A Method of Computational Correction for Optical Distortion in Head-Mounted Displays

Jannick P. Rolland and Terry Hopkins

Dept. of Computer Science, University of North Carolina,
Chapel Hill, NC 27599-3175, U.S.A.

Abstract Optical distortion is one of the optical aberrations of optical systems that do not affect image sharpness. Thus, it can be beneficial to correct optical distortion computationally rather than optically. Computational correction versus optical correction will generally yields lighter optical systems. This is of great importance for head-mounted displays where the weight of the viewer needs to be minimized for efficient performance, safety, and optimum duration of usage. We shall describe in this paper, one method of computational correction that consists in warping polygon vertices. It is the case, however, that the algorithms given can be easily rewritten to handle pixels rather than vertices warpings.

1. Introduction

A head-mounted display (HMD), head tracker, computer graphics system, and appropriate display software may be used to give a user the perception of a computer-generated space which is spatially stable. Different names, such as virtual reality, cyberspace, artificial reality, virtual environments, and synthetic experience, have been used to describe the perception produced by this apparatus. In order to generate a pair of stereoscopic images for the two display devices in the HMD, the display software must take into account the relative positions of the head tracker, the display devices, the optics, and the user's eyes (Robinett and Rolland, 1992).

To simulate objects that are spatially stable and shape invariant regardless of their position in the world, both spatial and temporal problems must be solved. This paper focuses on the optical imaging or static image generation problem. Temporal problems will be treated elsewhere (Adelstein et al. 1992 and work in progress at the University of North Carolina at Chapel Hill (UNC-CH)).

The purpose of the optics used in an HMD is to project equally magnified images to the eyes of the user in such a way that they fill or partially fill the user's field of view. Moreover, the optics also (and in some cases, only) serves as an auxiliary to the lens of the eye so that the latter remains at rest (i.e., at infinity focus). In any case, the focal length of the optics as well as the distance of the miniature displays to the optics must be chosen such that the so-formed virtual images fall within the range of accommodation (focus) of the eyes. When the eyes accommodate on the virtual images, the virtual environment can be seen in focus and its image sharpness is only limited by the resolution of the displays. There is evidence in the literature, however, that what sets accommodation in HMDs is not so simple and that the predictions which come from optical image formation can be violated. A review of the problem of accommodation in HMDs can be found in (Rolland et al., 1993).

Any image-forming optical system will cause some degradation to the quality of the images being formed. There are several types of optical image degradation which are referred to as *optical aberrations*. The term *aberration* emphasizes the different path that light rays follow in a real optical system such as a lens in comparison with the paths of rays in an ideal optical system. These idealized optical systems are described by a simple mathematical model called *paraxial analysis*. Most often, the lenses and other optical elements in an optical system are lined up and centered along a line called the *optical axis*. The term *paraxial* refers to rays that are close to the optical axis of the optical system being analyzed. Under the paraxial laws of image formation, however, rays are assumed to propagate not only near the optical axis but also at shallow angles with respect to the different optical surfaces. For optical systems in which more oblique rays occur, the behavior of the optical system deviates further from the paraxial model and optical aberrations become more pronounced. This is often the case of HMDs due to the requirement of having not only a wide field of view (FOV) to create an immersive virtual environment, but also a large exit pupil size to allow the eyes of the user to swivel in their sockets without causing *vignetting*, that is partial optical rays occlusion, in the FOV. The simultaneous requirement of a large FOV and a large exit pupil size causes oblique rays to be present and it becomes more difficult to minimize or balance out low- and high-order optical aberrations.

Optical aberrations may be described in terms of the amount by which a geometrically traced ray misses a specified location in the image plane formed by the optical system. The displacement of the ray is referred as the *transverse ray aberration*. Most often, the specified location for the ray in the image plane is that inferred from the first-order or paraxial laws of image formation (Longhurst, 1973). Some first-order properties that will be relevant later on in this paper are the focal length and the FOV of the optical system.

In the process of designing an optical system, the designer will often need to assemble several optical components (lenses, prisms, mirrors) to minimize each individual aberration and balance the residual aberrations out. If the final system must be corrected for all main aberrations, the weight of the system may become too great if the system is to be head-mounted. One approach toward resolving this problem is to look more closely at the types of optical aberrations. We can separate aberrations that decrease the image quality, as measured in terms of how sharp the image is, from those that change the shape of the objects being imaged. By designing an optical system where the only optimization constraints are on aberrations that would cause a loss in image sharpness, lighter systems can be designed and system performance goals can still be reached. To compensate for the generally unwanted warping of the optically formed images, an additional step can be introduced into the calculation of the computer-simulated images themselves. The images thus generated are pre-warped in such a way as to cancel out the optical warping generated through imaging. Thus, the images seen by the eyes are perceived sharp and unwarped after imaging.

