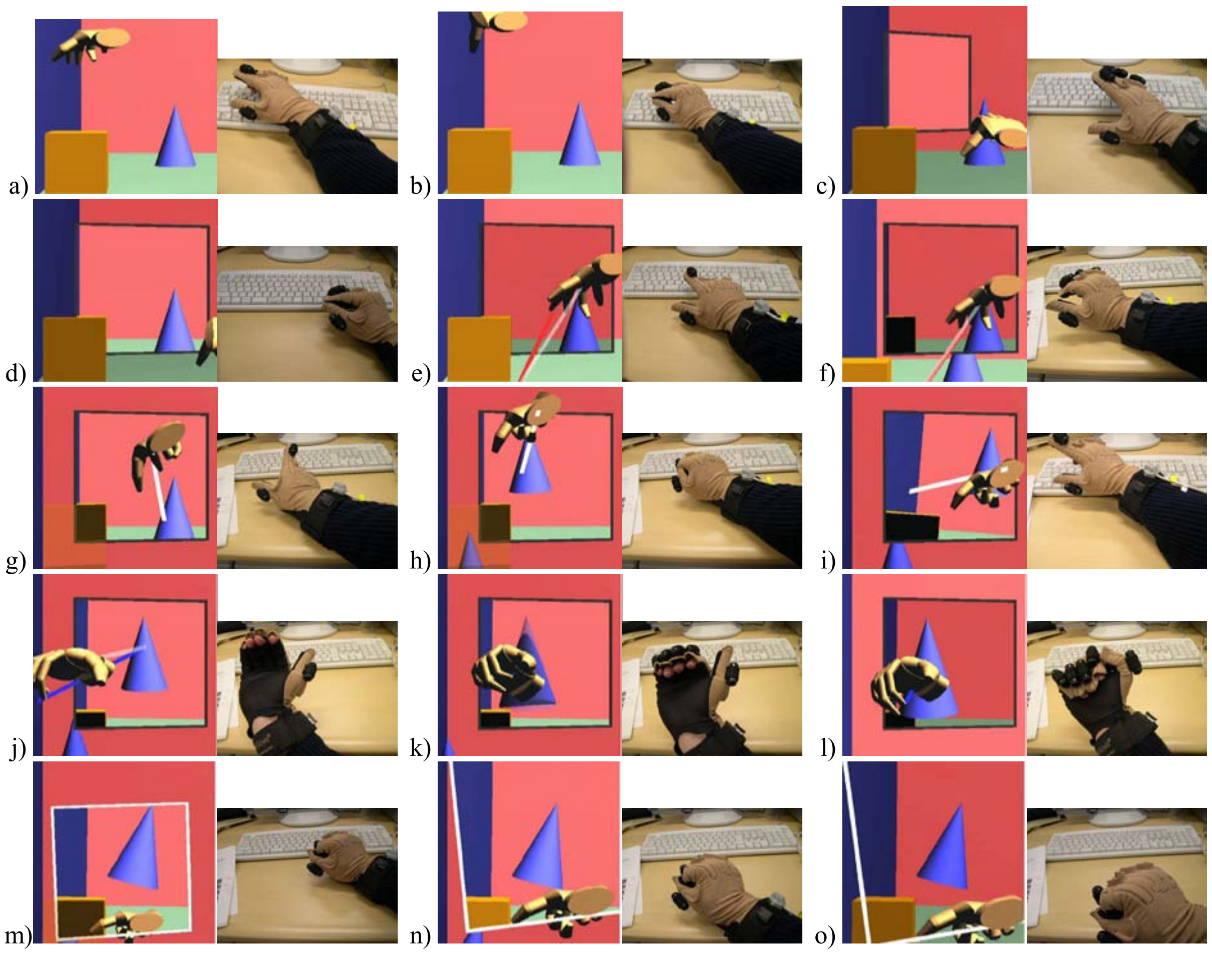

Figure 3: Tunnel-window operations in a desktop setup

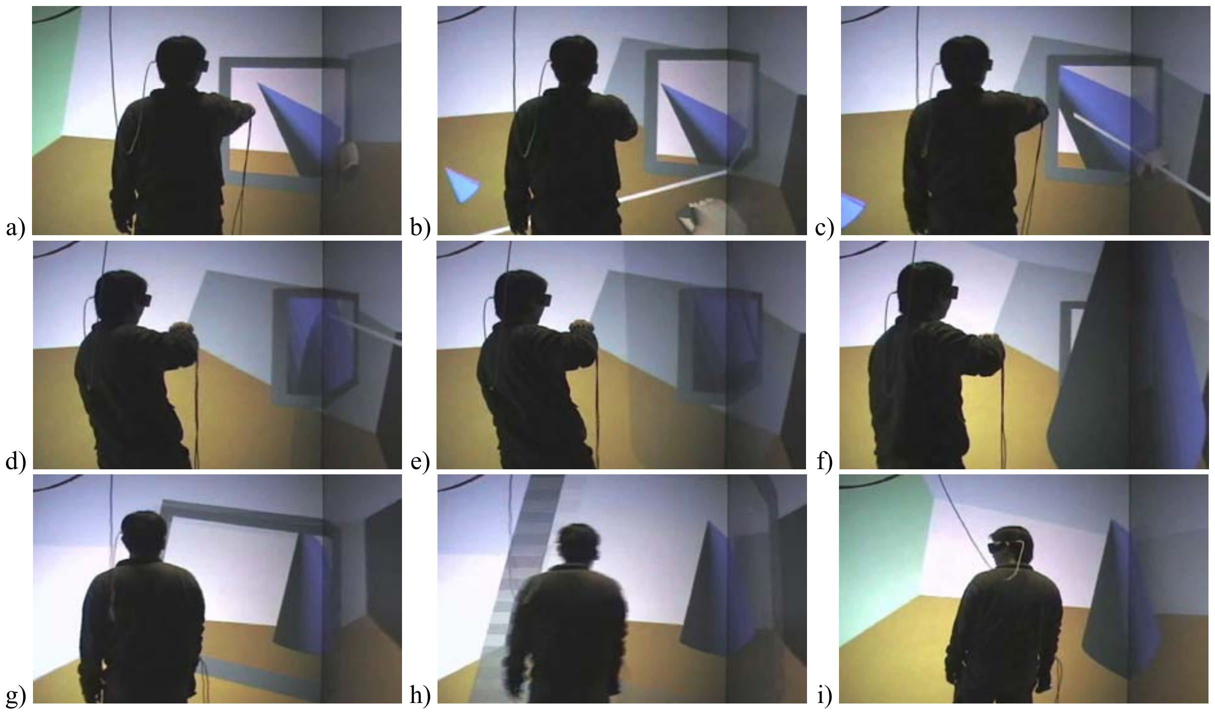

Figure 4: Tunnel-window operations in a CAVE setup

4.2.1 Creation and deletion of a tunnel window

Unlike the TTL technique, our approach provides no tunnel window in the beginning. Instead, our approach allows the user to create (and delete) an arbitrary number of tunnel windows at any position in space. To create a new tunnel window, the user simply needs to 'draw' a window frame in midair in two steps, positioning and framing. The creation process of a new tunnel window starts by specifying a corner of the window frame. When a user moves his or her hand to a candidate location (Figure 3a), and touches tips of his or her index finger and thumb (Figure 3b), the state changes to the *positioning* one (Figure 2a). In the positioning state, a small rectangle, which represents a corner of the new tunnel window, follows the tips of the fingers until the user opens them.

When the user opens the tips of his or her index finger and thumb, the first corner of the window frame is now fixated in space and the state changes to the *framing* one (Figure 2b). In this state, a window frame is rendered whose diagonal is dynamically specified by the first corner and the current position of the tip of the index finger. That is, the opposite corner follows the tip of the index finger, like a rubber band (Figure 3c). Note that a slightly darker lighting condition is used for the primary view to indicate that the secondary view is currently active. When the user touches tips of the index finger and thumb again (Figure 3d), creation process of the tunnel-window is completed and its position and size are finalized. Now the viewpoint inside the frame becomes independent from the primary viewpoint and the state changes to the *waiting* one (Figure 2c).

