HAND-EYE COORDINATION IN VIRTUAL REALITY, USING A DESKTOP DISPLAY, STEREO GLASSES, AND A 3-D MOUSE

THEODORE W. HALL
Department of Architecture, Chinese University of Hong Kong, Sha Tin, New Territories, Hong Kong <twhall@cuhk.edu.hk>

Abstract. Many virtual-reality displays augment the user's view of the real world but do not completely mask it out or replace it. Intuitive control and realistic interaction with these displays depend on accurate hand-eye coordination: the projected image of a 3-D cursor in virtual space should align visually with the real position of the 3-D input device that controls it. This paper discusses some of the considerations and presents algorithms for coordinating the physical and virtual worlds.

1. Introduction

Contrary to Hollywood fantasies, virtual reality is not an out-of-body experience. Humans experience inertia and gravity and have a proprioceptive sense of body posture that is independent of vision. Moreover, many virtual-reality displays augment the user's view of the real world but do not completely mask it out or replace it. Thus, intuitive control and realistic interaction with virtual reality depend on accurate hand-eye coordination: the projected image of a 3-D cursor in virtual space should align visually with the real position of the 3-D input device that controls it.

This paper draws from experience in developing virtual reality software for a desktop display, field-sequential stereo glasses, and a 3-D mouse. Such a system offers several advantages over head-mounted displays. Projecting the display onto a large wall-mounted screen has proven to be particularly effective: it provides a large view volume, a wide cone of vision, and more pixels than most head-mounted systems. One important distinguishing feature of desk-mounted and wall-mounted displays is that the user's real hands and input devices remain visible in conjunction with the virtual world. Thus, spatial correspondence of the physical and virtual worlds is critical to developing a natural interface.

The driving application for this research is a project sponsored by a major electric utility, to develop a training simulator for operating transmission equipment. Other applications include a simulation of Song Dynasty wood construction, and a sketching program that allows the user to place points, lines, and polygons in three dimensions.

2. System Components

This paper focuses on non-head-mounted display systems – typically desk-mounted or wall-mounted. These differ from head-mounted displays in three important respects:

- They project the left and right views on to a single stationary display that is not centered in the user's field of vision. With two eyes and only one display, it is certain that at least one eye has an off-center view. Since eye spacing is critical to stereo vision and depth perception, this off-centeredness is significant; the perspective computations must account for it.
- They rely on special eye wear to filter the display into separate left and right views – by synchronized field-sequential shuttering, orthogonal polarization, or other means. Imperfections in this filtering will partially expose a view to the wrong eye.
- They do not block the view of the real world between the user's eyes and the display screen. In particular, the user's real hands and 3-D input devices remain visible in conjunction with the projected virtual world.

Three-dimensional spatial interaction depends on tracking the position of a movable pointing device relative to a stationary reference device. Accurate hand-eye coordination depends on tracking the head as well as the hands. The tracking system uses ultrasonic pulses, electromagnetic fields, or other means to determine the pointers' (xt, yt, zt) translations and $(roll, pitch, yaw)$ rotations in a coordinate system aligned with the reference. Higher-level application software must transform these coordinates from the tracker reference to the stereo perspective display and ultimately to the virtual world.

In the remainder of this paper, "desktop display" is short-hand for any non-head-mounted display. The discussion applies as well to wall-mounted displays. "Stereo glasses" are the eye wear for filtering the display into separate left and right views. These must also be equipped for position tracking. "3-D mouse" refers to a hand-held pointing device in a three-dimensional position tracking system. This may be a simple point-and-click device similar to the common 2-D mouse, or it may be incorporated into a more sophisticated system such as a data glove.

3. Calibrating the Tracking System

The first step toward achieving any visible correspondence between the physical pointer and the virtual cursor is to calibrate the position tracking system relative to the display. This presumes that the tracker's reference device is mounted in a stable position relative to the display, and that the display is within the tracker's operating range. The precise position of the tracker relative to the display is arbitrary, and generally involves rotation and translation along all three axes. Moreover, the display hardware itself may be maladjusted, resulting in slightly different scale factors (pixels per millimeter) on the x and y axes.