There are two kinds of optical warpings that do not affect image sharpness: *lateral chromatism* and *distortion*. Lateral chromatism in optical systems is the change in image size, that is in *transverse magnification*, with the wavelength of light. Although it not the subject of this paper, it could be computationally corrected for, if desired. The authors are not aware at this time, however, of any optical system designed with uncompensated lateral chromatism that was then corrected for computationally (Weisman,1992). Optical distortion is the change in transverse magnification as a function of the off-axis distance of an object point in the FOV and is the subject of this paper. We shall note that because distortion causes non-uniform magnification across the FOV, it causes not only a warping of the image but also a variation in brightness across the FOV. This can also be compensated for on a pixel-by-pixel basis with a brightness-correction function.

To summarize, only optical aberrations that do not affect image quality, namely distortion, lateral chromatism, and brightness variation, can be compensated for computationally. Other optical aberrations which degrade image sharpness, such as spherical aberration, coma, astigmatism, and field curvature are not susceptible to computational correction. In this paper, we describe the

implementation of computational correction for optical distortion. We first review the mathematical description of optical distortion. We then describe one method of computational implementation for use in real-time computer graphics HMD systems.

2. Mathematical Model of Optical Distortion

An image-forming optical system forms an image (real or virtual) of an object. For the optics in an HMD, the display screen is the object, and the images formed by the optics are always virtual. This is shown in Figure 1 for one eye.

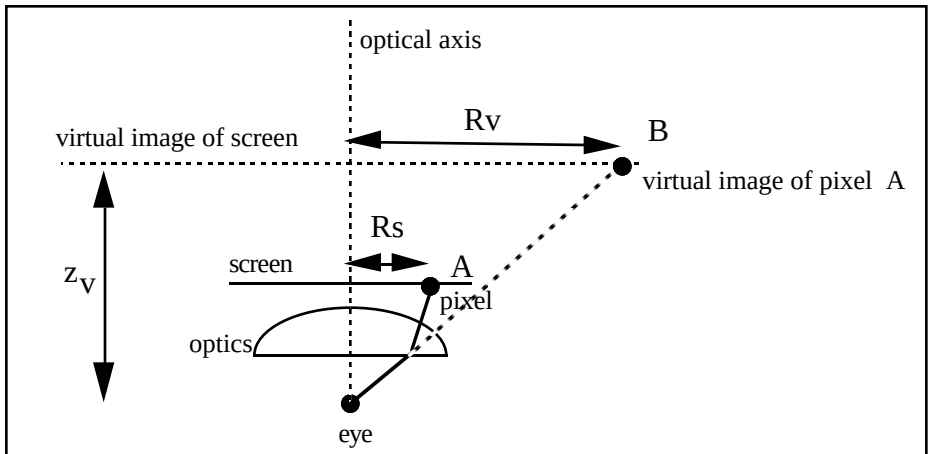

Figure 1. Model of the optical system for one eye in an HMD.

If we assume that the optical system is symmetrical around the optical axis, an ideal magnifier is linear as described by the paraxial laws of image formation. Thus, it can be described by a simple linear Equation

$$R = m\ R_S \quad , \tag{1}$$

where m is the transverse magnification, R_S is the radial distance of a point of light in the object plane, and R is the radial distance of the intersection of a paraxial ray emanating from a point of light located at R_S with the image plane. The location of the image plane is a function of the focal length of the optics and the location of the object plane with respect to the principle planes of the optics (Hecht and Zajac, 1974). The optical axis serves as the origin from which R_S and R are measured.

