Three-dimensional interactive cursor based on voxel patterns for autostereoscopic displays

Vladimir Saveljev a, Jung-Young Son a, Choonsik Yim b and Gwanghee Heo a

aPublic Safety Research Center, Konyang University, Nonsan city, Republic of Korea; bRCN Co., Ltd, Daejeon city, Yuseong-gu, Republic of Korea

ABSTRACT
A three-dimensional (3D) cursor for autostereoscopic 3D displays was proposed. The cursor is based on voxel patterns. In the experiments, the cursor position is interactively controlled in 3D by a computer mouse with a scroll wheel. A computer program operates in the image plane of the 3D display directly without complicated geometric calculations. The results were verified in a 3D display with a parallax barrier and can be applied to other types of autostereoscopic displays.

ARTICLE HISTORY
Received 15 November 2021
Accepted 28 December 2021

KEYWORDS
Autostereoscopic display; multiview image; voxel patterns; 3D interaction; 3D cursor

1. Introduction

Three-dimensional (3D) displays play an important role in modern imaging. Among them, the autostereoscopic displays [1,2] provide a natural perception of the depth with no eye gear. In this paper, we are intentionally limited by this particular type of 3D display. The virtual reality and holographic displays are not discussed; however, specific holographic patterns are discovered [3]. Binocular stereoscopic displays (e.g. polarization or time-sequential) are not considered.

An autostereoscopic 3D display (ASD) provides a ‘3D image without the viewer needing to wear any special viewing gear’, a quote from [2]; similar definitions can be found in [4–6] and elsewhere. In multiview glass-free displays, a binocular image is perceived freely. The position of an observer is not fixed, and there is freedom to move within a certain region. With a glass-free natural viewing, an observer may sit or walk in front of the display. No head/eye tracking is needed.

We consider a specific issue of interactive multiview imaging: how to make a dynamically controlled real-time 3D cursor. We propose the cursor based on patterns in the image plane of ASD, recalling that distinct arrangements consisting of multiple repeated fragments can be recognized in the image plane of various types of ASD [7–9]. As a practical example of such an arrangement, we show a magnified part of the testing image in Figure 1 (note the repeated slanted structures in this image).

These arrangements can be treated as overlaps of certain elementary patterns. The patterns can be systematized so that each pattern would correspond to a fundamental zero-dimensional object, the point in the 3D space (voxel). Thus, we may refer to these patterns as voxel patterns (VP). Strictly, VP is an approximate but accurate enough planar representation of the mathematical 3D voxel. Such a pattern looks like a specific formation (‘constellation’) of the elementary units in the image plane. (See Figure 2 with the patterns ‘extracted’ from the arrangements.)

When browsing 3D photographs or watching a 3D movie, only limited actions are required (e.g. start, stop, pause, next, previous, etc.). Sometimes, static markers (e.g. subtitles) are added. Known methods can implement such common operations because the mentioned actions are not related specifically to 3D.

However, in creating, developing, improving, and modifying 3D images in the 3D space, real-time 3D interaction becomes necessary. The papers [10,11] present reviews on 3D interaction. To implement an interactive control in 3D, a 3D display is needed. In addition, a 3D pointing device (in the operator’s hands) becomes necessary together with a 3D marker, which shows the current position in space. The book [12] covers many related issues, such as an interactively controlled 3D marker. In this context, a marker is a geometrical object (e.g. a line, a circle, an arrow, an alphabetic character, etc.).
For example, in [13], they used a small cube as a marker. A 3D marker should be controlled dynamically, similar to a conventional mouse cursor interactively controlled by a computer mouse in 2D. In this research, we added a controllable cursor to the existing 3D content.

In computer graphics, rendering is a process of creating an image based on the 3D model of the objects and their properties, such as texture. Typically, it is a view of a scene, which is photographed or calculated for a certain point in 3D, based on geometry, texture, lights, etc. In multiview imaging, this process has to be repeated for each view and then the rendered view images are displayed in a proper manner. For instance, some methods use multiple views (e.g. nine views in [13] and four views in [14]); however, to show the marker, they used to render views of the marker at each of its next locations. The rendering of multiple views is a critical, time-consuming and resource-consuming issue of 3D imaging, where special graphics processors are often used [15,16].

To avoid these difficulties and to avoid using special processors, we propose to display the cursor without calculation of rays, visibility, overlap, etc., such as no rendering. We take the prepared patterns and transfer them from the computer memory into the image plane instead of any calculations of that sort. In this approach, there is no need to render multiple views of an object, that is, we avoid complete rendering.

Then, various devices were used to take control in 3D, such as a hand-held light source in the interaction volume [13], a space-ball positioning device [17], a gesture [18], a ‘5D’ pointer [19], etc.

To avoid dealing with unique pointing devices, we extended the paradigm of 2D control, recalling that a regular mouse with a scroll wheel is already a 3D pointing device. The scrolling angle of the wheel can be treated as an additional coordinate. In our approach, it is the distance from the cursor to the screen of a display device (i.e. the ‘depth’ of a 3D image). Therefore, a specially designed 3D pointing device becomes unnecessary, and then the scroll down means ‘forward’ or ‘farther’ from an observer, while the scroll up means ‘backward’ or ‘closer’ to the observer.