The following discussion assumes right-handed coordinate systems for both the tracking system and the display. The display plane is $z = 0$, with the x axis horizontal, the y axis vertical, and the z axis normal to the display. The formulas define matrices to post-multiply row vertices. (To pre-multiply column vertices, transpose the rows, columns, and order of multiplication throughout.)

3.1. ROTATION MATRICES

Rotation matrices play an important part in mapping the pointer's position from the tracking system to the display and the virtual world. Before beginning a detailed description of the calibration procedure, it's useful to have a general expression for these matrices.

A rotation matrix to *roll* about the z axis, then *pitch* about the x axis, then *yaw* about the y axis, has the following form:

$$\mathbf{R} = \begin{bmatrix} R_{1,1} & R_{1,2} & R_{1,3} \\ R_{2,1} & R_{2,2} & R_{2,3} \\ R_{3,1} & R_{3,2} & R_{3,3} \end{bmatrix}$$

$$\begin{aligned}
R_{1,1} &= \cos(roll)\cdot\cos(yaw) + \sin(roll)\cdot\sin(pitch)\cdot\sin(yaw) \\
R_{1,2} &= \sin(roll)\cdot\cos(pitch) \\
R_{1,3} &= -\cos(roll)\cdot\sin(yaw) + \sin(roll)\cdot\sin(pitch)\cdot\cos(yaw) \\
R_{2,1} &= -\sin(roll)\cdot\cos(yaw) + \cos(roll)\cdot\sin(pitch)\cdot\sin(yaw) \\
R_{2,2} &= \cos(roll)\cdot\cos(pitch) \\
R_{2,3} &= \sin(roll)\cdot\sin(yaw) + \cos(roll)\cdot\sin(pitch)\cdot\cos(yaw) \\
R_{3,1} &= \cos(pitch)\cdot\sin(yaw) \\
R_{3,2} &= -\sin(pitch) \\
R_{3,3} &= \cos(pitch)\cdot\cos(yaw)
\end{aligned}$$

3.2. MAPPING THE TRACKER TO THE DISPLAY

In theory, three points are sufficient to determine the position of a plane in space. However, few computer displays are truly planar – most are cylindrical or spherical. It's difficult to position three points on such a surface that wouldn't result in a pitched plane – one point tends to bulge out more than the other two.

This algorithm uses four points, located symmetrically around the display at the center of each edge. These points define horizontal and vertical baselines, parallel to the x and y display axes. The vector cross product of the x and y axes determines the orientation of the z axis and the ideal display plane. It's likely that all four points don't lie on a single plane (due to the curvature of the display), but the only uncertainty is in the plane's translation along the z axis. The algorithm defines the location of the plane as the average z translation of the four points.

For each of the four display points, the procedure draws a target at predetermined display coordinates and prompts the user to touch and click with the 3-D mouse. It reads the mouse coordinates reported by the tracker and uses them to compute a transformation from the tracker to the display.

First, it rolls about the z axis, to align the tracker x coordinates of the bottom and top points.

Then, it pitches about the x axis, to align the tracker z coordinates of those points. The baseline connecting them is now vertical (aligned with the y axis), and the baseline connecting the left and right points is now horizontal, in the x-z plane.

Next, it yaws about the y axis, to align the tracker z coordinates of the left and right points. The transformed tracker coordinate axes are now parallel to the display axes.

After aligning the axes, the procedure computes scale factors for converting from tracker units (real-world millimeters) to display units (pixels). Display system software (such as the X Window System) often provides functions for querying the width and height of the display in both pixels and millimeters, and one might expect to be able to compute scale factors from these. Unfortunately, the display hardware is often significantly maladjusted relative to the nominal resolution reported by these system functions. A convenient consequence of calibrating the tracker relative to the display is that it allows one to compute accurate display scale factors, based on direct measurement of the real display size by the 3-D mouse.

Finally, the procedure subtracts the rotated and scaled tracker coordinates from the display coordinates to determine the translations. It distributes any residual error across the display by computing x, y, and z translations for each of the four target points and keeping the averages.