For wide FOV optical systems, distortion may occur and the mapping of a point of light in the object plane to a point of light (virtual in our case) in the image plane may be given more generally by

$$R_V = D(R_S) \quad , \tag{2}$$

where R_V is the radial distance measured from the optical axis of the intersection of a real ray emanating from a point of light located at R_S with the image plane, and D( ) is a non-linear mapping function. In order to express $D(R_S)$, we need to precisely define what is meant by distortion. Distortion is defined as follows: if we define the chief *rays* of the optical system to be the rays starting at any point on the screen (object space) and passing through the center of the

entrance pupil[1], distortion ΔR is the amount of displacement of the chief rays as they intercept the image plane from their position as given by paraxial or first-order optics:

$$R_V = R + \Delta R \quad . \tag{3}$$

If image aberrations are expressed in terms of ray aberrations, distortion ΔR can be written as a sum of polynomial expressions of the form

$$\Delta R = k R^3 + h R^5 + \text{higher order terms} \tag{4}$$

where k and h are the coefficients for the third- and fifth-order terms of the distortion, respectively (Smith, 1966). In the remainder of this paper, we shall be discussing distortion correction up to the third-order approximation, since it is most often sufficient as a correction strategy. Up to the third-order term, the intercept of a ray with the image plane is then given by

$$R_v = D(R_S) = m R_S + k (m R_S)^3 \quad . \tag{5}$$

If the lens is designed free of third-order optical distortion, k equal zero, R_V equals R, and the imaging equation reduces to Equation 1. A lens is often characterized by its percent distortion which is defined as the ratio of the ray displacement to the paraxial ray height value. Thus, it is given by the ratio

$$\% \text{ distortion} = 100 \frac{k (m R_S)^3}{m R_S} \tag{6}$$

$$= 100 k (m R_S)^2 \quad .$$

The transverse magnification of the optics is usually known or can be derived from first-order optical properties. The percent distortion can be obtained by optical raytracing for any point on the screen, R_S, given the optical specification of the lens. The corresponding value of k can then be calculated as

$$k = \frac{\%\text{distortion}}{100 (m R_S)^2} \quad . \tag{7}$$

Our goal is to modify the computer-generated image so that it appears free of distortion when viewed through the optics of the HMD. We shall refer to this as *predistortion*. To *predistort* an image, we need to move all the object points on the screen in an opposite manner to the distortion mapping so that, after imaging, the image points appear to be in the ideal positions as described by paraxial analysis.

If we describe the position of a non-distorted image point in virtual space as R, then its corresponding (pre distorted) point on the screen that we shall refer to as R_i will be given by

---

[1] Entrance pupil: in a lens or other optical systems, the image of the aperture stop[2] as seen from the object space.

[2] Aperture stop: A physical constraint, often a lens retainer, that limits the diameter of the axial light bundle allowed to pass through a lens.

$$R_i = D^{-1}(R) , \quad (8)$$

where D^{-1} is the inverse of the mapping function D defined by Equation 2. This will allow the point intended to be displayed at R by the graphics calculation to appear at

$$R_v = D(R_i) = D(D^{-1}(R)) = R . \quad (9)$$

In other words, the point will be seen by the eye in exactly the intended location, with the effects of the optical distortion compensated for exactly.

3. Computational Implementation of Predistortion

3.1 General Considerations

For real-time computer graphics, one way of implementing predistortion of the computer generated images, is to use a two-dimensional lookup table, with an input point (X_s,Y_s) in screen coordinates providing the two indices for the table, and a position (X_i,Y_i) in screen coordinates to which the point (X_s,Y_s) is to be moved.

In such an implementation, either polygon vertices or pixels could be remapped by the lookup table. A complete description of possible predistortion implementations and a comparison of implementation costs is under preparation (Lastra et al., in progress). We shall focus here on vertices predistortion. This approach has some problems because the remapping is non-linear but the edges and interior of the polygons are filled in the linear fashion that is standard in computer graphics. Vertices would be in the proper undistorted locations, but polygon edges might appear to be noticeably curved, particularly if the edges cross a large fraction of the screen. To deal with this problem, polygons that are large in screen space must be subdivided.

The computer simulated image remapping can be implemented as an optional step in the PPHIGS graphics library for the Pixel-Planes 5 graphics computer (Fuchs et al, 1989), which is the graphics engine used by the UNC HMD system. To put this predistortion mapping in the context of the overall HMD display calculation, the HMD coordinate system diagram is shown in Figure 2. This diagram and its associated nomenclature is discussed in more detail in (Robinett and Holloway, 1992, 1993).

The display calculation takes the vertices of polygons defined in objects space coordinates Po, as specified at the top of the tree diagram shown in Figure 2, and transforms these vertices to screen coordinates Ps, after which the renderer fills in the interior of the polygons. The object space mentioned here is three-dimensional in nature and is named as such in computer graphics nomenclature. It is not to be confused with the two-dimensional optical object defined in section 2 of this paper. The polygons mapping can be written as

$$Ps = Ts\_o \bullet Po , \quad (10)$$

where Ts_o is a mapping from 3D object space (o) to screen (s) space, the operation • being a functional composition. This notation is similar to that used by Foley et al. (1990). The Ts_o mapping may be decomposed into the component mappings shown in Figure 2 as

$$Ts\_o = Ts\_v \bullet Tv\_e \bullet Te\_h \bullet Th\_t \bullet Tt\_r \bullet Tr\_w \bullet Tw\_o , \quad (11)$$

where this mapping takes a vertex from object (o) coordinates successively through world (w), room (r), tracker (t), head (h), eye (e), virtual image of screen (v), and screen (s) coordinates.