In the current paper, we describe a binary 3D cursor with two levels of brightness as an easy example of a 3D object. The cursor was added to the existing 3D images. What is important for the 3D interaction is a fast way to draw the pointer in 3D and a convenient way to control it. Therefore, the rendering of multiple views is substituted by the selection of the known patterns. In addition, we did not need to create any 3D images in order to demonstrate the cursor. It was enough to use the existing ones.

The flowchart of the first (preliminary) part of the algorithm (calculation of the cursor patterns) is shown in Figure 3. This part of the algorithm can be implemented either in run-time or in advance (before the program starts). In either case, we generate the object (cursor) patterns as follows. For each depth plane, VP is copied several times, according to the number of the reference points in the cursor. Each copy is aligned to its reference point of the shape and added (by logical OR function for
Figure 3. Calculation of cursor patterns (a preliminary step of the algorithm).

Figure 4. Depth planes of ASD (horizontal cross-section of both HPO and FP cases; vertical cross-section of FP case). Some voxels are represented by small squares. The neighborhood of 3 voxels is grayed.

2. Voxel patterns

The image in the ASD image plane contains everything about the 3D images displayed in this device. This plane holds information about the positions of all 3D objects, their colors, textures, etc. The properly arranged structure of pixels in the image plane, together with the barrier/lenticular plate, provides the visual 3D picture for an observer. The ASD image plane is typically a flat 2D surface, which is logically split into image cells (one cell per lenticular/barrier element). In this context, the voxels and the depth planes are considered as the most important. A typical layout of ASD is shown in Figure 4.

The voxels lie in the discrete ‘depth’ planes represented by dashed lines in Figure 4. Quadrilaterals are areas around voxels (i.e. neighborhoods). All points from each quadrilateral are represented by one pattern. The voxels lie in the centers of the quadrilateral areas. In even and odd planes, the voxels are displaced laterally by one-half of their period.

The horizontal line at the bottom of Figure 4 shows the observer base at the optimal viewing distance (OVD). The clearest 3D image of the highest quality is perceived from the base. The depth planes lie at the intersections of the rays from the eyes to the light sources (openings in the barrier plate). In this example, the planes are numbered from −5 to 5. The pinholes/slits of the barrier plate are...
the optical equivalents of the microlenses of the lenticular plate. Some planes have special meaning: the image plane, the barrier plate, and the plane next to the x-axis. In our model of the voxel patterns, these planes are merged together.

These basic ideas (discrete depth planes and image cell) make the structure of the voxel patterns clear.

Examples of the patterns were shown in Figure 2. VP can be thought of as a ‘constellation’ of multiple identical fractions (partitions) of the cell. The number of the partitions in the pattern is equal to the modulus of the number of the depth plane. The effective area of VP (the area of the pattern directly involved in imaging [i.e. the sum of areas of all partitions]) is equal to the area of the single image cell. Such pattern in the image plane corresponds to the most fundamental zero-dimensional object, the point in 3D space (strictly speaking, to a neighborhood).

In our model of VP, we use the practically convenient enumeration of the depth planes borrowed from [20]. This enumeration shows the number of partitions; therefore, the planes with 1 partition (image, barrier, and next plane) are merged.

VP defines how a neighborhood of a 3D voxel in the 3D space around the screen (in front and behind it) is mapped onto the 2D image plane of ASD. The details of mapping are explained in [21]. In digital imaging, the pixels are commonly used, and that is why the voxel patterns are sometimes called ‘pixel patterns’.

Any 3D object, such as the 3D cursor described in this paper, can be constructed from voxels. The VPs in the 2D image plane represent the voxels. Then, a 3D object can be assembled from the voxels (technically, from the patterns in the image plane), such as a building made from bricks.

As a result, once we have VPs (read: bricks) with their designed locations in space (read: a sketch or a drawing), we can construct a 3D object (read: building). In this way, multiple rendering is unnecessary. The resulting pattern of the object is drawn over an existing 3D image in the image plane. This makes the cursor a part of a new 3D image that includes the previously existing 3D image plus the ‘added’ cursor.

The most important properties of VPs are as follows [22,23].

The distance from the screen (typically referred to as ‘depth’ of a 3D image) defines the shape of VP. All VPs in the same depth plane have an identical shape. There exists exactly one VP for each depth plane. VP can be displaced laterally by discrete units (image cells) only. Fractional displacement (by a partial or non-integer number of cells) is not allowed.

In the horizontal parallax only (HPO) displays, the width of the cell is \( W \) pixels, the height is 1 pixel. However, for convenience, they sometimes set the height of the HPO cell equal to the height of the screen or the width of the cell. In the full-parallax (FP) displays, the dimensions of the cell (width and height) are \( W \) and \( H \) pixels.

In the HPO case, the pattern for the \( k \)-th discrete depth plane is comprised of the \( |k| \) identical pieces [24] (partitions of the cell) of the size:

\[
W_k = \frac{W}{|k|}
\]  

(1)

The interval between partitions is

\[
W_k = \begin{cases} 
W, & k < 1 \\
W(1 - 2k), & k > 1 
\end{cases}
\]  

(2)

In the FP case, there are \( |k| \) identical partitions, and the size of each is \( W/|k| \) by \( H/|l| \). The interval is the same as in the 1D case. Often, the FP cells are squared (i.e. \( W = H \)); therefore, we may simply say that the \( k \)-th pattern occupies the \( k^2 \) cells. In that case, the area of each partition is \( W^2/k^2 \).