The result of the calibration is a matrix that converts the mouse's xt, yt, and zt translations from the tracker reference to the display. In the following, the subscript mt denotes coordinates of the mouse relative to the tracker, td denotes coordinates of the tracker relative to the display, and md denotes coordinates of the mouse relative to the display.

$$\begin{bmatrix} xt_{md} & yt_{md} & zt_{md} & 1 \end{bmatrix} =$$

$$\begin{bmatrix} xt_{mt} & yt_{mt} & zt_{mt} & 1 \end{bmatrix} \times \begin{bmatrix} xs \cdot R_{1,1} & ys \cdot R_{1,2} & zs \cdot R_{1,3} & 0 \\ xs \cdot R_{2,1} & ys \cdot R_{2,2} & zs \cdot R_{2,3} & 0 \\ xs \cdot R_{3,1} & ys \cdot R_{3,2} & zs \cdot R_{3,3} & 0 \\ xt_{td} & yt_{td} & zt_{td} & 1 \end{bmatrix}$$

The R terms constitute a rotation matrix $\mathbf{R}_{td}$ formed from the *roll*, *pitch*, and *yaw* of the tracker relative to the display. The xs, ys, and zs terms are the scale factors (pixels per millimeter). Since 2-D displays don't have a z coordinate, the algorithm arbitrarily sets the z scale equal to the x scale, and places the z origin at the hypothetical display plane. The xt, yt, and zt terms are translations.

3.3. MAPPING THE 3-D MOUSE TO THE DISPLAY

The tracker reports the mouse's xt, yt, zt, *roll*, *pitch*, and *yaw* coordinates relative to its reference device. The calibration matrix transforms xt, yt, and zt from tracker to display coordinates, but it does not directly transform *roll*, *pitch*, and *yaw*. The algorithm computes these angles in display space from the components of rotated unit basis vectors.

In the 3-D mouse's coordinate system, its basis vectors are:

$$\mathbf{i} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$
$$\mathbf{j} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$$
$$\mathbf{k} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

These are simply the rows of a 3 x 3 identity matrix. Rotating these vectors is analogous to pre-multiplying a rotation matrix by an identity. The result is simply the rotation matrix:

$$\begin{bmatrix} \mathbf{i}_{mt} \\ \mathbf{j}_{mt} \\ \mathbf{k}_{mt} \end{bmatrix} = \begin{bmatrix} R_{1,1} & R_{1,2} & R_{1,3} \\ R_{2,1} & R_{2,2} & R_{2,3} \\ R_{3,1} & R_{3,2} & R_{3,3} \end{bmatrix} = \mathbf{R}_{mt}$$

Here, the *roll*, *pitch*, and *yaw* in the matrix formulae are the mouse's rotations relative to the tracker. Multiply this matrix by the tracker's rotation relative to the display, to obtain the mouse's rotation relative to the display. Finally, compute new *roll*, *pitch*, and *yaw* values from the components of the rotated basis vectors.

To summarize, the mouse's basis vectors relative to the display are the rows of a rotation matrix. This matrix is the product of the mouse's rotation relative to the tracker, and the tracker's rotation relative to the display. The former matrix derives from the angles reported by the tracker, and varies with the position of the mouse. The latter matrix results from the calibration procedure, and is constant as long as the tracker reference remains firmly fixed relative to the display. The product of the rotation matrices is:

$$\begin{bmatrix} \mathbf{i}_{md} \\ \mathbf{j}_{md} \\ \mathbf{k}_{md} \end{bmatrix} = \begin{bmatrix} i_x & i_y & i_z \\ j_x & j_y & j_z \\ k_x & k_y & k_z \end{bmatrix} = \mathbf{R}_{md} = \mathbf{R}_{mt} \times \mathbf{R}_{td}$$

The components of the rotated basis vectors determine the *roll*, *pitch*, and *yaw* of the mouse relative to the display. In general:

$$\begin{aligned} roll_{md} &= \tan^{-1}\left(i_y / j_y\right) \\ pitch_{md} &= \tan^{-1}\left(-k_y \Big/ \sqrt{k_x^2 + k_z^2}\right) \\ yaw_{md} &= \tan^{-1}\left(k_x / k_z\right) \end{aligned}$$

