Shape Defined Panoramas

John Brosz and Faramarz Samavati
Department of Computer Science
University of Calgary, Canada
Email: {jdlbrosz, samavati}@ucalgary.ca

***Abstract*—Panoramic projections are often defined by the geometric surfaces used to derive the projections' equations (e.g., spherical and cylindrical panoramas). The parameterization of these surfaces greatly affects the resulting projection equations and image properties. Problematically, unusual parameterization can reproduce panoramas associated with other shapes. In this paper, we ensure an explicit link between surface shape and projection behavior by suggesting use of projection surfaces parameterized by arc-length, binding rendering behavior to surface modeling. This allows us to create new panorama variations beyond the conventional for creating panoramas of CG environments as well as for resampling panoramas created from cameras. Further we describe an interface for composing these panoramas and show how this technique lends itself to controlling distortion and composition of panoramic projections. Additionally we provide details on rendering these projections.**

***Keywords*-Panorama; Parameterization; Parametric Surface; Rendering; Image generation;**

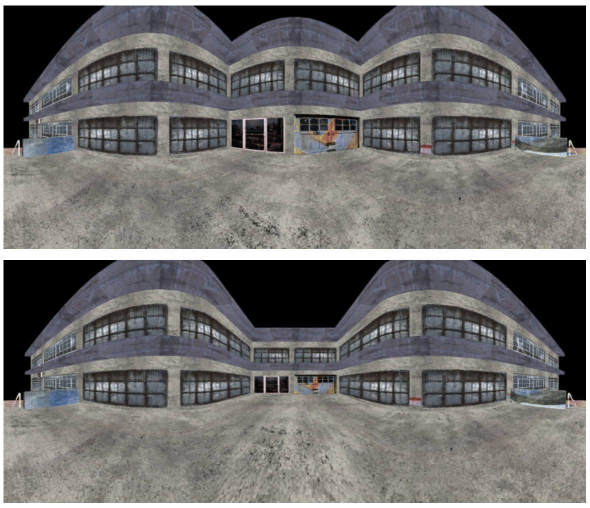
Fig. 1. A cylindrical panorama (top) is altered to display center of the image in perspective (bottom).

1. Introduction

Generally a panorama is an image depicting a wide angle view. In this work we specifically concern ourselves with full-view panoramas, that is panoramas with a single view position that encompass the entire 360 degree surroundings. For brevity, we refer to these full-view panoramas simply as panoramas. In photography these panoramas are created by taking several photographs while rotating the camera around a fixed point and then stitching them together into a single image. The word panorama was coined in the late 1700s to describe Robert Barker's paintings on the inside of large cylinders [1].

Many types of full-view panorama exist; common examples include cylindrical, spherical, and conical panoramas. These panoramas differ from one another both mathematically and in image characteristics. While these descriptions in terms of shape provide hints of definition, these are concretely defined through projection equations (i.e., mappings from three dimensions to two). An unfortunate aspect of these equations is in order to modify them, one must change the surface or its parameterization and re-derive a projection equation, a non-trivial task especially when several iterations are required to create the desired effect. Creating a panorama in this fashion is equivalent to "hard coding" a scene description, while serviceable, it greatly impairs the ability to explore other possibilities.

There are many reasons why such variations may be useful. For instance, a panorama that combines a spherical and a cylindrical panorama reduces the vertical distortion associated with spherical panoramas while capturing the surroundings above and below the viewing position that would be missed by a cylindrical panorama. Another possibility is changing areas of the image to behave as if created by a linear perspective projection thus avoiding the curving of straight lines produced in most panoramas (as shown in Figure 1). Additionally, custom panoramas could be useful for tailoring projections to produce images exactly suited for specific configurations of cave or dome environments. While customized panoramas can be useful for creating panoramas from existing (camera) images, the primary usefulness of this work is to provide new techniques for creating imagery from CG environments.

In this work we create panoramas by defining a projection surface. This can be any arbitrary surface that encompasses the scene and viewing position. However, the shape of the surface is not enough to define the panorama projection; we also require some specification of how to unwrap and flatten this surface to create a 2D image parameterization. We show that arc-length parameterization can naturally map existing panoramas to their characteristic surfaces (i.e., sphere for spherical panorama and cylinder to cylindrical panorama). Therefore, we use arc-length parameterization to create a unique correspondence between the shape of the projection surface and the characteristics of the produced image. This correspondence makes the panoramic projections' behavior visualizable by displaying the projection surface, allowing customized panoramas' behavior to be more easily understood and predicted.

To facilitate the creation of panoramas we have developed an interface for creating arc-length parameterized projection surfaces. This system is based on Brosz et al.'s [2] Flexible Projection Framework that provides a basis for using parametric surfaces to define projections. Additionally we show how the produced panoramas can be efficiently rasterized with current GPU hardware. Lastly, we describe several panoramas designed for particular purposes and further applications.

This work contributes:

- a modeling based technique for composing full-view panoramas from synthetic 3D models.
- arc-view parameterization that allows for explicit control over the distortions necessarily present in full-view panoramas.
- a GPU-based rasterization approach that drives real-time single-pass rendering of panoramas.
- applications for creating panoramic projections within virtual environments (including games and CG generated image/video), re-sampling of real-world photographed panoramas, and interactive local editing of these panoramas.

2. Related Work

In this work we introduce a nonlinear, non-physically based camera. There exists a wide variety of such cameras. Works such as Yang et al. [3] and Carpendale and Montagenese [4] describe nonlinear cameras that operate upon images. Nonlinear cameras that operate in 3D geometry have been described by Inakage [5], Glassner [6], and Sudarsanam et al. [7] and many others. We refer readers to the Brosz et al. [2] where a wide overview of many such cameras is given. For mathematical descriptions of linear and nonlinear projections we refer to Salomon's book [8]. Salomon includes derivations of projection equations for a several panoramas including cylindrical, spherical, and conical.

In applying panoramas Greene [9] describes how projection onto a cube can be used for efficient environment mapping. Glaeser and Gröller [10] describe the use of segments of spherical projections to reduce wide angle distortion. Polack et al. [11] perform a study showing that cylindrical projections improve size and depth perception. Szeliski and Shum [12] describe one of many methods in which image sequences taken with perspective projection can be stitched together into a single panoramic image. Their technique is noteworthy in that it works without explicit knowledge of camera orientation.

Many works construct multi-viewpoint panoramas; projections constructed by combining images from different viewpoints into a single image. Rademacher and Bishop [13] create single images from long, oriented camera paths for the purpose of replacing image sequences in image-based rendering. Román et al. [14] and Agarwala et al. [15] piece together images taken along a roughly planar path. Wood et al. [16] create multi-perspective panoramas that provide changes in viewing direction, position, and field-of-view within a single image for the purpose of creating backgrounds for cel animation. Yu and McMillan [17] create panoramas modeled after Wood et al.'s [16] panoramic constructions by combining the output from many different types linear cameras. These cameras must be manually positioned and adjusted to achieve the desired image as well as be placed following at set of adjacency rules to ensure image continuity. Degener and Klein [18] describe a technique where the 3D model is deformed rather than the camera to create appealing images. Agarwala et al. [15] review many other multi-viewpoint panoramas.