Except for the frontmost two mappings, Ts_v and Tv_e, all of these operations are linear transformations between 3D coordinate systems, and may be constructed from translation, rotation and scaling transformations. Tv_e is the viewing transformation which includes perspective and shear, and Ts_v is the mapping from the virtual image of the screen to the screen itself which was earlier referred to as D^{-1}. Since D^{-1} (or Ts_v) is a non-linear transformation, it is not represented as a matrix as linear transforms usually are. This transformation must be done as the last step to get from virtual image (v) to screen (s) coordinates after all the other coordinate transformations have been carried out.

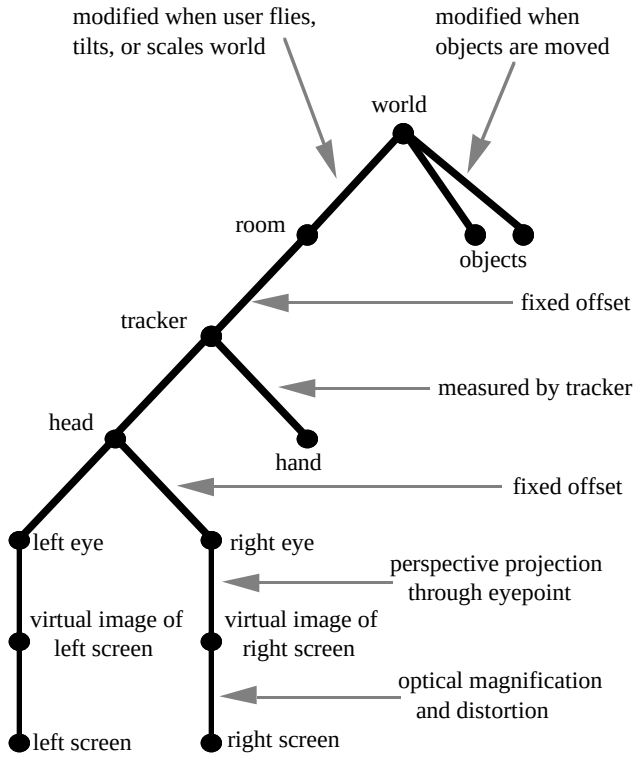

Figure 2. Coordinate system diagram for HMD

3.2 Calculation of the predistortion lookup table

Two initial steps are required to fill in the table used in the predistortion mapping operation: finding the R_i values that satisfy Equation 8 and filling in the lookup table values during initialization of the graphics system. After the table is filled in, the remapping of the vertices of the polygons representing the computer simulated image is done at run-time.

Several methods can be used to calculate the R_i values. If we consider Equation 5 for R_s equal R_i and R_v equal R, D is a polynomial in R_i, and a first method consists of assigning specific values to R and solving Equation 5 for R_i. A symbolic mathematical package such as Mathematica can be used to perform this extraction. The values computed can then be stored in a file and be loaded into the table for D^{-1} when needed. Another method is to solve the cubic polynomial D in closed form

for R_i, when R_v equal R (Beyer, 1984). This closed form solution can then be used to compute the lookup table values. This is the method we have adopted and that we shall now describe.

The roots of a cubic polynomial of the form

$$Y^3 + pY + q = 0 \quad , \tag{12}$$

are given by

$$Y_1 = A + B \quad , \tag{13}$$

where

$$A = \sqrt[3]{-\frac{q}{2} + \sqrt{Q}} \qquad \text{and} \qquad B = \sqrt[3]{-\frac{q}{2} - \sqrt{Q}} \tag{14}$$

$$\text{with} \qquad Q = \left(\frac{p}{3}\right)^3 + \left(\frac{q}{2}\right)^3 \quad . \tag{15}$$

If we rewrite Equation 5 (with Ri and R) to take the form of Equation 12, we get

$$R_i^3 + \left(\frac{1}{m^2k}\right) Ri - \left(\frac{R}{m^3k}\right) = 0 \quad , \tag{16}$$

where

$$p = \left(\frac{1}{m^2k}\right) \qquad \text{and} \qquad q = -\left(\frac{R}{m^3k}\right) \quad . \tag{17}$$