In the special case that i_y and j_y are both zero, or (equivalently) k_x and k_z are both zero, then the pitch is $\pm\pi/2$ (opposite to the sign of k_y). The roll can be set to zero, and the yaw can be computed from i_z and i_x:

$$\begin{aligned} roll_{md} &= 0 \\ pitch_{md} &= \begin{cases} \pi/2 \text{ if } k_y = -1 \\ -\pi/2 \text{ if } k_y = 1 \end{cases} \\ yaw_{md} &= \tan^{-1}\left(-i_z / i_x\right) \end{aligned}$$

4. Stereo Perspective Projection

In essence, stereoscopic perspective is simply a double application of the well-known procedure for monoscopic perspective. Extend rays from a viewing point through points in a scene. The intersections of these rays with a projection plane define the perspective projection of the scene on that plane. The ray from the viewing point perpendicular to the plane defines the center of projection. Computed perspective derives algebraically from the proportions of similar triangles: it scales width and height from the center of projection, according to the ratio of distances from the viewing point to the projection plane and the point in the scene.

There is nothing special about the center of vision (or center of attention). It can pan across the scene, from one ray to another, without any effect on the intersections of the rays with the plane – provided that the viewing point, the object, and the projection plane remain fixed in place. There is no requirement that the center of vision remain perpendicular to the projection plane.

Stereo vision depends on projecting simultaneously through two distinct viewing points: the eyes. The eyes rotate to converge their centers of vision on a point of interest. Computer displays (typically) do not rotate to match this convergence – this is true even for head-mounted displays. Thus, the center of vision is often not centered on the display, and the "line of sight" is often not perpendicular to it.

There is a temptation to compute stereoscopic perspective on the basis of a rotation between the two views, as shown in Figure 1. This assumes that the eyes' centers of vision are also their centers of projection, resulting in two distinct, incongruous projection planes. This procedure is incorrect for any system that ultimately renders the left and right views on a single shared display. Rotating the scene is analogous to rotating the projection planes relative to the scene. This leads to vertical parallax between the two projections that should never occur as long as the eyes are level and equidistant from the display. Points in the plane of the display, which should project to the same pixel in both views, project to two different pixels. The error is small near the center, but becomes increasingly severe near the edges. These discrepancies confound depth perception and hinder hand-eye coordination in the virtual world. The rotation algorithm ignores the fact that the display itself is viewed in different perspectives by the two eyes. To argue that this is insignificant is to belie the very underpinnings of stereoscopic vision.

Figure 2 shows the correct algorithm for stereoscopic perspective projection. Both views project onto a common plane, with only a horizontal offset between them. If one assumes a symmetric view frustum, then one must enlarge the horizontal angle and clip one side, to conform to the display boundaries from the off-center viewing point. Alternatively, one can modify the equations for perspective projection to account for the asymmetry of the frustum.

5. Zooming and Scaling in Stereo

It's desirable to control the apparent depth of the stereo image relative to the display surface, to avoid large discrepancies between optical focus and binocular convergence, as well as to insure that interactive elements in the