The reference point, which is the point for the alignment of the pattern with the cell and other patterns, is also important. In this paper, we prefer the left upper (LU) corner, but the reference point can be chosen anywhere (in the middle, at the edge, etc.). The reference points in all involved objects should not be changed during processing.

Examples of VPs for the 1st and \( \pm 5 \)th depth planes in HPO and FP displays are shown in Figures 2 and 5, where the cells are shown as ‘empty’ squares, and the partitions as dark rectangles or squares. The reference points of the pattern shown as small circles are not actually parts of the pattern and are not drawn at all.

The difference between the VPs for the positive and negative depth planes is in the location of the partitions within the cells. In the negative depth planes (in front of the screen), the leftmost and rightmost partitions of the VP lie near the outmost edges of the leftmost and rightmost cells, as shown in Figure 5(a), whereas in the positive depth planes (behind the screen), they lie near the innermost edges, as shown in Figure 5(c). The partitions are uniformly distributed across the \( k \) cells in the HPO case and across the \( k^2 \) cells in FP.

Based on these properties, VPs can be built for any depth plane. Let’s explore how the VPs would visually appear for an observer.

ASDs are arranged so that the rays from the corresponding elements of all cells (including the reference
points) go to the same part of the viewing zone [25], as shown in Figure 6. The row of squares in the upper part of the illustration shows pixels of the screen. The row of rectangles at the bottom is comprised of the parts of the viewing zone. The parallax barrier is represented by a bold dashed line. The rays go from the pixels to the corresponding parts of the zone through the openings of the barrier. These rays show the correspondence between the parts and pixels. In this example \( W = 6 \), the rays from the reference points of all cells go to the rightmost part of the zone from the other pixels to the other parts. Therefore, the number of logical parts of the zone is equal to the number of pixels in the cell.

Among the rays, we may keep the rays from the partitions only. This is illustrated for the first plane (pattern is a complete cell), an intermediate plane (pattern of 2 partitions), and the \( n \)-th plane with the maximum number of partitions \( (n = W) \). In Figure 7, the smallest partition is 1 pixel.

The visual voxels lie in the intersections and intersection regions. Among the intersections, the intersection with \( k = 1 \) is closest to the screen, and the intersection with the maximum \( k = W \) is the most distant from the screen.

The intersections of rays are pointed by arrows in Figure 7. All rays in Figure 7(a,c) intersect at a single point. Therefore, an observer with eyes anywhere within the zone (i.e. with any interocular distance) will unambiguously see the voxels at these intersection points. We imply that one part of the viewing zone is smaller than the interocular distances, as in various ASDs. However, in the intermediate voxel \( (k = -2) \) shown in Figure 7(b), there are several intersections within an encircled region. It means that the observers with different interocular distances may see this voxel at slightly different positions.

Nonetheless, with an increased number of partitions, the distance from the screen to the voxels increases from zero in Figure 7(a) to the maximum in Figure 7(c), and the distance is measured from the barrier.

In this extreme case of the maximal depth plane, the visible distance \( h \) from the screen to the visible voxel can be approximately estimated as,

\[
h = \frac{1}{1 + \frac{d}{a_n}}
\]

where \( d \) is the optimal distance to the observer (OVD), \( b \) is the width of the whole viewing zone, and \( a_n = nW_c \) is the width of the maximal pattern occupying the \( n \) cells.

3. Cursor patterns

Once we get the elementary VPs, we can arrange them into a more complicated template (e.g. a pattern of a cursor). This procedure does not require rendering. The absence of rendering means that we did not perform the ray tracing of multiple views of an object. Instead, the prepared patterns were assembled with aligned reference points according to the required shape of an object. The binary cursor templates (with two levels of brightness) were assembled from VPs by the logical OR function.
The position of a dynamic cursor is controlled interactively, and technically, a new 3D image is generated and displayed with the same background 3D image for each cursor position. Only a small part of that image is redrawn. This is essential for a quick interaction.

### 3.1. Orthogonal layout of cells

In the 1D case, a vertical lenticular/barrier plate forms the cells of the width $W$. For convenience, we assume the height of the cell is equal to its width (i.e. the cell is a square $W \times W$). In the 2D case, the identical perpendicularly crossed vertical and horizontal barrier/lenticular plates form the square image cells $W \times W$.
Figure 9. Patterns for cross-shaped cursor: ±5th depth planes; HPO case: (a) −5th plane and (b) +5th plane. Reference points and silhouette are shown in Figure 8.

Figure 8 shows the cursor pattern for the 1st depth plane (in the image plane of ASD) of both HPO and FP cases. Formally, this shape can be obtained by using VPs from Figures 2(b) or 5(b) with the aligned reference points. For the other depth planes, this shape should be transferred to other distances by using the reference points and patterns. The reference points of the patterns should coincide with the reference points of all five cells defining the shape in Figure 8. The cursors for the HPO and FP cases generated this way are shown in Figures 9 and 10. For graphical clearance of these illustrations, the neighboring patterns are painted in different false colors; however, they actually have the same color. The reference point of each VP coincides with one of the reference points of the shape, which are also shown in Figures 2 and 5.