If a user makes a pose of a pair of scissors around the window frame, the tunnel-window is deleted (Figure 2f).

4.2.2 Operation examples using a tunnel window

In the waiting state, now the user is able to perform whatever operations, which are available to the primary view, to the secondary view in the window frame. The secondary viewpoint in the tunnel window is initially identical to the primary viewpoint. So, the user needs to change a viewpoint of either inside or outside the window frame first, for subsequent operations. In Figure 3e-f, the user changes his primary viewpoint by a string shot (waving away). A slightly darker lighting condition is used for the secondary view in this case to indicate that the primary view is currently active. In Figure 3f, the same objects (cone and cube) are visible both inside and outside the window frame. In the same way, a user in Figure 4 creates a tunnel window in a CAVE setup (Figure 4a) and changes his primary viewpoint by a skewer (Figure 4b). In the default condition, the window frame is anchored to the world (physical) coordinates. So, user's physical movement in the CAVE display affects the viewing frustum defined by the user's eyes and the window frame, which further affects the view through the tunnel window. Variations of the coordinate systems are discussed in Section 5 in detail.

Once a remote scene is shown in the secondary view, short and middle range operations can now be applied for performing long range operations to the remote scene. Figure 3g-o and Figure 4c-i illustrate a number of operations using a secondary view. Figure 3g-h and Figure 3i show examples of skewer operations for a movable object (cone) and a fixed object (wall), respectively.

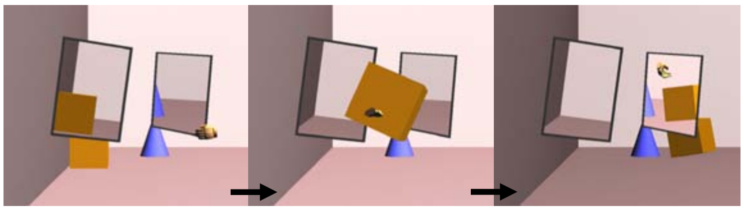
Figure 5: A teleportation operation between multiple remote scenes

If a grabbed or a hooked object passes through the window frame, it is instantly moved between scenes. Figure 3j-l and Figure 4c-f illustrate a teleportation operation using a string shot (beckoning) and a skewer, respectively. When the manipulated object is near the window surface, portion of the object is rendered semi-transparently to visualize seamless teleportation (see Figure 3k and Figure 4d-e). By using a teleportation operation sequentially, an object can be moved even between multiple remote scenes seamlessly without changing user's position in the primary scene. In Figure 5, a cube is moved from the left tunnel window to the right one. Similarly, if the user's viewpoint passes through the window frame his or her viewpoint teleport to the other side. This is done either by moving the window frame toward the user (Figure 3m-o), or by user's moving toward the window frame (Figure 4g-i). The window frame the user has just passed through remains behind him or her. Note that all functions of the primary and secondary scenes interchange after user's teleportation. In this way, a user can perform a variety of teleportation operations in a large-scale virtual environment.

As shown in the last example, a user is able to grab and move a window frame. The state then changes to the *holding* one (Figure 2d) until he or she releases the frame (Figure 2e). Normally, the secondary scene is fixed to the world (physical) coordinates and movement of the window frame has no effect on the relative position and orientation between the user and the secondary scene. Therefore, moving the frame results in showing a different area of the secondary scene as illustrated by Figure 3m-n. Variations in combinations of transformation matrices among several coordinate systems will be discussed in Section 5.

4.3 Implementation

4.3.1 Gesture recognition

Our gesture recognition algorithm is simple and straightforward. The system recognizes each hand gesture by comparing user's current hand shape with N registered hand shapes frame by frame. For each registered hand gesture G_i $(1 \leq i \leq N)$, an error E_i is calculated which is the sum of angular distances of all joint angles between i-th hand shape and user's current shape. By choosing the minimum error E_k among N errors, the current hand shape is recognized as k-th registered hand shape when E_k is smaller than a threshold D_{k1}. Otherwise, the hand shape is not associated with any registered hand shapes, i.e., the system issues no gesture commands. Once recognized as a registered hand shape, the user's hand shape will be kept recognized as the same registered hand shape until the error E_k becomes larger than D_{k2} ($>D_{k1}$). This hysteresis alleviates unintentional state transitions, for example, between grabbing and releasing. However, this also introduces unwanted drag of an object when releasing. To compromise, a released object is placed at the last hand position whose error E_k was smaller than D_{k1}, not D_{k2}.