Several works have focused on removing distortions in panoramic or wide-angle images. Zorin and Barr [19] correct photographic, perspective images by adjusting the interplay between perspective and spherical (direct view) projections. Zelnik-Manor et al. [20] remove distortion in single-viewpoint panoramas by stitching together either horizontally or vertically neighboring perspective photographs into a single image. Discontinuities at seams are handled by user-assisted placement at feature edges. Due to this technique's ability to only stitch in one direction it cannot address deformation near the poles of the panoramic projections. Most recently Carroll et al. [21] perform optimization on wide angle images with the guidance of user-provided indications of straight lines. This work is not demonstrated for full-view panoramas and cannot resolve distortions in images that feature parallel lines that converge at on-image vanishing points [21].

In our work we have chosen to constrain ourselves to single-viewpoint panoramas that encompass the viewpoint's entire 360 degree surroundings. A strongly related variety of this type of projection are the many projections that have been used in making maps [22]. Most related to our work is Trapp and Döllner's [23] system for creating full view panoramas. In their work single-center projections, including but not limited to panoramas, are created by specifying a normal map. The normal map controls the projection, specifying the direction from the center of projection where objects will be projected. Rendering is accomplished in realtime by creating a cube environment map that is resampled with hardware in a second rendering pass to create the projected image.

In comparison, our technique has several strengths. The first is that we describe an object space-based rendering technique that avoids image sampling issues. The second is that our technique uses surface modeling rather than normal maps to specify our panoramic projections; this allows the use of existing modeling tools to ease the task of specifying panoramic projections.

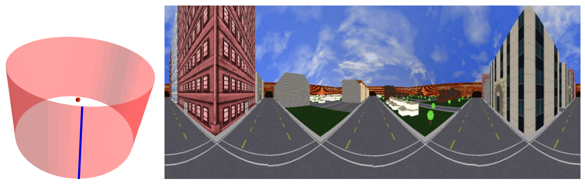

Fig. 2. A cylindrical panoramic projection surface with marked seam and center of projection (left) and projected image (right).

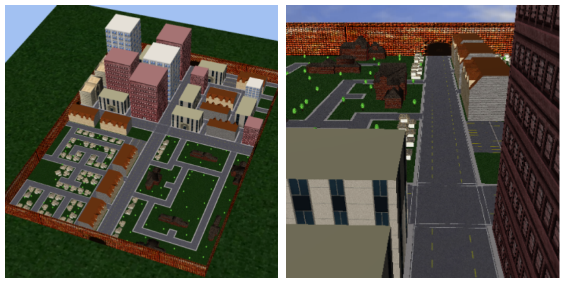

Fig. 3. Perspective views of the city model used to demonstrate the panoramas.

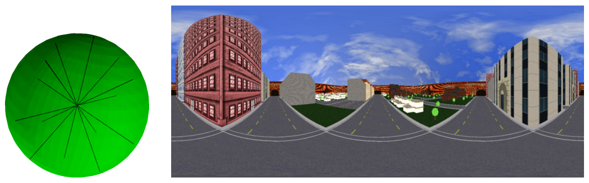

Fig. 4. A spherical panoramic projection surface (left) and projected image (right). The blue lines within the surface indicate rays used that might be used to a column of the image.

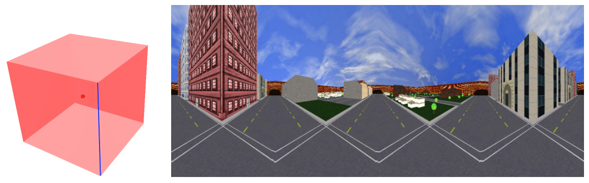

Fig. 5. A cubic panoramic projection surface with marked seam (left) and projected image (right).

3. Panoramas

Panoramas can be described as projections onto cylinders, spheres, cubes, or other surfaces that surround a viewing point. This viewing point, also known as the center of projection, is a point at which we might imagine the viewer's eye to be positioned. The up axis is the axis around which the viewing direction is rotated so that the entire 360 degree surroundings can be viewed.

The process of creating a panorama can be broken into two steps: projection through the eye onto some sort of projection surface; then mapping that surface to a flat, usually rectangular, image. This mapping relies on a 2D parameterization of the projection surface onto a rectangle; creating a seam by splitting the surface and flattening it. A difficulty is that many such mappings can exist depending on the 2D parameterization.

Our intention in this work is to develop a geometric technique for creating new panoramas that builds on the specification of existing panoramas. To provide the reasoning behind our chosen technique we begin by examining common types of panoramas: cylindrical, spherical, and cubic.

A *cylindrical panorama* is most often the result of projection onto an open-ended cylinder; its center coincident with the eye position and its primary axis of aligned with the panorama's up axis (Figure 2). The original model is shown in perspective in Figure 3. After projection onto the cylinder, the parameterization is created by cutting the cylinder's surface to form a seam and then unrolling a flat image.

Let us consider this projection as if we were ray tracing it. When tracing a row of pixels, each pixel's ray will be the same as the previous pixel's, but rotated by some constant angle around the up axis. This is due to the one-to-one mapping between the pixels and points spaced equally around a circle on the surface of the cylinder. For a column of pixels the situation is different since they sample a line segment parallel to the cylinder's primary axis. Each pixel is spaced equidistant along the line segment causing the angle between samples to vary. This angle will be largest as rays near to perpendicular with the up-axis and smaller as they approach parallel. Let us refer to the angle between pixels' rays (or samples) for the rows of the image as θ and the angle between pixels of the columns as ϕ. So, for this type of cylindrical panorama, θ experiences a constant change while the change to ϕ varies.

A *spherical projection* is created by projecting objects onto a sphere. Flattening the sphere creates trade-offs between preserving area, straight lines, angle, scale, and shape [22]. For simplicity we discuss the simplest approach, i.e., an equidirectional spherical projection. This uses the sphere's longitudinal and latitudinal coordinates directly as x and y coordinates in the image as demonstrated in Figure 4.

Now let us examine this projection in terms of θ and ϕ. As in the cylindrical panorama θ experiences constant change between neighboring row pixels. The change to ϕ in this scenario is also constant since each pixel will be separated by a constant arc-length.

The last panorama we review is a *cubic panorama* created by projecting onto a cube with an open top and bottom. In essence, this panorama is a concatenation of four separate perspective projections (each rotated 90 degrees around the up axis from its neighbor). A cubic panorama is shown in Figure 5.

Examining cubic panoramas in terms of θ and ϕ we find that both angles change variably due to equidistant steps along the planar sides of the cube. The variation in angle is largest at the center of each face and smallest at the cube's edges.