From these illustrations, individual VPs can be recognized and imagined how they form the template (i.e. cursor pattern).

3.2. Non-orthogonal layout (HPO)

Practically, the barrier/lenticular plate of an HPO display is twisted, not vertical, in order to reduce the moiré effect. Then, VPs should be accordingly modified. For instance, there are groups of identical rows, as shown in Figure 11. Each of the next group is shifted by 1 pixel horizontally. With the rational slant angle of arctan (1/q), the group contains q rows, the slant angle is 14.0° and, therefore, q = 4 (i.e. 4 rows in the group).

Based on these slanted VPs of 4 rows, we can build the cursor pattern for the same shape as in Figure 8, but with a different (slanted) layout, as shown in Figure 12.

Due to the simple shape of the cursor, the 4th row of cells is empty. Note the differences in Figure 9. Although

Figure 10. Patterns for cross-shaped cursor: ±5th depth planes; FP case: (a) −5th plane and (b) +5th plane. Reference points and silhouette are shown in Figure 8.

the reference points in Figure 12 are shifted, they are still within the contour of the shape.

4. Experiment

In the experiments, we used the laptop computer Lenovo Yoga 730 (13") with a 4K UHD display (3840 × 2160 pixels, pixel pitch 0.077 mm) with the parallax barrier plate installed at a slant angle of 18.4° = arctan (1/3). The width of the cell in this ASD is 10 pixels; thus, the maximal depth obtained using VP is at ±10th planes. The optimal viewing distance is 100 cm and the width of the viewing zone is 42 cm.

We assume that the smallest element of the visual 3D picture in an ASD is the image cell (one barrier/lenticular element). With a small enough pitch of cells (0.77 mm in this experiment), the visual image observed from the optimal distance appears smooth and continuous, without jumps, discreteness, etc.

In particular, for experiments with an interactive mouse-controlled 3D cursor in the display device, we
prepared skew-arrow-shaped cursor patterns up to ±10th depth planes based on the principles explained in Sections 2 and 3, as shown in Figure 13. This cursor was used with a previously generated 3D image of the alphabetic characters. Note the repeated near-vertical arrangements similar to Figure 1. However, Figures 1 and 11 were obtained in completely different ways. The arrangements are the result of the overlapped voxel patterns.

In the experiments, the binary black-and-white 3D image and the black-and-white cursor were used. None of them have intermediate levels of brightness. Therefore, not all integer depths in the interval \([-10, 10]\) were practically available, and only those \(k\) values, which are the divisors of \(W\), i.e. when \(w_k\) in Equation (1), is the integer number without the fractional part.

The size of the cursor patterns in Figure 13(a,c) is \(127 \times 54\) pixels; in Figure 13(b), it is \(87 \times 54\). To demonstrate the 3D cursor in action, we developed a computer program that displays 3D images with a controlled 3D

---

Figure 11. VPs for slanted barrier (HPO case) for ±5th and 1st depth planes: (a) –5th plane, (b) 1st plane, and (c) +5th plane. Note that the rows of pixels are only shown in the 1st cell of each row; however, they do exist in other cells. Reference points and silhouette are shown in Figure 8.

Figure 12. Cursor patterns for slanted HPO: (a) –5th depth plane and (b) +5th depth plane. Reference points and silhouette are shown in Figure 8.

Figure 13. Arrow-shaped HPO cursor patterns based on VPs: (a) –5th depth plane, (b) 1st plane, and (c) +5th depth plane. (For graphical simplicity, cell boundaries are not shown.)
cursor, a sort of 3D image browser (with the standard Windows cursor switched off completely).

The software program monitors the mouse movement and modifies the position and shape of the cursor accordingly. In response to the movement of the mouse, no complicated calculations are made. We combine the prepared patterns and display them by using the regular Windows API, namely, the GDI functions BitBlt() and AlphaBlend(). Rendering is not involved in this procedure; therefore, the delay time is essentially small.

A dynamic cursor requires restoring the previous content in its current place and re-drawing the cursor (probably modified) in another location. When the position of the mouse or its scroll wheel is changed beyond a predefined limit, the program erases the previous cursor, i.e. restores the background (the original 3D image) and draws the same cursor at another location or uses another VP (if the distance to the screen was noticeably changed, as determined by the wheel rotation). After that, the cursor immediately appears for the observer’s eyes at the desired location in 3D.

The flowchart of the second (run-time) part of the algorithm is shown in Figure 14 (the first part was shown in Figure 3). In the beginning, we open a file and draw the background 3D image. (Technically, the term ‘draw’ here means to copy something into the image plane of the ASD.) An operator can see a 3D image from the very beginning. In order to ease the visual tension, another 3D image can be shown as requested by the operator. However, this does not affect the behavior of the cursor. Then, we wait for something to happen with the mouse; it could be a move or scroll. In principle, the mouse click can also be taken into account (e.g. to show some additional information), but this does not affect the principal topic of this paper, the cursor.