4.3.2 Coordinate systems

Because our system needs to handle the primary and (possibly multiple) secondary scenes equivalently in terms of both object manipulation and user navigation, each coordinate system is defined based on the world (physical) coordinate system W for simplicity. For example, by using a transformation matrix U from W to the user coordinate system, a position X_U in the user coordinate system is described as UX_U in W. Similarly, P, S_i and F_i are defined as transformation matrices from the world coordinate system W to those of the primary scene, i-th secondary scene and i-th window frame, respectively. By using these transformation matrices, a variety of operations are described in a simple notation. For example, an object at the location X_{Si} in i-th secondary scene' coordinate system (which appears at S_iX_{Si} in the world coordinate system W) is teleported to a new location $X_{Sj} = S_j^{-1}S_iX_{Si}$ in j-th secondary scene's coordinate system by passing through the i-th and j-th window frames. Similarly, a user at the location X_P in the primary scene is teleported to another place $X_{Si} = S_i^{-1}PX_P$ in i-th secondary scene.

4.3.3 Rendering

There are many studies that deal with multiple viewport areas for different 3D views in a single screen including 3D Magic Lenses (Viega, Conway, Williams & Pausch, 1996) and SEAMS (Schmalstieg & Schaufler, 1999) using a set of standard features of today's graphics hardware, such as a stencil buffer and a clipping plane. First, the primary scene is rendered without any tunnel windows. Second, colour and depth buffers of each visible portion of every tunnel window are filled with the background colour and the infinity value, respectively. Third, each of the secondary scenes for every tunnel window is rendered in the corresponding window frame using stencil mask and clipping plane features. Stencil mask is used for rendering a secondary scene only in the visible portion of the corresponding tunnel window, while clipping plane is used for rendering the scene only behind the window surface. The entire rendering process is repeated twice for stereo viewing.

5 Variations of the transformation hierarchy

As explained in Section 4.3.2, our system mainly exploits four types of transformation matrices U (the user coordinates), P (the primary scene's coordinates), F (the window frame's coordinates) and S (the secondary scene's coordinates). Conceptually, the user coordinates can be further decomposed into several coordinate systems, e.g., coordinate systems of user body, head, hand, and so on. In our implementation, the user coordinate system is defined by the translation component (not rotation) of the user's head in the world coordinate system. The entire transformation hierarchy is primarily composed of U, P, F and S with the world coordinate system W at its root. In the initial condition, all of them are direct children of the root (W). In this condition, movement of neither the user (locomotion in the CAVE display), the primary scene (navigation), the window frame (relocation) and the secondary scene (navigation) affects another entity's location. However, a variety of combinations of transformation matrices in the hierarchy are conceivable each having different impacts on user interface design. In this section a thorough discussion of variations in the transformation hierarchy is given including characteristics and application scenarios for typical matrix combinations.

5.1 Coordinate systems in 3D user interface

3D information can be anchored to any of the coordinate systems in the transformation hierarchy in a 3D user interface. Billinghurst, Bowskill, Jessop & Morphett (1998) proposed *head-stabilized*, *body-stabilized* and *world-stabilized* information displays. Head-stabilized information stays in the user's view regardless of the user's head motion. Body-stabilized information stays in the same position relative to the user's body. World-stabilized information stays in the world coordinates. They claim each type of information display has its own advantages and suitable application domains. For example, wearable computers with a head mounted display usually provide head-stabilized information to a user, while augmented reality systems usually attempt to provide world-stabilized information. On the other hand, plenty of interaction techniques for a virtual environment are body-stabilized, including World-in-Hand, WIM (Stoakley, Conway & Pausch, 1995), Mine's techniques (Mine, Brooks, & Sequin, 1997). Piekarski et al discussed world-relative, location-relative, body-relative and head-relative interactions for outdoor geometry modelling in augmented reality (Piekarski & Thomas, 2004).