Substituting Equation 17 into the solution of the cubic inverse given by Equations 13 and 14, we obtain for R_i,

$$R_i = D^{-1}(R) = \sqrt[3]{-\frac{q}{2} + \sqrt{\left(\frac{p}{3}\right)^3 + \left(\frac{q}{2}\right)^3}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\left(\frac{p}{3}\right)^3 + \left(\frac{q}{2}\right)^3}} \quad , \tag{18}$$

where p and q are defined by Equation 17. The vertex locations R_i, given by Equation 18, are now the actual locations where screen vertex locations R_s should be moved to on the screen for the final optical image to appear undistorted to each eye. The steps to find the predistorted coordinates for a vertex are given as algorithm 1.

Algorithm 1

Given the coordinates X_s, Y_s of a vertex in screen coordinates, calculate

1. $R_s = \sqrt{X_s^2 + Y_s^2}$
2. $R = m R_s$

3. $R_i = D^{-1}(R)$ as given by Equation 18

4. $X_i = X_s \left(\frac{R_i}{R_s}\right)$ and $Y_i = Y_s \left(\frac{R_i}{R_s}\right)$.

The predistortion is implemented in a HMD system by using a PPHIGS library function, pg_distortion(). The syntax of the call is given by

pg_distortion(xsize, ysize, xorig, yorig, xsc, ysc, stereo, table)

where

xsize, ysize:	row and column size of the lookup table
xorig, yorig:	coordinates of the upper left corner of the lookup table in screen space
xsc, ysc:	lookup table indexes scaling factors
stereo:	flag specifying if predistortion is done for one or both eyes
table:	lookup table (table[xsize][ysize][2])

Theoretically, the function pg_distortion() takes a polygon vertex and moves it to a new location defined by *table*. In practice, however, one must account for the *video overscan*, often referred to as *image cropping* on the LCD displays and illustrated Figure 3. The term *overscan* refers to the fact that while a video signal may be generated from a 640 x 512 pixel frame buffer, for example, only a subset of those pixels will be displayed on the screens. Moreover, the cropping is usually asymmetric which implies that the center pixel of the screen does not corresponds to the center pixel of the frame buffer. Since distortion is symmetrical around the center of the imaging optics, not only should any offsets of the screens with respect to the optics must be taken into account (see Algorithm 2) but also the video overscan accounted for. We shall refer to the coordinates of the intercept of the optical axis point with the screens as A_x and A_y for the x and y coordinates,

respectively. The expressions for A_x and A_y given below for each eye reflect both the offsets of the

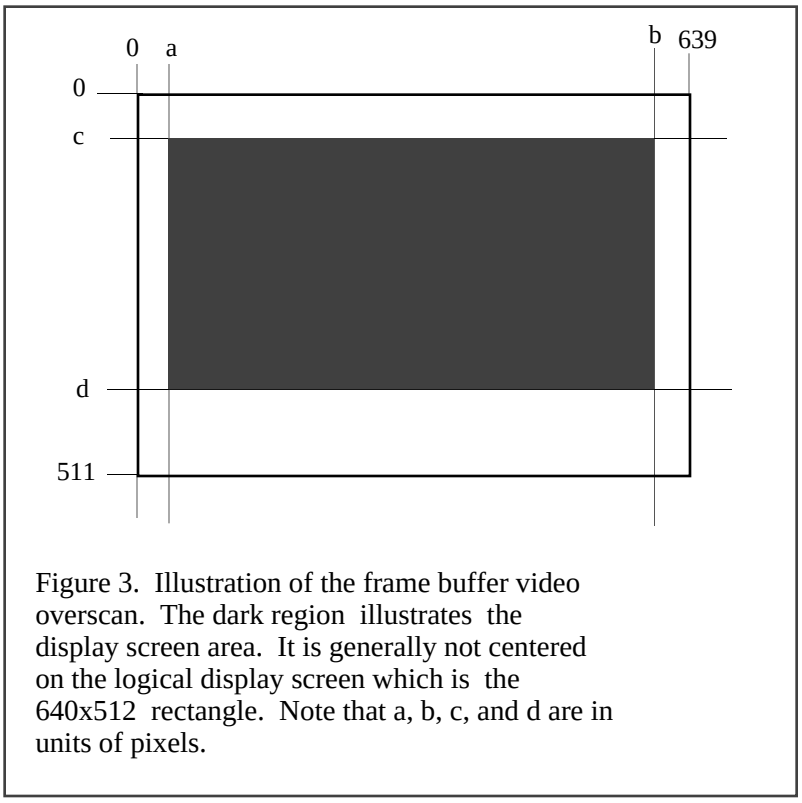

Figure 3. Illustration of the frame buffer video overscan. The dark region illustrates the display screen area. It is generally not centered on the logical display screen which is the 640x512 rectangle. Note that a, b, c, and d are in units of pixels.

displays with respect to the optics and the frame buffer video overscan.