First, we have to determine the run-time, if the move or scroll is large enough to make any change. We need the position change to over one cell or the depth change to over one plane, although the mouse can produce smaller increments. Otherwise, we will continue to wait. Then, if the depth has changed, another cursor pattern is taken from the array prepared at the first stage. For the changed position, the new reference point on the screen

---

**Figure 14.** Displaying cursor patterns at proper position and depth according to the mouse move.
is calculated. The background image is then restored at the previous position of the cursor, and the new cursor is drawn at a new position with the reference points aligned. Now, the operator can see the result of the move or scroll in 3D with his own eyes.

After the basic procedure, everything will depend on the 3D content, with options far beyond the scope of the current paper. In the simplest case, the operator can visually align the cursor to different parts of the 3D image by three coordinates and thus visually estimate the depth of this or that part.

The time required for the above actions (from the mouse move until the displayed cursor) should be less than the mean reaction time of humans because the operator would not feel a delay. Practically, the time required to redraw the cursor in the developed C++ program running a 3.4 GHz computer with 16 GB RAM is less than 35 msec (even with the largest cursor pattern at the maximal plane), as measured in the Visual Studio IDE. Such an elapsed time does not appear to be a noticeable delay for the real-time interaction because it is several times shorter than the average reaction time of humans (200–250 msec, as reported in [26]).

The experimental photographs of the cursor displayed at several depth planes are shown in Figure 15.

In Figure 15(a), the cursor is near the digit ‘4’, then it moves to the screen plane in Figure 15(b), and finally to the letter ‘B’ in Figure 15(c). The experimentally measured ‘depth’ of the character ‘4’ is about 0.5 cm. (The depth of character ‘B’ was not measured in the experiment because it lies behind the screen.) Note that these are characteristics of either the display or image, not the cursor.

Figure 15 is comprised of three rows (a) – (c), each containing two photographs taken by cameras with a relative displacement of 6.5 cm. Therefore, each row of this illustration can be watched as a stereo pair with the naked eyes directly from the journal page. However, certain practical skills are required. For example, in the so-called free viewing (parallel viewing method), the viewer looks ‘through’ the stereo pair with the parallel axes of the eyes. (Try to relax your eyes and look straight ahead, with each eye having parallel lines of sight.) When successful, you will see three images instead of two printed ones, with the central image being 3D. Thanks to Helmholtz, this direct viewing method has been known for almost two centuries, as mentioned in [27]. More explanations on free viewing can be found in [28,29] and elsewhere.

5. Discussion

We combine VP’s in computer memory, and then the cursor pattern is physically drawn in the image plane of the ASD over a background image. Therefore, the 3D cursor partially overwrites the previously displayed content, even if, according to the mouse scroll, the cursor should be behind a 3D object. In this case, the 3D cursor seems to ‘penetrate’ into an object and may look like an unexpected hole in the frontal surface of the 3D object. A similar visual effect is noted in [30]. Of course, nothing comparable can happen in the real world. By chance, such
an artifact will not destroy the stereoscopic perception but may be annoying for the observer.

There are many known types of autostereoscopic displays (e.g., multiview, integral imaging, plenoptic, and light-field) [1,5,31–33]. Particularly, it is believed that the multiview displays have a few discrete parallaxes, and the integral imaging supports the continuous parallax, and the plenoptic images allow refocusing. Nonetheless, the structure of the image in the image plane of these displays is similar [24], and in this paper, we are focused on the patterns in the image plane.

VPs were verified in the multiview displays [22,23] and can also be applied to the integral/plenoptic/light-field displays with the lenticular or barrier plates. As shown in [24], the same method of processing produces virtually the same result for all mentioned types of images, despite the fact that the images were taken from different independent sources.

In principle, our results can be probably applied to the near-eye displays [34] or other binocular displays with two views only (left and right). However, this suggestion needs an additional experimental investigation. Moreover, instead of regular barrier slits (as shown in this article), a random or irregular pinhole array can be used [35]. In that case, the operator might have a better security because there is only one viewing zone in the random case instead of multiple zones of the regular case. However, the voxel patterns were not observed with the random pinholes.

Furthermore, the openings of the barrier are not infinitesimal points, but pinholes of finite size. The pixels of the screen have a fixed non-zero size as well. These facts mean that instead of the mathematical rays intersecting at a point, wide rays should be considered, which will intersect within a region. However, regions instead of points are not unnatural for the voxels. This reminds us that VP is an approximate representation of the exact rays, where the voxels represent the points approximately (see [22,23], and Section 1). Thus, there would be certain (but still small) deviations within this region for various observers. Imagine that a couple of rays in Figure 7(c) have non-zero width. Then, the intersection of two widened rays would be a rhomboidal area rather than a point. Rough analogs of that could be the Gaussian beams in optics and the umbra/penumbra in astronomy.

In our experiments, VPs were prepared prior to the start of the program. The cursor shape is not modified within the same depth plane; only a lateral displacement by an integer number of cells may be required. The shape needs a modification only for another depth plane. On the other hand, VP’s clear geometric structure allows for the dynamic construction of cursors of any shape and depth in real-time.

