Interactive Design of Multi-Perspective Images for Visualizing Urban Landscapes

Augusto Román Gaurav Garg Marc Levoy
Computer Graphics Laboratory, Stanford University, CA*

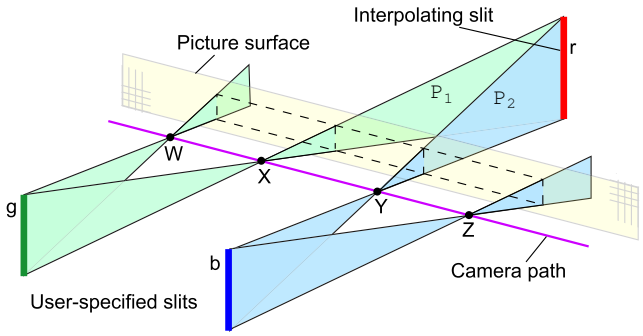


Figure 1: We visualize urban landscapes using a blend of adjacent cross-slits images. The figure shows two user-specified cross-slits cameras, represented by slit pairs WX-*g* and YZ-*b*. This partitions the camera path WXYZ into three sections. The plane P_1, formed by the slit *g* and point X, represents the rightmost column of pixels in cross-slits camera WX-*g* and their associated ray directions. Similarly, P_2 is the plane formed by slit *b* and point Y. These two planes P_1 and P_2 intersect in line *r*, which becomes our interpolating slit. The XY-*r* cross-slits pair becomes our interpolating camera. Note that the interpolating camera has the same ray directions on its edges as its neighboring cameras. This ensures that the generated image contains no discontinuities.

Abstract

Multi-perspective images are a useful way to visualize extended, roughly planar scenes such as landscapes or city blocks. However, constructing effective multi-perspective images is something of an art. In this paper, we describe an interactive system for creating multi-perspective images composed of serially blended cross-slits images. Beginning with a sideways-looking video of the scene as might be captured from a moving vehicle, we allow the user to interactively specify a set of cross-slits cameras, possibly with gaps between them. In each camera, one of the slits is defined to be the camera path, which is typically horizontal, and the user is left to choose the second slit, which is typically vertical. The system then generates intermediate views between these cameras using a novel interpolation scheme, thereby producing a multi-perspective image with no seams. The user can also choose the picture surface in space onto which viewing rays are projected, thereby establishing a parameterization for the image. We show how the choice of this surface can be used to create interesting visual effects. We demonstrate our system by constructing multi-perspective images that summarize city blocks, including corners, blocks with deep plazas and other challenging urban situations.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Viewing Algorithms; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques; I.3.8 [Computer Graphics]: Applications; I.4.9 [Image Processing and Computer Vision]: Applications

Keywords: cross-slits image, multi-perspective image, city block

*e-mail:aroman|ggaurav|levoy@graphics.stanford.edu

IEEE Visualization 2004
October 10-15, Austin, Texas, USA
0-7803-8788-0/04/$20.00 ©2004 IEEE

1 Introduction

Visualization of cities and urban landscapes has been a theme in western art since biblical times. The key problem in making these visualizations successful is summarizing in a single image the extended linear architectural fabric seen at eye level along a possibly curving or turning street, and doing so without introducing excessive distortions. In this paper we address this problem. Possible applications include using these visualizations for in-car navigation, an augmentation to online route mapping applications, and web-based tourism information.

One possible approach to depicting the eye level urban fabric is using wide angle or omnidirectional views around a single viewpoint, typically captured at street corners. Omnidirectional cameras [Nayar 1997] provide a possible optical solution for capturing such views. Photo-mosaicing (the alignment and blending of multiple overlapping photographs) is an alternative approach for creating wide field of view images. These mosaics can be made by capturing a part of the scene surrounding a single point by panning a camera around its optical center [Chen 1995; Shum and Szeliski 2000]. However, such omnidirectional views are still perspective projections and therefore, objects at any considerable distance from the camera become too small to be recognizable. A set of such views, each taken at more closely spaced intervals along the street, overcomes this limitation. However, such a set still fails to capture the linear nature of most urban fabrics and its continuity as experienced by a motorist or a pedestrian.

Another possible approach is to use pushbroom [Hartley and Gupta 1994; Peleg et al. 2000] or cross-slits imaging [Zomet et al. 2003]. A pushbroom image is defined as an image that is perspective in one direction (e.g., vertically) and orthographic in the other while a cross-slits image is an image which is perspective in one direction but is perspective from a different location in the other direction. The perspective structure of cross-slits cameras are the set of all rays intersecting two fixed lines (slits) in space. For pushbroom cameras, one of the slits is at infinity. In both cases, one is free to select the placement of the slits. Changing these placements strongly affects the visualization and the associated distortions as we show later in our results. In the context of visualizing eye level urban landscapes, we show that we can combine multiple cross-slits images seamlessly to reduce distortions.