As one of most relevant studies, Stoev et al have already discussed taxonomy of the Through-The-Lens techniques and proposed nine types of transformation combinations of a window frame coordinates and the secondary scene's coordinates (Stoev & Schmalstieg, 2002). In their work, a window frame can either be anchored to the user's view, the primary world, or the user's hand. The secondary world can either be anchored to the primary world, the window frame, or the world coordinate system. We have extended this scheme so that each of the coordinate systems of the user, the primary scene, the secondary scene and the window frame can be a child of any other coordinate systems in the transformation hierarchy. Each combination is expected to provide different impacts on user interface design.

5.2 Discussion on pairwise transformation matrices

Theoretically, the entire transformation hierarchy has as many as 125 patterns; 1, 12, 48 and 64 patterns for those patterns that have four, three, two and one direct root's children, respectively. A tree structure of the transformation hierarchy here is denoted by a form with commas and parentheses, like a function call. For example, W(U, P, F, S) stands for a situation where all of the entities are direct children of the world coordinate system and W(U(F(S)), P) stands for a situation where the secondary scene is anchored to the window frame which is anchored to the user. Table 3 shows typical 25 patterns of the transformation hierarchy. Among all of 16 parent-child combinations, U(P), P(U), F(U) and S(U) are omitted because they are relatively less useful. In the following, all of 16 parent-child combinations between two coordinate systems will be discussed to examine characteristics of each pattern of the transformation hierarchy in more detail.

Table 3: 25 patterns of the transformation hierarchy.

a) W(U, P, F, S)	b) W(U, F(P), S)	c) W(U, F, S(P))	d) W(U(F), P, S)	e) W(U, P(F), S)
f) W(U, P, S(F))	g) W(U(S), P, F)	h) W(U, P(S), F)	i) W(U, P, F(S))	j) W(U, S(P, F))
k) W(U(F), S(P))	l) W(U, S(P(F)))	m) W(U, S(F(P)))	n) W(U, F(P, S))	o) W(U(S), F(P))
p) W(U, F(P(S)))	q) W(U, F(S(P)))	r) W(U(F, S), P)	s) W(U, P(F, S))	t) W(U(F), P(S))
u) W(U(S), P(F))	v) W(U(F(S)), P)	w) W(U(S(F)), P)	x) W(U, P(F(S)))	y) W(U, P(S(F)))

5.2.1 The primary scene

The primary scene's coordinate system can have four types of parents as briefly explained below.

- W(P): The primary scene is a child of the world coordinate system. In this case, the primary scene stays in the physical environment. This is normal for a VR system to provide presence in an immersive environment. In Table 3, patterns a), d), e), f), g), h), i), r), s), t), u), v), w), x) and y) have this relationship.
- U(P): The primary scene is a child of the user coordinate system. This condition is rarely useful because user motion in the CAVE display does not affect views of the primary scene. In Table 3, no pattern has this relationship.
- F(P): The primary scene is a child of the window frame. As shown in Figure 6, World-in-Hand technique is available in this condition. The window frame in this case functions as a handle of the primary scene. When there is no good target in the primary scene which is a fixed object, a user is still able to change his or her viewpoint of the primary scene by creating a tunnel window in this condition. Notice that the viewpoint in the secondary scene does not change in this case. In Table 3, patterns b), m), n), o) and p) have this relationship.

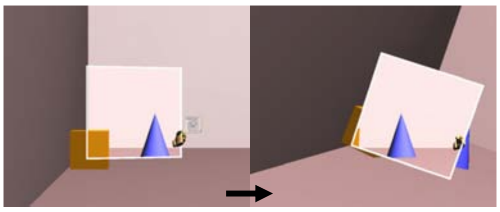
Figure 6: World-in-Hand in F(P) condition

- S(P): The primary scene is a child of the secondary scene. In this case, the primary scene is anchored to the secondary scene, but not vice versa. This condition is not used often, but is useful when the user wants to adjust the primary scene to the secondary scene. In Table 3, patterns c), j), k), l) and q) have this relationship.