In the case of 10 views and the abovementioned parameters of the experimental display device, the maximum distance calculated by Equation (3) (to the 10th plane) is 1.6 cm, while the depth observed in the experiments is 1.7 cm. The neighboring (9th) depth plane is approximately 1.5 mm closer to the screen, and it is probably difficult to distinguish these planes visually from 1 m.

However, this should not be difficult for the cursor driving routine. Remember that with a 4K monitor, the computer mouse potentially can move by one pixel at a time. However, can we clearly distinguish the neighboring pixels at a distance of 1 m? Note also that the simplified formula Equation (3) is biased by 1 depth plane, but such a small mismatch (less than 2 mm in the previous example) does not have a practical meaning for vision. Nevertheless, the depth is the characteristic of the display device, not the characteristic of the cursor.

The shape and size of the image cells of the image must correspond to that of the cursor. VPs designed for one device may not fit another. Thus, in a device with another layout of the image cells, both the image and the cursors should be remade. We specifically show the VPs for two layouts, orthogonal and slanted. In Figures 9 and 12, where the shape is the same, the positions of the pixels and reference points are different. However, only a few parameters are enough to bring everything in order (number of pixels in the image cell; its shape [i.e. width and height]; and the slant angle of the barrier/lenticular plate). This indicates that the proposed 3D cursor is hardware dependent, but not overly so.

A practical requirement of the current implementation is the integer image cells comprised of an integer number of pixels. Other conditions could include the integer number of cells across the MV image, with at least one complete cell in the LU corner. Both are not fundamental restrictions.

The current findings, along with [36], confirm that the processing of integral/multi-view images is better based on the image cells.

The model of VPs implies that physical/virtual cameras with parallel axes lie in a planar rectangular matrix with identical spacing and the plane of the matrix is perpendicular to the z-axis. Such cameras produce so-called orthographic rectified views. There are no VPs in non-rectified images.

Controlling the depth by using the scroll wheel seems to be convenient for users. Once the mouse was touched, an operator does not need further explanation regarding movement in a visual 3D space using the scroll wheel.

The generation or modification of the 3D images themselves is beyond the scope of the current paper. We did not analyse the color, depth, or geometry of the 3D
images in this paper. Depth analysis is another research topic (refer to [37]). In this research, we prepared the basic features; however, the practical interaction with the 3D content falls outside the scope of the paper because it requires some knowledge of the particular content (e.g., locations of objects, their color, texture, etc.). In other words, access to the content database becomes necessary for the interaction. As a 2D example, one deal is to find that the current position of the mouse pointer is, for example, (598, 215) or (319, 613), and a completely different deal is to recognize that the first location points to the word ‘it’ and the second to ‘the’, which was mistakenly typed as ‘hte’, thus requiring a correction. In this paper, we are intentionally limited by the first step (displaying the cursor in 3D according to the mouse movement/rotation) and do not consider any 3D database at all.

The white cursor described in this paper is clearly visible on the dark background used in the current experiments. Generally speaking, the cursor can be added to any grey-scale or color 3D image. In that case, a colored or textured background can reduce the visibility of the cursor. Compared to the 2D case, the cursor is better separated from the background because depth plays an important role even with slightly different colors. To make the white cursor definitely visible on an arbitrary background, the white cursor should be surrounded by black voxels.

6. Conclusion

We proposed a 3D cursor and verified the results experimentally in a barrier 3D display. The cursor was displayed with previously made multiview 3D images. It was controlled in real-time by a regular computer mouse. An advantage of the proposed approach is that it does not require complicated geometric calculations in 3D; the program works without the rendering of multiple views. This makes our approach different from others.

We do not need a complicated 3D pointing device. A regular computer mouse with a scroll wheel can effectively control the position in 3D. The proposed method could be the first step towards a simple and natural 3D interaction.

Based on this paper, the reader can learn how to draw a 3D cursor simply by assembling the pre-defined VPs in the image plane of the ASD. Visually perceived distances to the voxels were estimated.

The dynamic 3D cursor based on the voxel patterns was verified in ASD display with the parallax barrier plate. Potential applications include designing, editing, arranging, and transforming the 3D images interactively in the multiview, integral, and plenoptic images.

Acknowledgments

The authors thank the Priority Research Centers Program for the support of this work through the National Research Foundation of Korea (NRF) funded by the Ministry of Education.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by National Research Foundation of Korea [grant number NRF-2018R1A6A1A03025542].

Notes on contributors

Vladimir Saveljev is a Researcher at Konyang University (Nonsan, Korea) with over 40 years of scientific research experience. He holds a PhD in electrical and computer engineering from the Siberian Branch of the Russian Academy of Sciences. He started his carrier at the Institute of Automation and Electrometry, Novosibirsk, Russia. Afterward, he continued his research at Korean Institute of Science and Technology (Seoul, Korea), Hanyang University, Dongguk University (both Seoul, Korea), and Myongji University (Gyeonggi, Korea) as a Research Scientist/Consultant, a Lecturer, and a Research Professor. He has authored and co-authored over 150 academic papers. These include 4 book chapters, 95 articles on 3D displays, and 68 studies of the moiré effect. To illustrate, the publications by Dr. Saveljev are highly cited. For instance, several of his journal articles received over 100 citations. The current research interests of Dr Saveljev include but not limited to the applied optics (autostereoscopic 3D displays, image quality), the physics (moiré effect, moiré minimization, nanoparticles), and the mathematics (wavelet transform).