Table is a three-dimensional array composed of two indexes to the table and a third parameter selecting between the new x-coordinate and y-coordinates outputs. The two indexes are the x and y coordinates of a given polygon vertex in screen space before it is moved to its new location specified by the predistortion operation. As an example, suppose that we want to know the predistorted coordinates of a vertex whose coordinates are (Xs, Ys). The table is accessed as follows:

Vertex coordinates	(Xs, Ys)
Predistorted vertex coordinates	$(X_i, Y_i) = (table[X_s][Y_s][x], table[X_s][Y_s][y])$

The predistortion table can be generated in advance for a particular HMD and stored in a file if it proves to be necessary. To account for a lookup table size that is smaller than the resolution of the screens, the scaling factors "xsc" and "ysc" must be used: multiplying the x-coordinate of a pixel in logical screen space[3] by "xsc" will give the x-index into table, and similarly for "ysc". If we refer to the logical screen as a two-dimensional array of size x_screen by y_screen, xsc and ysc are given by[4]

---

[3] By logical screen space here, we refer to the frame buffer resolution capability.

[4] Equation 19 assumes that pixels are numbered starting with zero. As an example, for a typical 640x512 logical display size, x_screen-1 and y_screen-1 will take the values 639 and 511, respectively.

$$xsc = \frac{xsize - 1}{(x\_screen-1)} \qquad ysc = \frac{ysize - 1}{(y\_screen-1)} \quad , \tag{19}$$

The parameter "stereo" is used to predistort the computer-simulated images for each eye separately. The optical distortion is the same for each lens, but the LCD screens are often decentered with respect to the optical axis in opposite directions for the two eyes due to the large size of the displays relative to the interpupillary distance of the user of the HMD. Hence, the most general case is to have two distortion tables, one for each eye.

We now have enough information to construct the predistortion lookup table, and this is given as algorithm 2 below.

Algorithm 2

Steps to construct the predistortion lookup table.

1. For all table entries (x_tab, y_tab) with 0 < x_tab < xsize and 0 <y_tab < ysize
   do steps 2 through 6.

2. Convert (x_tab, y_tab) to screen coordinates (X_s,Y_s)

$$X_s = (x\_screen-1)\left(\frac{x\_tab}{xsize-1}\right) \qquad Y_s = (y\_screen-1)\left(\frac{y\_tab}{ysize-1}\right)$$

3. Adjust (X_s,Y_s) for the coordinates to become relative to the optical axis

$$X_s = X_s - A_x \qquad Y_s = Y_s - A_y$$

4. Predistort the coordinates X_s and Y_s using Algorithm 1 yielding X_i and Y_i, respectively

5. Adjust (X_i,Y_i) so that the coordinates become relative to the upper left corner of the screen

$$X_i = X_i + A_x \qquad Y_i = Y_i + A_y$$

6. Fill in the table value with

   table[x_tab][y_tab][x] = X_i and table[y_tab][y_tab][y] = Y_i

Given the corresponding A_x value for each eye, a lookup table for either eye can be built using Algorithm 2.

Figure 4 shows the positioning of the two LCD displays of center C with respect to the optical axis A, in the most general case. Using Figure 3 and 4, the coordinates of point "A" in units of pixels is given for the right eye by

$$R: A_x = a + \left(\frac{e}{f}\right)(b-a) \qquad A_y = c + \left(\frac{g}{g+h}\right)(d-c) \quad . \tag{20}$$

For the left eye, A_y is the same as for the right eye, while A_x becomes

$$L: A_x = a + \left(\frac{f-e}{f} \right)(b-a) \quad . \tag{21}$$

4. Application of Predistortion Correction to two HMDs

We have applied our correction algorithm to two optical systems, the VPL EyePhone and the Virtual-Research (VR) Flight-Helmet HMDs. Both systems use the LEEP optics designed by Erik Howlett (Howlett, 1983) who kindly provided us with the optical specification for the lenses. The lens specification by itself is insufficient to implement the distortion-correction algorithm, since the transverse magnification of the optical system must be known. Magnification is a function of the positions of the display screens with respect to the optics, positions that were provided by VPL and VR for the Eyephone and the Flight Helmet HMDs, respectively. Those different positions for the two systems yielded different magnifications *m* for the EyePhone and the Flight Helmet HMDs as summarized in Table 1.