By setting the properties of each of the described surfaces we can derive projection equations (see Salomon [8] for an example) and then use these equations to create panoramic images from virtual scenes. While this derivation is straightforward it becomes markedly more difficult if one wishes to experiment or alter these projections. For instance, what if we desire to place the center of projection away from the cylinder's center? It would be convenient if we could automate this derivation process for a variety of shapes and possibilities.

In Figure 6 we compare the θ sampling of a cylindrical to a cubic panorama. If we parameterized a cylinder such that the θ sampling matched that of the cube's samples projected onto a cylinder we could exactly reproduce a cubic panorama with a cylindrical projection surface. This makes the projection surface unimportant in determining the projection since changing the surface parameterization is alone sufficient to control the projection.

So if the surface does not matter, then why use a surface? Working with and changing parametrization to control projections is not an intuitive operation. It is akin to using a hard coded scene description rather than a external controlled scene description that can be visualized and easily modified. Our approach makes use of the ability of surfaces to be similarly visualized and that they are already associated with conventional panoramic projections. Consequently our goal is to create a technique for specifying panoramic projection surfaces that creates a fixed correspondence between parameterization and the surface shape.

These three types of panorama introduce important differences in image characteristics. In general panoramas feature trade-offs between spherical projection and planar projections. This is exactly the observation made about projections in general by Zorin and Barr [19]. Spherical projection provides direct viewing, making all parts of the image appear as if they are viewed straight on. This matches the way we rotate our eyes to examine our surroundings. The drawback is that while isolated areas of the image appear correct, the entire image is viewed as a whole the result seems strange. These image characteristics are produced by constant angle sampling in the image (e.g., in spherical panoramas and in the sampling of θ in cylindrical panoramas.

Planar projection maintains a global coherence, preserving straight lines. Unfortunately, this projection introduces distortions as one samples away from the center of projection, leading to problems with wide angle images, as well as in projections of circles and spheres [19]. These characteristics are seen in cubic panoramas and in the sampling of ϕ in cylindrical panoramas.

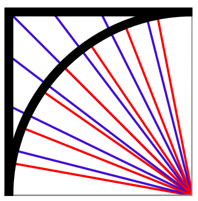

Fig. 6. Comparison of sampling of a quarter square (blue) and a circle (red), analogous to a sphere and cube.

In deciding upon a particular panoramas one selects a particular balance between spherical and planar projections; our aim is to provide more freedom in adjusting this balance to create desired image properties.

4. Controllable Panoramas

Our discussion thus far has lead to us to the conclusion that in order to control the composition of panoramas we must have control over the sampling of θ and ϕ (i.e., the spacing between the projection rays used to create the image).

In our review of common panoramas two mechanisms control the panorama type: the projection surface and the parameterization of this surface. Due to conflicts between these mechanisms we suggest constraining definition of panoramic projections entirely to the shape of the projection surface. We will accomplish this by making the parameterization entirely defined by the surface shape. Use of only the projection surface to control the panorama has several benefits: modeling 3D surfaces is a common task with a concrete physical analogy and, in general, easier to perform than specifying or modifying a surface parameterization; a 3D surface can be easily visualized providing clear feedback to the user; and common panoramas are currently described based on their projection surface, by continuing to define panoramas by surface shape builds upon existing practices.

The remaining question is how to bind the parameterization to the surface shape. Since our goal is to control the sampling of θ and ϕ it is logical to make use of a parametric surface $Q(u, v)$ and assign one parameter to θ and the other to ϕ. Then, to control the sampling, construct our surface from iso curves that define the sampling. In order to attach the sampling behavior to the curve, we space our samples at uniform distances along the curve thus parameterizing the curve and eventually the surface by arc-length. Thus a flat curve will produce a variable change sampling (as in the cubic panorama) while a circular arc will produce a constant rate of sampling. This follows Farin's suggestion [24] of using chord length to closely approximate arc-length and has the added benefit of allowing use of a wide-variety of analytical curves (e.g., parametric, implicit) as well as discrete ones (e.g., user-sketched curves).

4.1 Projection Surface

We control sampling of θ with a 2D closed curve $P_o(u)$ parameterized by arc-length. We call this curve the *outline*. If we imagine a row of x pixels in our desired panoramic image and the total arc-length of the outline is L, then each subsequent pixel is $\delta L = \frac{1}{x-1}L$ further along the outline than the previous pixel. Areas with high arc-length will occupy more pixels of the image than areas with lesser arc-length. A circle as an outline produces a constant change of θ as seen in cylindrical and spherical panoramas. The cubic panorama has a square outline. Flat areas of the outline cause sampling

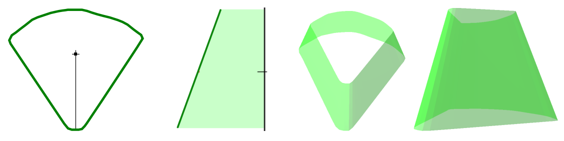

Fig. 7. An arc-length parameterized projection surface created with Equation 1. In outline diagrams the origin is marked; the descending line's intersection with the outline indicates the seam. The front of the projection (the center of the image) is halfway along the arc-length of the entire outline from the seam. Profile diagrams show the profile as well as a light fill to relate the curve to the up axis.

of θ to assume the characteristics of planar projection. The corners of the square are treated as vertices of a piecewise curve.

For control over ϕ sampling we create a *profile*. A profile $P_p(v)$ is a 2D open curve parameterized by arc-length that controls sampling of the columns of the image. For example, the profile of a spherical panorama would be a hemi-circle whereas the profiles of cylindrical and cubic panoramas are vertical line segments parallel to the up axis.

To create a surface from a profile and an outline let us first consider using a circle as our outline. In this case we can use a surface of revolution (see Farin [24] for a formal description). That is, we create a surface by revolving the profile 360 degrees around the up axis. We can also see this surface as an extrusion of a circle with varying scale along the up axis controlled by the profile's distance from the axis.

To move beyond circles, creating surfaces from other outlines, we use the same construction of extrusion using the given outline rather than a circle. The equation for this surface, assuming the outline and profile have both been defined on the XY plane, is:

$$Q_d(u,v) = \begin{pmatrix} P_o(u)_x P_p(v)_x, \\ P_p(v)_y, \\ P_o(u)_y P_p(v)_x \end{pmatrix}, \quad u,v \in [0,1]. \quad (1)$$

This surface bears similarity to the cross section oversketch surface described by Cherlin et al [25]. It should be clear that when the outline is a circle i.e., $P_o(u) = (\sin(2\pi u), \cos(2\pi u))$, that the produced surface is exactly a surface of revolution. Other values for $P_o(u)$ cause the profile to be scaled inward and outward from the origin as shown in Figure 7.

In examining the surface created in Figure 7 we see that outline directly controls the shape of the surface as seen from above. If we take any slice of the surface perpendicular to the up axis we obtain a scaled copy of the outline. If we slice the surface with a plane containing the up axis we obtain two copies of the profile, one a mirror of the other.