5.2.2 The window frame

The window frame's coordinate system can have four types of parents as briefly explained below.

- W(F): The window frame is a child of the world coordinate system. The window frame stays in the same position relative to the CAVE display regardless of the user's motion and navigation. The user can place window frames directly in the physical environment regardless of any other virtual objects. This means the user can layout a virtual multi-screen display around the user to build a 3D user interface. In Table 3, patterns a), b), c), g), h), i), n), o), p) and q) have this relationship.

- U(F): The window frame is a child of the user. The window frame stays in the same position relative to the user regardless of user's motion and navigation (Figure 7). The user can 'carry' the window frame hands-free and keep it within reach. This condition enables a variety of body-relative interaction techniques with the window frame. In Table 3, patterns d), k), r), t) and v) have this relationship.
- P(F): The window frame is a child of the primary scene. The window frame stays in the same position in the primary scene (Figure 8). This condition is useful when the user wants to associate a tunnel window with a certain location in the primary scene. For example, the user can create a wormhole in a virtual kitchen connected to a supermarket, or a virtual library connected to a virtual study room. In Table 3, patterns e), l), s), u) and x) have this relationship.
- S(F): The window frame is a child of the secondary scene. This condition is rarely useful because user navigation of the second scene results in drastic movement of the window frame and visible portions of the secondary scene seen through the frame, making user navigation even more difficult in the secondary scene. In Table 3, patterns f), j), m), w) and y) have this relationship.

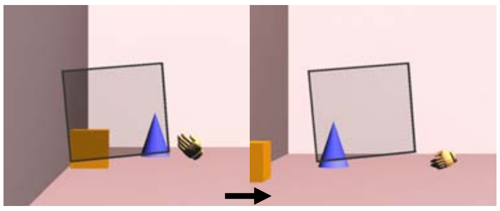
Figure 7: A window frame in U(F) condition

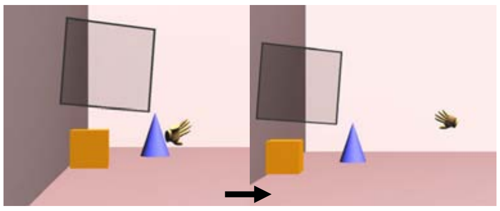
Figure 8: A window frame in P(F) condition

5.2.3 The secondary scene

The secondary scene's coordinate system can have four types of parents as briefly explained below.

- W(S): The secondary scene is a child of the world coordinates. In this condition, the secondary scene stays in the physical environment regardless of the position of the window frame (Figure 9). The window frame functions as a looking glass to the secondary scene. In Table 3, patterns a), b), c), d), e), f), j), k), l) and m) have this relationship.
- U(S): The secondary scene is a child of the user. This condition is similar to W(S) in a sense that the user is able to keep almost the same view of the secondary scene while navigating in the primary scene. When the user moves in the CAVE display, W(S) gives different views, but U(S) gives the same view of the secondary scene. In Table 3, patterns g), o), r), u) and w) have this relationship.
- P(S): The secondary scene is a child of the primary scene. This condition is useful when the user wants to adjust relative position of the secondary scene to the primary scene. The relative position is kept constant regardless of the user navigation of the primary scene. A tunnel window can function as 3D Magic Lens. For example, in a history education scenario, children can compare modern and ancient city models at the same location. In Table 3, patterns h), p), s), t) and y) have this relationship.
- F(S): The secondary scene is a child of the window frame. In this condition, the window frame functions as a handle for the World-in-Hand operation to the secondary scene (Figure 10). The user sees the same view of the secondary scene as long as relative position and orientation of the window frame to the user are the same. In Figure 3, patterns i), n), q), v) and x) have this relationship.