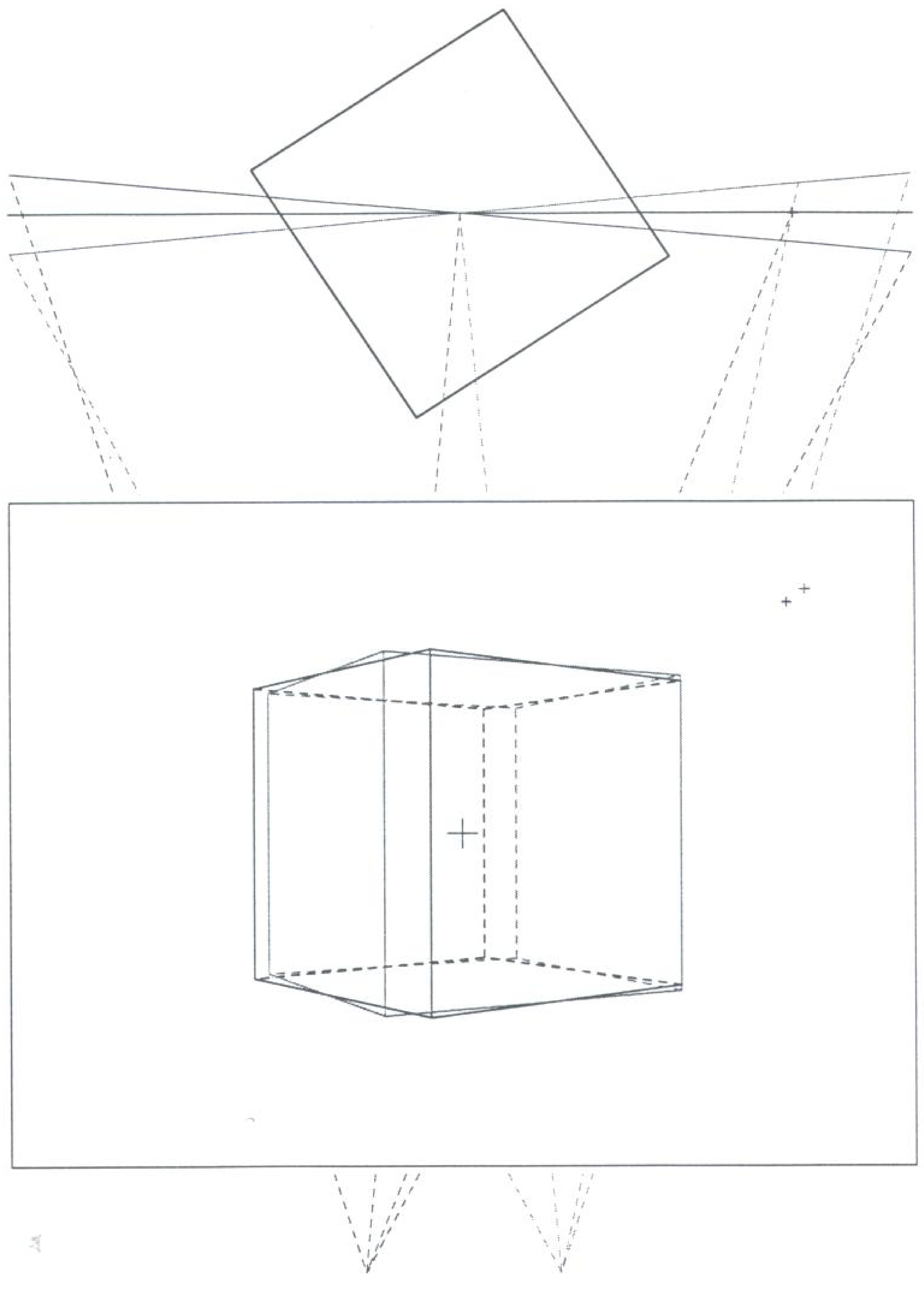

Figure 1. Incorrect construction of stereoscopic perspective via rotation.