With a surface we can evaluate θ by fixing v, allowing u to vary uniformly, and examining the angular change between the points. As the x and z coordinates of the surface are determined by the orientation of ray around the eye position, it should be clear that the outline $P_o(u)$ provides direct control over θ as the uniform scaling by $P_p(v)$ does not alter the angular change. When examining ϕ and fixing u the resulting curve is a non-uniformly scaled version of the profile.

To allow greater control over θ and ϕ we can allow for multiple profiles. These n profiles, $p_0, p_1, ..., p_{n-1}$, are associated with the u values, $u_0, ..., u_{n-1}$ denoting where each profile is positioned along the outline. We order the profiles such that $u_0 < u_1 < ... < u_{n-1}$. For a given u either the value is exactly that of one of the specified profiles (in which case we use that profile), or it is not. If not, we linearly interpolate between profiles p_i and p_{i+1} where $u_i < u < u_{i+1}$. This interpolation is based on the parameter $t = (u - u_i)/(u_{i+1} - u_1)$ thus our profile is $p = t * p_i + (1-t)p_{i+1}$. For end conditions, $u < u_0$ or $u > u_{n+1}$ we interpolate between p_{n+1} and p_0. Figure 8 provides an example of a projection surface created with multiple profiles as well as the resulting image.

While linear interpolation is adequate, we usually prefer specified profiles to have more influence than merely serving as end points of interpolation. Consequently creating a sort of *ease-in, ease-out* effect between specified profiles is desirable. We achieve this using a 1D Bézier based interpolation of $p = ((1-t)^3 + t(1-t)^2)p_i + ((1-t)t^2 + t^3)p_{i+1}$, although other interpolation techniques would serve equally well. This *ease-in, ease-out* causes the profiles to remain similar to the defined profiles over a larger surface area and then transition quickly. Figure 9 demonstrates both techniques.

In examining the resulting surface we can extract the 2D profile by fixing u. This means that this exact profile creates a column of pixels in the projected image. It is also the case that this curve exists at a particular spherical angle θ. This allows us to align a profile to a world object. While not immediately important this is noteworthy later when we discuss combining multiple profiles and multiple outlines in the same surface.

Similarly we can also allow for multiple outlines. If we desire multiple outlines and a single profile we can use the m outlines $q_0, ..., q_{m-1}$ exactly as we did the multiple profiles by specifying the profile points at which each outline is attached to. Because there is no wrapping at the edges we assume q_0 extends fully to the bottom and q_{m-1} extends fully to the top. Elsewhere the surface is created by interpolating between the outlines.

If we desire to have both multiple profiles and multiple outlines we must choose how to attach profiles and outlines to one another in order to form the surface. If we choose to again use arc-length parameter values (u and v) we preserve the effect of outlines corresponding to rows in the image and profiles to columns. The alternative, our preferred approach, is to attach the curves at specified spherical coordinates (θ and ϕ). This causes outlines and profiles to curve through multiple rows/columns in the image but ensures that the profile and outline curves exist undistorted in the surface. This is useful as it means we directly link outlines and profiles with objects in world space. A drawback of this

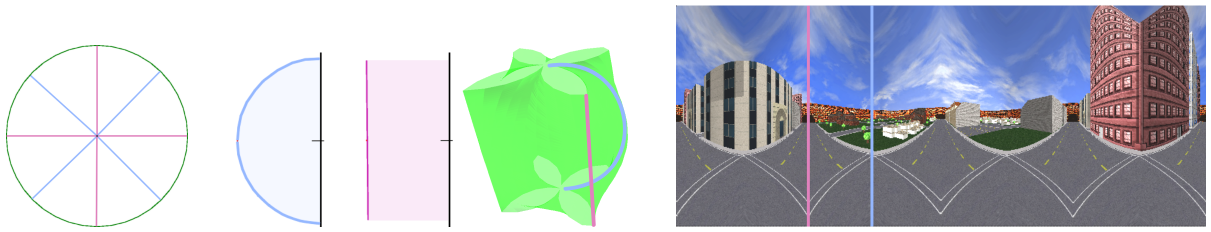
Fig. 8. An example of multiple profiles blended with linear interpolation. From left to right the images show the outline, the profiles, the surface and the panoramic image. The colored lines on the surface and image indicate positions where a single profile has influence.

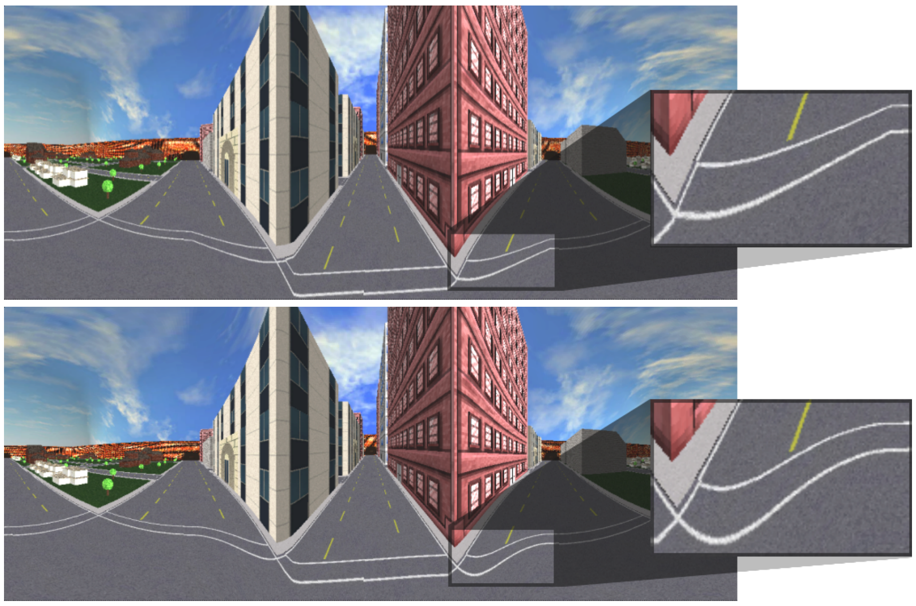
Fig. 9. Comparison between linear (top) and Bézier based (bottom) interpolation between profiles. Transitions between profiles are most noticeable in the magnified area examining the crosswalk lines. With linear interpolation the crosswalk lines transition sharply; Bézier interpolation creates a curved transition.

approach is that it limits our surfaces to those that map one-to-one with spherical coordinates; this is not a large drawback as rendering parts of the scene more than once is undesirable in most applications.

With this design choice made each profile $p_0, ..., p_{n-1}$ is positioned by a spherical coordinate $\theta_0, ..., \theta_{n-1}$ and each outline $q_0, ..., q_{m-1}$ is positioned by $\phi_0, ..., \phi_{m-1}$. Interpolation between specified curves is performed as in previous scenarios.