Jung-Young Son (Member, IEEE) received the B.Eng. degree in avionics from the Korea National Aviation University, South Korea, in 1973, and the M.S. degree in electronics and the Ph.D. degree in engineering science from the University of Tennessee, Knoxville, TN, USA, in 1982 and 1985, respectively. From 1980 to 1985, he was a Graduate Research Assistant, and from 1985 to 1989, a Research Scientist at the University of Tennessee Space Institute. From 1989 to 2002, he worked at the Korea Institute of Science and Technology as a Principal Research Scientist in Optics. He is currently a Professor at the Biomedical Engineering Department, Konyang University, Nonsan, South Korea. His primary interests are focused on 3-D image displays, electroholography and electromagnetic wave images for nondestructive testing. Dr Son is Doctor of Technical Science, and Academician of Academy of Technological Sciences of Ukraine, and a Fellow of the International Society for Optics and Photonics (SPIE). He is also serving as an Associate Editor of Optical Engineering. He has more than 150 Journal articles written in Korean and English and 300 international and Korean conference articles.
Choon-Sik Yim received his BS and MS degrees in Electronics and computer engineering from Korea Aerospace University, Seoul, Rep. of Korea, in 1975 and 1986. His PhD degree in Electrical Engineering and Computer Science from Yokohama National University, Japan, in 1993. Since 1978, he has been with the Electronics and Telecommunications Research Institute(ETRI), Daejeon, Rep. of Korea, where he is currently a principal research member. From 2014, he worked as a wireless communication and computer engineer at RCN Co. Recently, he has been engaged in the development of Low Power Wide Area communications(LPWA) for their system applications. His main research interests are long-range IoT system based on LPWA.

Gwanghee Heo received a Ph.D. from the Department of Civil Engineering at the University of New Mexico in 1997. He is currently Professor of Civil Engineering at Konyang University and also Director of the Public Safety Institute of the Konyang University. His research interests lie in Structural Health Monitoring System, Smart Sensing Technology, Smart Structural Dynamics and Control, Vibration Control and Damping System and so on.