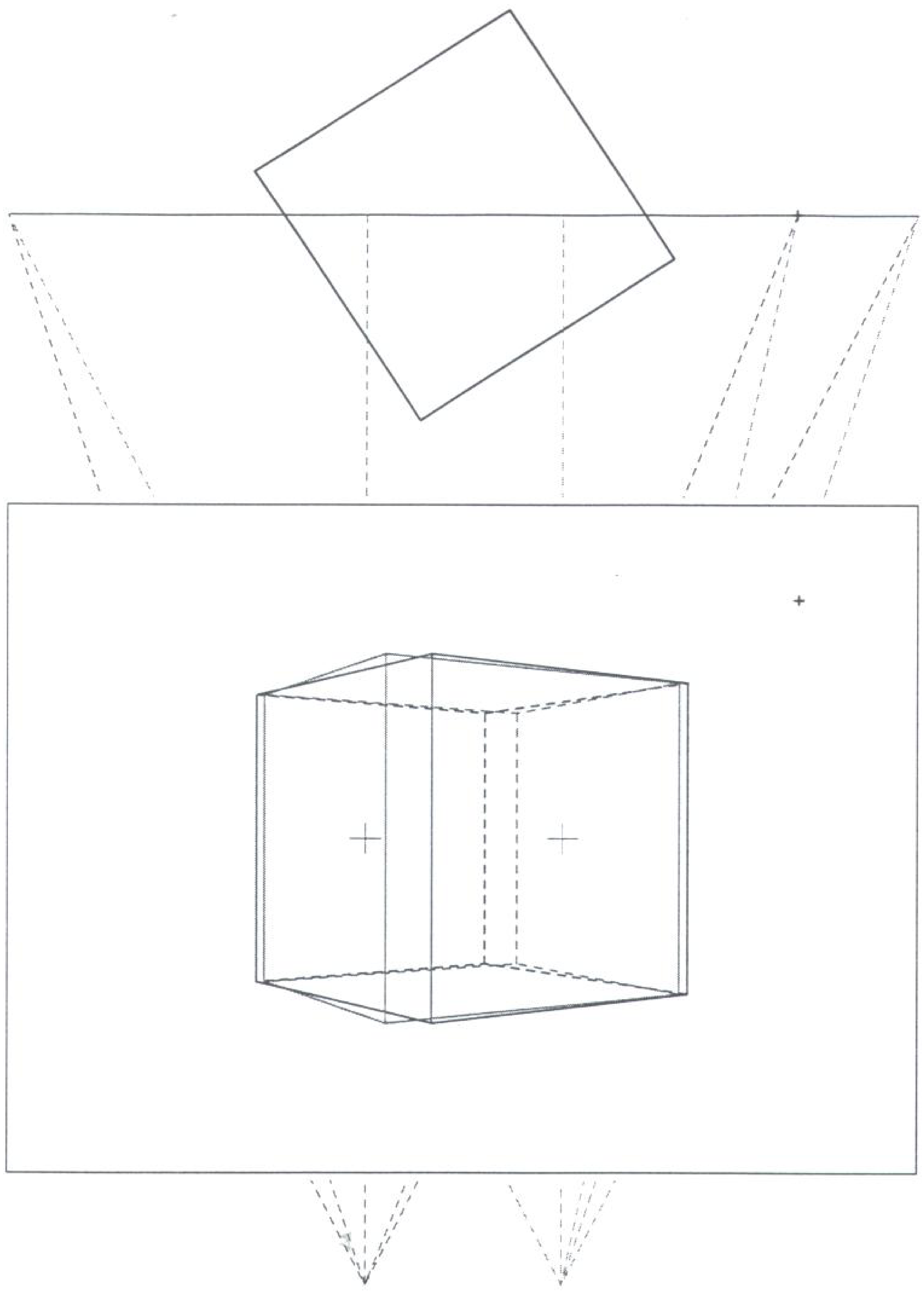

Figure 2. Correct construction of stereoscopic perspective via translation.

virtual world appear in front of the display. (The mouse can't penetrate the surface of the display to interact with virtual objects that appear farther away.)

A single scale factor, applied to *x*, *y*, and *z*, maps the physical space of the user's eyes and computer display to the virtual space of stereo cameras and projection plane. This same scale factor, applied to the offset between the user's eyes (64 millimeters, on average) determines the offset between the stereo cameras. This maps an arbitrarily large scene to an arbitrarily small display. The scene appears as a small model near the display plane, rather than a large model far behind it. The distance perception depends on the appropriate scaling of the camera offset.

Problems may arise in trying to interact with a small image of a large object: the interactive elements may be too small to resolve, let alone control. Merely moving the stereo cameras closer to the object, without changing the scale of the virtual world, does not solve the problem. Though it enlarges the image, it also pulls it away from the display. The result still appears as a small object, but now uncomfortably close to the user's face.

If the projection plane remains constant, then moving the stereo cameras closer must change the scale from physical to virtual space (assuming the user's eyes remain a fixed distance from the display). To preserve the visual distance of the projected image, the application must scale the offset between the stereo cameras, in proportion to their distance from the projection plane.

6. Conclusions

The success or failure of virtual reality hinges on the veracity of its input, output, and interaction. Accurate hand-eye coordination is critical. It requires some computing power to track and transform the user's hand and head motions in real time. It does not require top-of-the-line displays or tracking devices. Investment in expensive hardware does not guarantee success, and does not compensate for inattention to detail in the setup and programming.

Acknowledgments

This research is being conducted in the Department of Architecture, Chinese University of Hong Kong, under a contract with the China Light & Power Co. Ltd. I wish to thank CUHK and CLP for their continuing support.