Our panoramic surface equation for multiple outlines and profiles becomes:

$$Q_d(u,v) = \begin{pmatrix} P_{o_{f(\theta)}}(u)_x P_{p_{g(\phi)}}(v)_x, \\ P_{p_{g(\phi)}}(v)_y, \\ P_{o_{f(\theta)}}(u)_y P_{p_{g(\phi)}}(v)_x \end{pmatrix}. \quad (2)$$

where f and g are the interpolation functions used to determine the profile and outline curves respectively thus $P_{o_{f(\theta)}}$ is the outline interpolated from specified outlines based on the spherical coordinate θ.

It is important to note that our panoramic projection surfaces are constrained by the defining outline(s) and profile(s) so that some enclosing surface shapes are not possible (for instance one cannot create a twisting around the camera's up axis). This is a design feature that emerged out of experimentation and finding, as also noted by Carroll et al. [21], that panoramas are often oriented so that the up vector is parallel to vertical lines in the scene and that such lines should remain vertical in the resulting image. Profiles, and their attachment to specific θ values ensure these vertical lines; similarly outlines maintain the horizontal orientation of lines orthogonal to the up axis.

4.2 Outline and Profile Behavior

When manipulating outline and profile curves there are two major operations that produce specific image outcomes. The first is making a section of (or the entire) curve a straight line. This creates an area of planar projection. For true planar projection it is necessary to have straight lines in both outline and the profile to produce a truly flat area on the surface.

The second operation is altering the curve to change its arc-length. Reducing the arc-length causes fewer samples to be taken over that area of the image. This is useful if you wish to compress an area of the image. Increasing the arc-length of the shape causes more samples to be taken over the corresponding area of the image and expands an area's resolution.

5. Rendering

An important consideration in rendering is that all projector rays originate at the eye and are then directed outward at the entire 360 degree environment. Consequently the specification of the different panoramas only affects the distribution of these rays; visibility, lighting, and other phenomena do not change with the alterations to the panorama specification. Thus it is possible to render a spherical projection of the environment and then use two-pass rendering with image resampling operations to achieve the desired panorama. This is the technique described by Trapp and Döllner [23] although they make use of a cube map rather than spherical projection. While this technique is definitely applicable to our panoramic projection surfaces, the use of image resampling in this technique problematic due to changes in resolution and aliasing issues.

The fact that our surfaces are defined in object space provides additional rendering options that avoid the problems from resampling images. Ray tracing, a slower process, uses the arc-length parameterized surface to define ray direction. Rasterization, the process of using a projection equation to place triangle vertices and then using fill algorithms to rapidly draw the triangles.

5.1 Ray Tracing

To ray trace these panoramas we draw upon the Flexible Projection Framework [2]; defining the projection's viewing volume by an interpolation between a point (the eye) and the arc-length panoramic projection surface. This gives us a parameterized viewing volume $Q(u, v, t)$ where t parameterizes the depth of the projection, and u, v correspond to the parameters of Q_d. If we wish to adjust the distance of the near and far surfaces from the eye position we adjust the range of t. Rays are created by casting a ray from the center of projection through the surface at fixed u, v. To ensure the sampling specified by the projection surface is propagated to the image we should iterate through the u and v with equal step sizes; for instance pixel (i, j) should be traced with the ray $Q(t) = Q(i/width, j/height, t)$.

While this technique is much slower than the other techniques it is important to note that it is extremely generic as we are only describing how rays should be positioned and oriented. Consequently these panoramas can be traced in anything from most efficient realtime ray casting techniques to the most detailed and accurate ray tracing approaches.

5.2 Rasterization

In our rasterization approach we develop a projection equation that allows us to project triangle vertices based on the panoramic surface Q_d. This nonlinear projection can be performed on the GPU with vertex or geometry shaders [26] by simply substituting our nonlinear projection equation for the standard projection matrix. The remaining work of rasterization (i.e., filling, clipping, etc) is completed by the hardware's default algorithms.

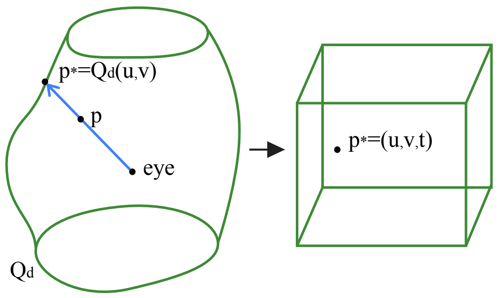

Fig. 10. When rasterizing we transform p from camera coordinates (x, y, z) to parameterized volume coordinates (u, v, t). We can find (u, v) quickly by identifying the ray (with unique spherical coordinates) that passes through they eye, p, and p^* as shown on the left. On the right we produce interpret and rescale the volume parameters (u, v, t) as existing in the normalized device coordinate viewbox.

Unfortunately, these algorithms assume the use of linear projection causing two issues to be dealt with: seams and accuracy. The remainder of this subsection discusses the projection equation, handling seams, and the accuracy of the filled triangles.

Projection Equation. Our main task in rasterization is to develop a projection equation that will transform a point in camera coordinates (x, y, z) to a projected point on Q_d and from there to normalized device coordinates. To accomplish this we again define panorama's projection volume by bounding it between the eye position and Q_d:

$$Q(u, v, t) = (1-t)(0, 0, 0) + tQ_d(u, v) \quad 0 \le u, v, t \le 1.$$

We can create near and far depth clipping by altering the range of t. The inverse $Q^{-1}(x, y, z) = (u, v, t)$ is our desired projection equation. With Q^{-1} we can rescale (u, v, t) to provide normalized device coordinates. This inverse equation is nonlinear and, depending upon the profile(s) and outline(s), can be difficult to analytically derive.

To simplify the inverse calculation we limit ourselves to surfaces that map onto spherical coordinates. While this does introduce a restriction, it is only necessary for rasterization and ensures that spherical coordinates uniquely map to points on Q_d; allowing us to use spherical coordinates as an intermediate step in conversion between (x, y, z) and (u, v, t). With spherical coordinates of p, we can search for the point on the surface Q_d with the same spherical coordinates. As shown in Figure 10 these coordinates uniquely identify the ray passing through the eye, p, and p^* on Q_d. Knowing $p^* = (x^*, y^*, z^*)$ on Q_d and the associated parameters (u^*, v^*) we only have to calculate depth. Depth is the ratio between distance of the (x, y, z) from the origin to the distance of p^* from the origin. The algorithm is:

1) find spherical coordinates (θ, ϕ) of coordinate (x, y, z)
2) search to find point $Q_d(u^*, v^*) = (x^*, y^*, z^*)$ with the same spherical coordinates as (x, y, z)

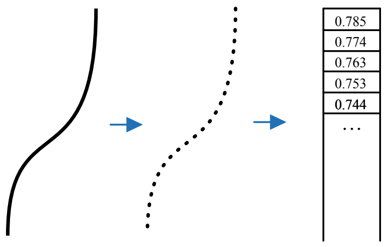

Fig. 11. Precalculation for outline and profile curves. The curve (left) is first sampled into point by arc-length (middle) and then the appropriate spherical coordinate (θ for outlines, ϕ for profiles) is stored in an ordered array (right).