ORCID

Vladimir Saveljev © http://orcid.org/0000-0003-2187-704X

References

[1] B. Lee, S.-G. Park, K. Hong, and J. Hong, Design and Implementation of Autostereoscopic Displays (SPIE Press, Bellingham, WA, 2016).
[2] N.A. Dodgson, Autostereoscopic 3D Displays, IEEE Comput. 38 (8), 31–36 (2005).
[3] M. Liebling, T. Blu, and M. Unser, Fresnels – A New Wavelet Basis for Digital Holography, Proc. SPIE 4478, 347–352 (2001).
[4] N.S. Holliman, N.A. Dodgson, G.E. Favalora, and L. Pockett, Three-Dimensional Displays: A Review and Applications Analysis, IEEE Trans. Broadcast 57 (2), 362–371 (2011).
[5] J. Geng, Three-Dimensional Display Technologies, Adv Opt Photonics 5 (4), 456–535 (2013).
[6] W. Mphopo, Stereoscopy and Autostereography, ch. 9 in Mixed Reality and Three-Dimensional Computer Graphics, edited by B. Sobota and D. Cvetković (IntechOpen, 2020), pp. 1–13. DOI:10.5772/intechopen.92633.
[7] H. Liao, S. Nakajima, M. Iwahara, N. Hata, I. Sakuma, and T. Dohi, Real-Time 3D Image-Guided Navigation System Based on Integral Videography, Proc. SPIE 4615, 36–44 (2002).
[8] G. Park, J.-H. Jung, K. Hong, Y. Kim, Y.-H. Kim, S.-W. Min, and B. Lee, Multi-Viewer Tracking Integral Imaging System and Its Viewing Zone Analysis, Opt. Express 17 (20), 17895–17908 (2009).
[9] M.C. Forman, N. Davies, and M. McCormick, Continuous Parallax in Discrete Pixelated Integral Three-Dimensional Displays, J. Opt. Soc. Am. A 20 (3), 411–420 (2003).
[10] D.B. Douglas, R.E. Douglas, C. Wilke, D. Gibson, J. Boone, and M. Wintermark, A Systematic Review of 3D Cursor in the Medical Literature, AIMS Electron. Electr. Eng 2 (1), 1–11 (2018).
[11] N. Osawa, Evaluation of 3D Pointers and Annotations on Autostereoscopic Displays for 3D Video Communications, Proc. IEEE Int. Conf. Syst. Man Cybern., 954–959 (2007). DOI:10.1109/ICSMC.2007.4413931.
[12] B.G. Blundell, 3D Displays and Spatial Interaction (Walker & Wood, Auckland, New Zealand, 2011).
[13] Z.Y. Alpaslan, and A.A. Sawchuk, Three-Dimensional Interaction with Autostereoscopic Displays, Proc. SPIE 5291, 227–236 (2004).
[14] H. Azari, I. Cheng, and A. Basu, Stereo 3D Mouse Cursor: A Method for Interaction with 3D Objects in a Stereoscopic Virtual 3D Space, Int. J. Digit. Multimed. Broadcast, 2010, article ID 419493 (2010).
[15] J. Unterguggenberger, B. Kerbl, M. Steinberger, D. Schmalstieg, and M. Wimmer, Fast Multi-View Rendering for Real-Time Applications, Proc. Eurogr. Symp. Parallel Graph. Vis., 13–23 (2020). DOI:10.2312/pvg.20s2017.
[16] T. Hübner, Y. Zhang, and R. Pajarola, Single-Pass Multi-View Rendering, IADIS Int. J. Comput. Sci. Inf. Syst. 2 (2), 122–140 (2007).
[17] T.N. Bardsley, and I. Sexton, Task Performance Using 3D Displays, in IFIP Advances in Information and Communication Technology - Virtual Prototyping, edited by J. Rix, S. Haas, and J. Teixeira (Springer, Boston, MA, 1995), pp. 228–246.
[18] J. Zhang, X.-Q. Xu, J. Liu, L. Li, and Q.-H. Wang, Three-Dimensional Interaction and Autostereoscopic Display System Using Gestture Recognition, J. SID 21 (5), 203–208 (2013).
[19] J. Tompkin, S. Muff, J. McCann, H. Pfister, J. Kautz, M. Alexa, and W. Matusik, Joint 3D Pen Input for Light Field Displays, Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology, 637–647 (2015).
[20] V. Saveljev and I. Palchikova, Analysis of Autostereoscopic Three-Dimensional Images Using Multiview Wavelets, Applied Optics 55 (23), 6275–6284 (2016).
[21] V. Saveljev and S.-K. Kim, Reference Functions for Synthesis and Analysis of Multiview and Integral Image, J. Opt. Soc. Korea 17 (2), 148–161 (2013).
[22] J.-Y. Son, B. Javidi, and V. Saveljev, Synthesizing 3D Images Based on Voxels, Proc. SPIE 5202, 1–11 (2003).
[23] J.-Y. Son, V. Saveljev, S.-K. Kim, and B. Javidi, Pixel Patterns for Voxels in Contact-Type 3D Imaging Systems, Jpn. J. Appl. Phys. 45 (2A), 798–803 (2006).
[24] V. Saveljev, Analysis of Depth in Integer, Multiview and Plenoptic Images by Using Multiview Wavelets, ch. 11 in Emerging Trends in Engineering Research and Technology, edited by A.K. Goel (Book Publisher International, London, 2020). Vol. 1, pp. 141–154.
[25] N.I. Petrov, M.N. Khromov, and Y.M. Sokolov, Large-Screen Multi-View 3D Display, OSA Cont. 2 (9), 2601–2613 (2019).
[26] A. Jain, R. Bansal, A. Kumar, and K.D. Singh, A Comparative Study of Visual and Auditory Reaction Times on the Basis of Gender and Physical Activity Levels of Medical First Year Students, Int. J. Appl. Basic Med. Res. 5 (2), 124–127 (2015).
[27] A. Parkin, Digital Imaging Primer, 218 (Springer, London, UK, 2016).
[28] Wikipedia, Stereoscopy: Freeviewing, 2021 https://en.wikipedia.org/wiki/Stereoscopy.
[29] Triaxes, 3D Theory: Parallel/Cross-Eyed Viewing Method, 2021. https://triaxes.com/docs/3DTheory-en/522ParallelCrosseyedviewingmetho.html.
[30] F. Steinicke, G. Bruder, K. Hinrichs, and T. Ropinski, 3D User Interfaces for Collaborative Work, Ch. 17 in Human Computer Interaction, edited by I. Pavlidis (IntechOpen, 2008), pp. 279–294. DOI:10.5772/6297.
[31] E. Lueder, 3D Displays (Wiley, 2012). ISBN:978-1-119-96304-2.
[32] R. Ng, M. Levoy, M. Bredif, G. Duval, M. Horowitz, P. Hanrahan, and D. Design, Light Field Photography with a Hand-Held Plenoptic Camera, Stanford Tech Report CTSR 2005-02, 2005.
[33] M. Levoy and P. Hanrahan, Light Field Rendering, Proc. ACM SIGGRAPH, 31–42, 1996.
[34] G.A. Koulieris, K. Ak, M. Stengel, R.K. Mantiuk, K. Mania, and C. Richardt, Near-Eye Display and Tracking Technologies for Virtual and Augmented Reality, Comput. Graph. Forum 38 (2), 493–519 (2019).
[35] W. Song, Q. Cheng, P. Surman, Y. Liu, Y. Zheng, Z. Lin, and Y. Wang, Design of a Light-Field Near-Eye Display Using Random Pinholes, Opt. Express 27 (17), 23763–23774 (2019).
[36] V. Saveljev, and S.-J. Shin, Layouts and Cells in Integral Photography and Point Light Source Model, J. Opt. Soc. Korea 13 (1), 131–138 (2009).
[37] D.-C. Hwang, K.-J. Lee, S.-C. Kim, and E.-S. Kim, Extraction of Location Coordinates of 3-D Objects from Computationally Reconstructed Integral Images Basing on a Blur Metric, Opt. Express 16 (6), 3623–3635 (2008).