The substantial difference in transverse magnification between the EyePhone and Flight Helmet HMDs arises from the fact that a change of a few millimeters in the screens position with respect to the nearest optical surface causes a large change in the distance of the virtual image from the user's eye. For the LEEP optics, as the LCD screen is moved outward to approach the LEEP focal point, the virtual image goes to infinity (Figure 11, Robinett & Rolland, 1992). However, the angles at which image points are seen changes very little as the magnification changes.

A common source of error in implementing a distortion-correction function is to overlook the importance of the video overscan defined earlier. We previously mentioned its importance in using the lookup table correctly; it is also an important parameter in specifying the FOV required by the graphics library. It is especially important if predistortion is to be performed, since the amount of third-order optical distortion varies as the cubic power of the FOV height.

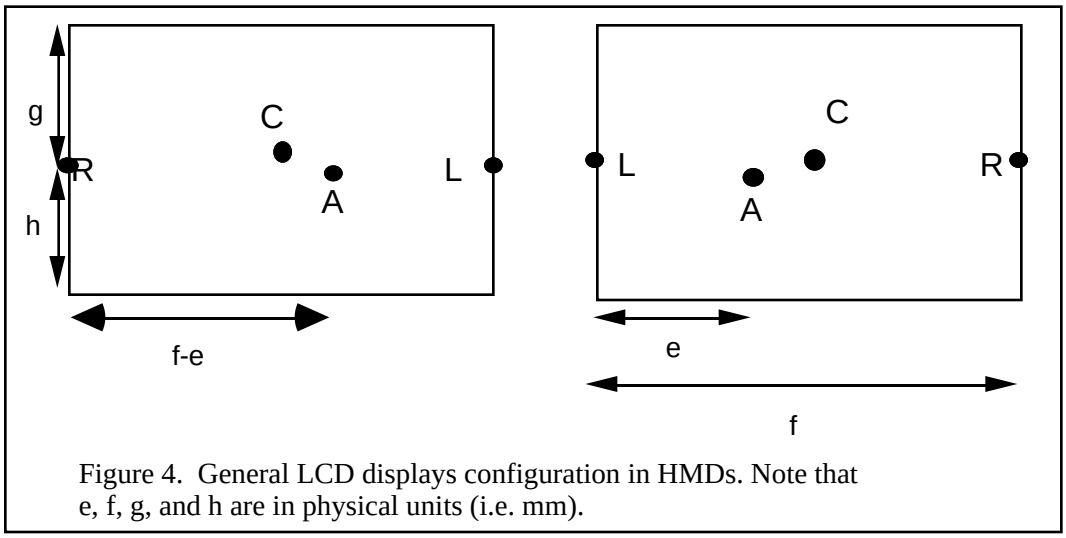

Figure 4. General LCD displays configuration in HMDs. Note that e, f, g, and h are in physical units (i.e. mm).

We measured for the Flight Helmet HMD a vertical video overscan of 35 lines. This means that only 477 lines among 512 lines are being displayed on the LCD screens. Moreover, the vertical dimension of the screens was measured to be 41.7 mm. So if 477 lines correspond to 41.7 mm, 512 lines correspond roughly to 44.76 mm. Therefore R_S equal 256 pixels correspond to 22.38

mm. The value of Rs in mm can be used to find the percent distortion of the lens at a certain point in the FOV using an optical raytracing software. By substituting the magnification, the percent distortion and its corresponding R_S in pixels in Equation 7, k can be determined. Values of k for the VPL Eyephone and the Flight Helmet HMDs are given in Table 1 for an eyerelief of 25 mm. In any case, the values of k would only be approximate for any other eyerelief values since the amount of distortion in an optical system is a strong fonction of the pupil position with respect to the optical system. The overscan of the VPL Eyephone display was not measured due to the difficulty involved in trying to access the displays. Because, we were told by the manufactors that the two displays used in the two systems were quasi identical, we used the same video overscan values for both systems. If critical to your application, it would be necessary to measure the video overscan for each system and recalculate the number of pixels corresponding to R_S equal 22.38 mm. This new value of R_S in pixels could then be used to recalculate the coefficient of distortion k.