3) $(u, v, t) = \left(u^*, v^*, \frac{\sqrt{x^2+y^2+z^2}}{\sqrt{x^{*2}+y^{*2}+z^{*2}}}\right)$.

Note that t (proportional to depth) is necessary for occlusion testing and clipping.

Explanation is necessary to describe how we search to find (u, v) from spherical coordinates. The first step of this search requires precalculation on the CPU. That is, for each outline and profile, we calculate $n+1$ points, equally spaced by arc-length, along the curve. The value $n + 1$ should be chosen to balance the accuracy needed for the number of pixels across the image without requiring excessive memory; we find a value of half the image's larger dimension works well. We store the list of ordered points by converting each point into spherical coordinates, recording θ for outlines and ϕ for profiles (see Figure 11). This data, along with the attachment points for each curve (also in spherical coordinates), are passed to the graphics card for a given Q_d and only need to be recalculated when Q_d is changed.

The GPU, given this data and a point's spherical coordinates (θ, ϕ) calculates (u, v). This begins by calculating u. In the case of a single outline we find the array entries i and $i+1$ that bound θ and u is calculated by interpolating between these the array values. Care must be taken in the boundary case when θ is between array entries 0 and $n - 1$. With multiple outlines we must include the additional step of interpolating between outlines based on ϕ and the interpolation function $g(\phi)$ from Equation 2. The calculation of v is similar making use of the profiles and ϕ. The only key difference are the boundary cases. As profile curves are not closed and do not entirely surround the center of projection there will be points beyond the first and last array entries that should be culled from the image by assigning values outside the range of the surface (i.e., $v \notin [0..1]$).

Seams. Seams refer to where the projection surface has been split to be flattened. Without special handling, triangles that intersect this seam become spread across the image. This is the result of one of a triangle's vertices being projected to one side of the image, the other vertices projected to the other side, and the linear fill algorithm interpolating between them. With geometry shaders this problem can be handled by testing for triangles that overlap the seam. When found, a copy of the triangle is created; the original is placed entirely on the left side of the image, the copy on the right.

Accuracy. The accuracy problem is caused by the standard algorithm's use of linear interpolation that do not properly handle the curving of triangle edges caused by nonlinear projection. The effects of this are similar to a coarse approximation of a curved surface and is noticeable when low polygon models are used. Gascuel et al. [27] provide an elegant solution for spherical projections where each triangle's maximum warped area when projected is calculated and then shaders are used to determine the extent of the curved triangle inside this area. While accurate, this technique relies heavily on the regular curving structure of spherical projections to calculate the warped area and thus would be difficult to adapt to our potentially irregular surfaces. Our suggestion is to instead adaptively subdivide triangles with the geometry shader when an edge's midpoint deviates significantly from a linearly interpolated midpoint. Another possibility is to use standard modeling software to apply planar subdivision uniformly to the scene.

In our implementation, that includes handling of seams but not adaptive subdivision, we achieve frame rates of over 60 fps with a model of $\sim 100K$ triangles on an Intel Core2Duo 8400 with 4GB RAM and a NVIDIA 8800 GTS 640 MB graphics card.

5.3 *Choice of Rendering Algorithm*

Choosing a rendering algorithm, is a balance between speed and quality. For the utmost in quality, ray tracing is the clear candidate. When real-time rendering is desired the choice between Trapp and Döllner's cube map resampling [23] (Cube Maps) and Section 5.2's rasterization algorithm (Rasterization) depends upon the type of scene being projected, the shape of the projection surface, and on the desired resolution of the output image.

For scenes of extremely high numbers of polygon it is possible that in Rasterization the number of calculations performed per vertex on the GPU may outweigh the Cube Maps cost of a second image pass and image resampling. For low polygon scenes the Rasterization technique will either yield inaccuracies in image quality as discussed in the preceding subsection or suffer slow downs due to planar subdivision of the scenery models. The image based nature of the Cube Map technique will cause speed reductions when creating higher resolution output that will not be as noticeable in the object-space Rasterization process. Additionally high resolution output will cause Cube Map's reduction in final image quality due to greater aliasing artefacts associated with resampling the relatively lower resolution cube map.

The Cube Maps technique will also suffer increasing inaccuracies as the shape of the projection surface diverges from that of the cube map, especially for areas of the surface corresponding to large arc-length where the cube map lacks sufficient resolution to provide necessary image detail for sampling. Lastly, the Cube Maps technique will suffer when the arc-length parameterization

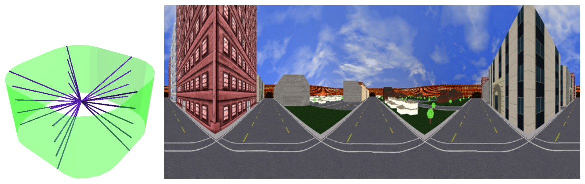
Fig. 12. A cubic panorama modified by smoothing the corners of the square outline. Resulting projection surface (left) and image (right). Compare with Figure 5.

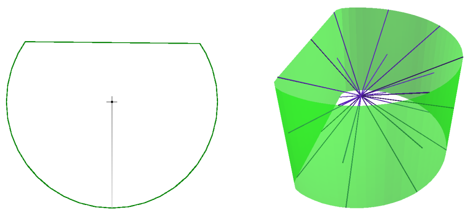
Fig. 13. The outline (left) and resulting projection surface (right) used to create the image shown in Figure 1.

of the cube is significantly different from the arc-length parameterization of the panoramic projection surface. Differences in this parameterization indicate under/over sampling scenarios that adversely affect image quality.

6. Results

In this Section we provide several examples of how panoramas can be tailored to create unique images.

In our first example, Figure 12, we begin with a square outline that creates a cubic panorama. However, we then alter the square, rounding off corners to avoid the sharp discontinuities normally present in cubic panoramas. The produced image provides views down each of the streets in perspective with a smooth transition between each of the four directions.

In motivating this work we mentioned the possibility of altering a cylindrical panorama to contain an area of perspective projection. Figure 1 and 13 provides an example of this where the outline has been flattened to make clear the rectangular nature of the building in the scene.

Our next example begins with the goal of creating a panorama depicting a small hallway that includes the ceiling, floor, and walls. In Figure 15 we begin with a spherical panorama but wish to change two aspects of this image: first we want to minimize the outward bulging of the wall in the middle of the image, secondly we desire to reduce the amount of distortion at the poles of the sphere. To solve the first problem we change the outline, flattening it slightly into an ellipse. In the image this horizontally compresses the walls and expands the ends of the hallways. The second problem is corrected by translating the outline away from the axis so that the top of the surface will project a circle rather than a single point.

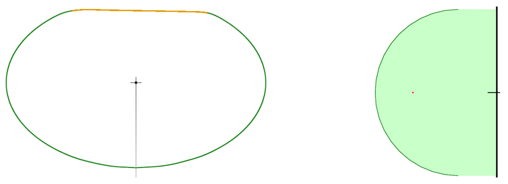
Fig. 14. The outline (left) and profile (right) used to create the changes in Figure 15. The flattened area of the outline has been colored orange for emphasis.