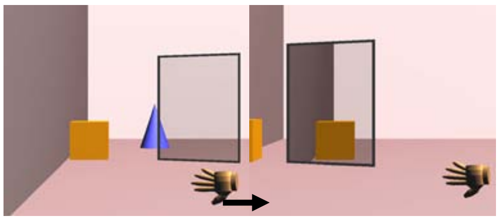
Figure 9: A secondary scene in W(S) condition

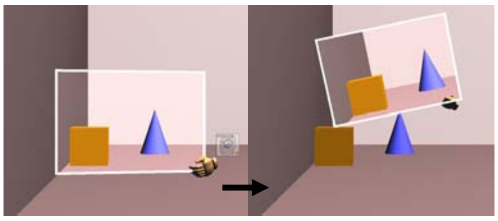
Figure 10: A secondary scene in F(S) condition

5.2.4 The user

Normally, the user is a child of the world coordinate system, that is denoted as W(U). Other conditions, i.e., P(U), F(U), and S(U) have little meaning unless we consider a multi-user setup. In the condition P(U), for example, the user is moved in the same way as the primary scene when it is moved by another user.

6 Conclusions and future work

This paper has proposed a *tunnel window*, a versatile three dimensional interaction technique, for both user navigation and remote object manipulation in a large scale virtual environment. A tunnel window, which is created on demand at an arbitrary position in space using a set of hand gesture commands, provides a secondary view whose viewpoint can be controlled independently from that of the primary view. A tunnel window allows various kinds of seamless teleportation operations including teleportation of the user's viewpoint and that of a virtual object from one position to another by simply passing through the window frame. The tunnel window scheme has been further extended by considering variations of the transformation hierarchy among coordinate systems of the user, the local and remote scenes, and the window frame. Every pairwise combination of parent-child relationship among those four types of coordinate systems has been discussed from a perspective of user interface design. Future work includes conducting rigorous user studies to further examine characteristics of each pattern of the transformation hierarchy, exploiting two-handed interactions, multi-user support, and building practical large-scale virtual environments.

References

Billinghurst, M., Bowskill, J., Jessop, M. & Morphett, J. (1998). A Wearable Spatial Conferencing Space. *Proceedings of the IEEE International Symposium on Wearable Computers*, 76-83.

Bowman, D. A., Koller, D. & Hodges, L. F. (1998). A Methodology for the Evaluation of Travel Techniques for Immersive Virtual Environments. *Virtual Reality Research Development and Applications*, 3 (2), 120-131.

Mine, M. R., Brooks, F. P. & Sequin, C. H. (1997). Moving Objects in Space: Exploiting Proprioception in Virtual-Environment Interaction. *Proceedings of the ACM SIGGRAPH 1997*, 19-27.

Piekarski, W & Thomas, B. H. (2004). Augmented Reality Working Planes: A Foundation for Action and Construction at a Distance. *Proceedings of the IEEE/ACM International Symposium on Mixed and Augmented Reality*, 162-171.

Poupyrev, I., Billinghurst, M., Weghosrt, S., & Ichikawa, T. (1996). The Go-Go Interaction Technique: Non-Linear Mapping for Direct Manipulation in VR. *Proceedings of the ACM Symposium on User Interface Software and Technology*, 79-80.

Schmalstieg, D. & Schaufler, G. (1999). Sewing Worlds Together With SEAMS: A Mechanism To Construct Large-Scale Virtual Environments. *MIT Press Presence – Teleoperators and Virtual Environments*, 8(4), 449-461.

Stoakley, R., Conway, M. J. & Pausch, R. (1995). Virtual Reality on a WIM: Interactive Worlds in Miniature. *Proceedings of ACM CHI 1995*, 265-272.

Stoev, S. L. & Schmalstieg, D. (2002). Application and taxonomy of through-the-lens techniques. *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, 57-64.

Tomozoe, Y., Machida, T., Kiyokawa, K. & Takemura, H. (2004). Unified gesture-based interaction techniques for object manipulation and navigation in a large-scale virtual environment. *IEEE Virtual Reality 2004*, 259- 260.

Viega, J. Conway, M. J., Williams, G. & Pausch, R. (1996). 3D Magic Lenses. *Proceedings of the ACM Symposium on User Interface Software and Technology*, 51-58.

Ware, C. (2000). Information Visualization: Perception for Design. San Fancisco: Morgan Kaufmann.

Ware, C. & Osborne, S. (1990). Exploration and Virtual Camera Control in Virtual Three Dimensional Environments. *Symposium on Interactive 3D Graphics*, 175-183.