The screen coordinates of the point at which the LEEP optical axis intercepts the LCD screen is also different for these two systems and the values of A_x and A_y are also given in Table 2. Those were calculated using Equation 20 with a equal 16, b equal 619, c equal 7, and d equal 484 pixels. Due to the similar displays used in both systems, we have assumed the same values for a, b, c, and d in the calculation of A_x and A_y. Rigorously, they should be measured for each system. The distances e, f, g, and h shown Figure 3 and their values are given in Table 2 for the VPL Eyephone and the Flight Helmet, respectively.

Conclusion

We have described how to computationally predistort an image using vertices warping. The emphasis of this work is on the mathematical framework necessary to perform the predistortion. It turns out that the two algorithms described can be applied to pixels warping as well. To get satisfactory results with vertices warping, large polygons need to be cut into small pieces. An in depth description of advantages and disadvantages of diverse methods such as vertices warping, pixels warping, and texture warpings will be described in a following manuscript with more emphasis on the computer graphics aspects of this work.

Table 1

	VPL Eyephone	**Flight Helmet**
m	9.66	28.55
R_S (mm)	22.38	22.38
Rs (pixels)	256	256
% distortion	17.88	19.33
k (pixels^{-2})	$2.92\ 10^{-8}$	$3.62\ 10^{-9}$

Table 2

	VPL Eyephone	**Flight Helmet**
e (mm)	20.7	20.9
f (mm)	54.2	54.8
g (mm)	21.8	20.85
h (mm)	18.5	20.85
A_x(pixels)	L: 388.70 R: 246.30	L: 389.02 R: 245.98

$\mathbf{A_y}$(pixels)	265.03	245.50

Acknowledgments

We especially want to thank Mark Olano for writing the PPHIGS library function pg_distortion, and Anselmo Lastra for his stimulating discussions and guidance in implementing the distortion correction. We thank the Head-Mounted Display team and especially Gary Bishop for providing supervision of this work and Vern Chi for his help with planning some of the measurements. We extend special thanks to Warren Robinett for suggestions about this report, for writting part of section 3.1, and for the stimulating discussions we shared during this work. We thank Carl Mueller, Chip Hill, and Jim Chung for helping with implementing the distortion correction. We thank Jack Goldfeather for his contribution to multiple aspects of this work over the years.

References

Adelstein B.D., E.R. Johnston, and S.R. Ellis (1992), "A Testbed for Characterizing Dynamic Response of Virtual Environment Spatial Sensors," Fifth Annual ACM Symposium on User Interface Software and Technology.

Beyer, W. (1984), CRC *Standard Mathematical Tables* . CRC Press, INC. Boca Raton, Florida.

Foley J., A, van Dam, S. Feiner, J. Hughes (1990). *Computer Graphics: Principles and Practice* (2nd ed.). Addison-Wesley Publishing Co., Reading MA. 222-226

Fuchs H., J. Poulton, J. Eyles, T. Greer, J. Goldfeather, D. Ellsworth, S. Molnar, G. Turk, B. Tebbs, and L. Israel (1989), "Pixel-Planes 5: A Heterogeneous Multiprocessor Graphics System Using Processor-Enhanced Memories," Computer Graphics, 23 (3), 79-88

Hecht E., and A. Zajac (1974). *Optics*. Reading, MA:Addison-Wesley

Howlett E.M. (1983). "Wide angle color photography method and system," U.S. Patent Number 4,406,532.

Longhurst R.S. (1973). *Geometrical and physical optics*. New York: Longman.

Robinett W., and J.P. Rolland (1992), "A Computational Model for the Stereoscopic Optics of a Head-Mounted Display," Presence, I (1).

Robinett W., and R. Holloway (1992), "Implementation of Flying, Scaling, and grabbing in Virtual Worlds," *ACM Computer Graphics: Proceedings 1992 Symposium on Interactive 3D Graphics* (Cambridge, Mass., April 1992), 189-192.

Robinett W., and R. Holloway (1993), "The visual display computation for virtual reality," (in press)

Rolland, J.P., C. Burbeck, W. Gibson, D. Ariely, "Towards quantitative assessment of depth and size perception in virtual environments," Technical Report TR-93-044, Dept. of Computer Science, University of North Carolina, Chapel Hill (1993).

Smith W. J. (1966). *Modern Optical Engineering, The Design of Optical Systems.* Mc. Graw Hill, Inc.

Weisman P. (1992). Williams Airforce Base. Personal Communication.