The last example, a panorama of a train station, is shown in Figure 16. In this example the outline was made to be cylinder-like for the bottom third of the image in order to minimize the presence of the dark railway bed. The top of the outline is a hemi-circle translated away from the axis. This provides a spherical projection of the roof and emphasizes its arching nature. The bottom half of the hemi-circle causes elements of the train station at pedestrian eye level to be vertically expanded in the image.

7. Applications

Aside from use of this framework to specify unique panoramas there are several possible applications that take advantage of our panoramic projection surfaces.

Animated Panoramas. The first such application is in creating animated panoramas. Frames from such an animation are shown in Figure 18. An example of a situation where this might be useful is in providing a simulation of perceived peripheral vision for a moving viewpoint. In such a simulation as the viewpoint moves faster the center of projection, within a circular outline, is moved towards the front edge (i.e., opposite the seam) of the outline. This causes the world in front of the viewer to enlarge in the created image and areas beside and behind the viewer to shrink, simulating a change in focus when moving quickly.

Panorama Reprojection. Another possible application lies in reprojecting panoramas. That is, we can reprocess existing panoramic images to appears as if created by a different panorama. That is, if there is an existing panorama this technique can be used to create a different panorama of the same scene. The steps of this conversion are:

1) create a projection surface $Q_1(u, v)$ that corresponds to the projection used to produce the image
2) use $Q_1(u, v)$ to find spherical coordinates for each pixel
3) use $Q_2(u, v)$ to determine the resulting spherical coordinates in each pixel of image produced by the new panorama
4) use both sets of spherical coordinates to sample and determine the pixel values.

Figure 17 provides an example where a cylindrical panoramas is reprojected to another custom-made panorama. The areas sampled outside that of the original cylindrical panorama have been colored black.

Interactive Local Editing. The last application we will discuss in an interactive tool for local editing of panoramic projections. With the described realtime rendering technique it is possible to display and change

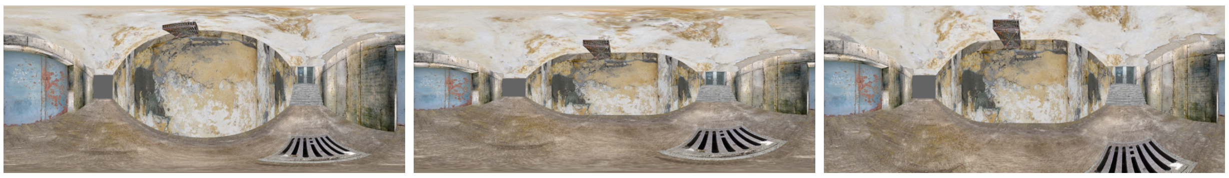

Fig. 15. Altering a spherical panorama (left) of a hallway to produce customized panoramic projections (middle and right).

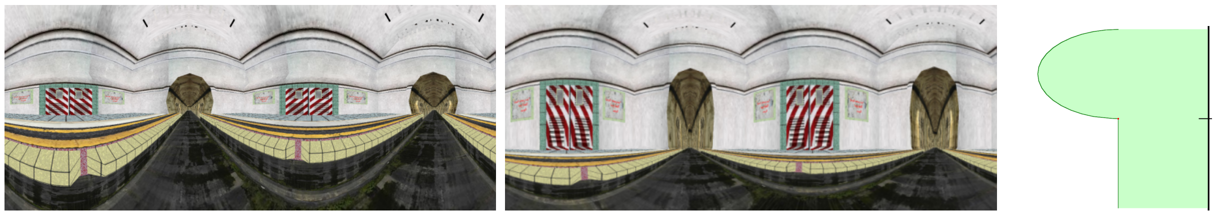

Fig. 16. Altering a cylindrical panorama (left) to a customized panorama (center) that vertically stretches the scene at eye level as is evident in the scene's doors. The projection has also been designed to reduce the amount of image displaying the railway floor while retaining the arched roof. The outline in the altered panorama is a circle, the profile is shown in the rightmost image.

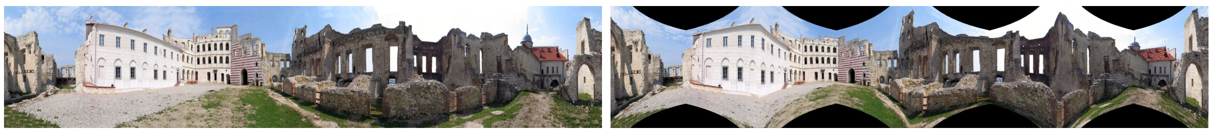

Fig. 17. Left: a cylindrical panorama created from stitched together camera images. Right: reparameterization to meet the specification of the panorama shown in Figure 8.

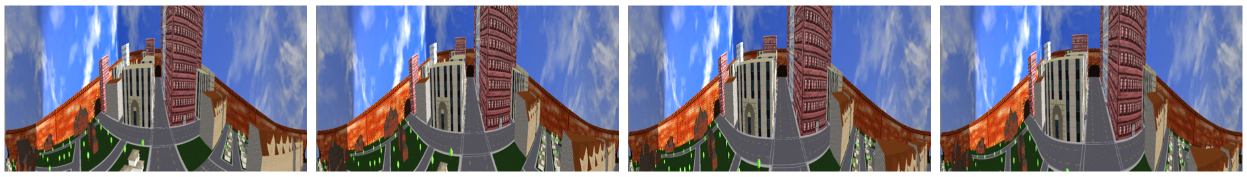

Fig. 18. Four frames from an animation transitioning between a cylindrical panorama and the projection shown in Figures 1 and 13.

the arc-length panorama surface interactively in image space.

In our proof of concept application the user begins with any arc-length panoramic surface and then drags the mouse over the image specifying a rectangular area that they want altered to be projected in perspective. This changed is accomplished by calculating the rectangular extent in spherical coordinates. These coordinates are used to extract outline and profile iso-curves from the projection surface at boundaries of the specified area. We extract four outlines and four profiles, one corresponding to each boundary of the rectangle and another slightly inside the rectangle to provide an area of interpolation between the original panorama and the perspectively projected area. The interior outlines and profiles are flattened to produce a flat area on the surface while the boundary curves maintain the original surface (see Figure 20).

These new outline and profile specifications are added to the surface definition, relayed to the geometry shaders, and the specified alteration is instantly displayed in realtime. Additionally this surface definition can be used to produce a high quality ray traced version of the panorama (Figure 19) This technique can easily be extended to allow other local alterations to be made to a panorama's image.

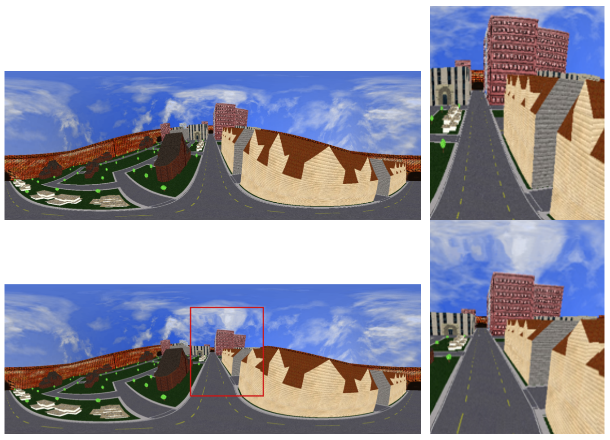

Fig. 19. Interactive editing a panorama in image space. Top: the original spherical panorama. Bottom: the altered panorama. The red lines indicate the altered image region. Right: enlarged comparison of altered region.

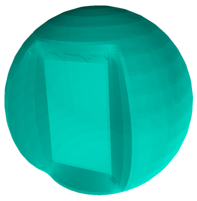

Fig. 20. The arc-length parameterized surface produced by the interactive process shown in Figure 19.

8. Conclusion

In this work we have shown how custom panoramas can be created for a variety of effects. Through use of projection surfaces parameterized by arc-length we create an explicit relationship between the easily visualized surface and the projection behavior in the resulting image. This allows for creation of a wide variety of panoramas in a visual manner and draws upon existing modeling tools and conventions.

Additionally we have described how these projections surfaces can be rendered as well as described several possible applications. Lastly we demonstrated the use of these projections surfaces for changing existing panoramas into different panoramas.

8.1 Future Work

One area of future work is to expand beyond single center of projection panoramas. While creating such panoramas is straightforward, research is required in providing application for such projections, mechanisms for controlling their image characteristics that ensures construction of visually meaningful images.

Another issue is that we have limited ourselves to rectangular images. While alternatively shaped images can easily be achieved through 2D transformations it could prove useful to bind image area to arc-length (or another measure) of the projection surface.

Acknowledgments

We thank the reviewers for many helpful comments and suggestions. We also thank Luke Olsen and Adam Runions for their assistance preparing this work. The support of the Natural Sciences and Engineering Research Council of Canada and University Technologies International is greatly appreciated.

References

[1] B. Comment, *The Panorama*. Reaktion Books, 1999.

[2] J. Brosz, F. Samavati, S. Carpendale, and M. C. Sousa, "Single camera flexible projection," in *Proceedings of the 5th international symposium on non-photorealistic animation and rendering*. ACM, 2007, pp. 33–42.

[3] Y. Yang, J. X. Chen, and M. Beheshti, "Nonlinear perspective projections and magic lenses: 3d view deformation," *IEEE Computer Graphics and Applications*, vol. 25, no. 1, pp. 76–84, 2005.

[4] M. S. T. Carpendale and C. Montagnese, "A framework for unifying presentation space," in *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*. ACM Press, 2001, pp. 61–70.

[5] M. Inakage, "Non-linear perspective projections," in *Modeling in Computer Graphics (Proceedings of the IFIP WG 5.10)*, 1991.

[6] A. Glassner, "Cubism and cameras: Free-form optics for computer graphics," Microsoft, Tech. Rep. MSR-TR-2000-05, January 2000.

[7] N. Sudarsanam, C. Grimm, and K. Singh, "Non-linear perspective widgets for creating multiple-view images," in *NPAR '08: Proceedings of the 6th international symposium on Non-photorealistic animation and rendering*. ACM, 2008, pp. 69–77.

[8] D. Salomon, *Transformations and Projections in Computer Graphics*. Springer-Verlag, 2006.

[9] N. Greene, "Environment mapping and other applications of world projections," *IEEE Computer Graphics and Applications*, vol. 6, no. 11, pp. 21–29, 1986.

[10] G. Glaeser and E. Gröller, "Fast generation of curved perspectives for ultra-wide-angle lenses in vr applications," *The Visual Computer*, vol. 15, no. 7-8, pp. 365–376, November 1999.

[11] J. A. Polack, L. A. Piegl, and M. L. Carter, "Perception of images using cylindrical mapping," *The Visual Computer*, vol. 13, no. 4, pp. 155–167, June 1997.

[12] R. Szeliski and H.-Y. Shum, "Creating full view panoramic image mosaics and environment maps," in *SIGGRAPH '97*. ACM Press, 1997, pp. 251–258.

[13] P. Rademacher, "View-dependent geometry," in *SIGGRAPH '99*. ACM Press, 1999.

[14] A. Román, G. Garg, and M. Levoy, "Interactive design of multi-perspective images for visualizing urban landscapes," in *VIS '04: Proceedings of the conference on Visualization '04*. IEEE Computer Society, 2004, pp. 537–544.

[15] A. Agarwala, M. Agrawala, M. Cohen, D. Salesin, and R. Szeliski, "Photographing long scenes with multi-viewpoint panoramas," in *SIGGRAPH '06*. ACM Press, 2006, pp. 853–861.

[16] D. N. Wood, A. Finkelstein, J. F. Hughes, C. E. Thayer, and D. H. Salesin, "Multiperspective panoramas for cel animation," in *SIGGRAPH '97*. ACM Press, 1997, pp. 243–250.

[17] J. Yu and L. McMillan, "A framework for multiperspective rendering," in *15th Eurographics Symposium on Rendering (EGSR04)*, 2004, pp. 61–68.

[18] P. Degener and R. Klein, "A variational approach for automatic generation of panoramic maps," *ACM Trans. Graph.*, vol. 28, no. 1, pp. 1–14, 2009.

[19] D. Zorin and A. H. Barr, "Correction of geometric perceptual distortions in pictures," in *SIGGRAPH '95*. ACM Press, 1995, pp. 257–264.

[20] L. Zelnik-Manor, G. Peters, and P. Perona, "Squaring the circle in panoramas," in *IEEE International Conference on Computer Vision (ICCV)*, vol. 2, 2005, pp. 1292–1299.

[21] R. Carroll, M. Agrawal, and A. Agarwala, "Optimizing content-preserving projections for wide-angle images," in *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*. ACM, 2009, pp. 1–9.

[22] J. P. Snyder, *Flattening the Earth*, 2nd ed. University of Chicago Press, 1997.

[23] M. Trapp and J. Döllner, "A generalization approach for 3d viewing deformations of single-center projections," in *International Conference on Computer Graphics Theory and Applications (GRAPP)*, 2008, pp. 162–170.

[24] G. Farin, *Curves and Surfaces for CAGD: A Practical Guide*, 5th ed. Morgan Kauffman, 2001.

[25] J. J. Cherlin, F. Samavati, M. C. Sousa, and J. A. Jorge, "Sketch-based modeling with few strokes," in *SCCG '05: Proceedings of the 21st spring conference on Computer graphics*. ACM, 2005, pp. 137–145.

[26] T. Akenine-Möller, E. Haines, and N. Hoffman, *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., 2008.

[27] J.-D. Gascuel, N. Holzschuch, G. Fournier, and B. Peroche, "Fast non-linear projection using graphics hardware," in *ACM Symposium on Interactive 3D Graphics and Games*. ACM, Feb 2008.