Our main contribution is that we describe an interactive system

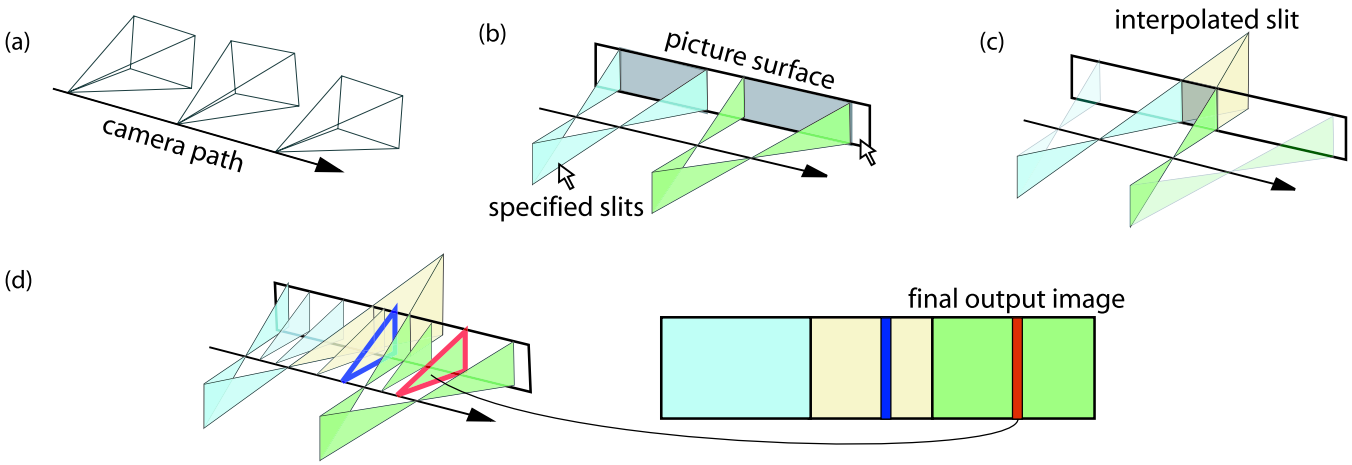


Figure 2: This figure summarizes our algorithm for generating multi-perspective images. (a) First, we process each input video frame to estimate the corresponding position and orientation of the camera. (b) Second, the user specifies the picture surface and any number of cross-slits camera locations (the green and blue regions), thereby defining valid region (gray shading) on the picture surface. (c) For the remaining regions (gray shaded), we automatically compute the interpolating cross-slits camera (yellow). (d) Within each camera, each planar fan of rays (blue or green triangles) denote one line of pixels (typically vertical) in the final output image. To produce this image, these pixels must be extracted from the appropriate frame of video, as described in section 4.4.

for constructing multi-perspective images from sideways-looking video captured from a moving vehicle. The input to our system is a set of video frames with known camera pose. The interface then provides a set of tools that allow the user to define the picture surface and place cross-slits cameras. Our system then automatically computes an additional cross-slits camera between every pair of adjacent user-specified cameras leading to a smooth interpolation of viewpoint in the final multi-perspective image. Our system provides the tools necessary to minimize distortions and discontinuities for creating good multi-perspective images for urban landscapes. Using our system, a person can create a multi-perspective image of a whole city block in a few minutes. The process is summarized in figure 2 and also demonstrated in the accompanying video.

Section 2 describes related work on multi-perspective imaging, while section 3 describes the multi-perspective representation that we use. Section 4 covers our interactive multi-perspective image design tool. We describe the input data, the design choices that we made, the user interface of our interactive system and the image rendering engine. We present our results in section 5 and present our conclusions and future work in section 6.

2 Multi-Perspective Imaging

Multi-perspective images are nothing new in the art world. 10th century Chinese paintings used multiple perspectives to depict many religious sites in a single image without noticeable distortions. More recently, the work of the cubists and M. C. Escher explored combining multiple perspectives.

More recently there has been an interest in computer generated multi-perspective imaging. The synthesis of multi-perspective images has been explored in Wood et al. [1997] and Rademacher and Bishop [1998]. Their methods construct an image from multiple viewpoints mapped onto a 3D model as a texture to generate new images from a single viewpoint. Glassner [2000] explores the use of multi-perspective imaging as an effective tool for illustration or story telling. He includes a plugin for a 3D modeling program that allows creating a multi-perspective image through a two-surface parameterization. We find it more convenient to use cross-slits images as fundamental modeling primitives (and ordinary perspective images as a special case of cross-slits images) for multi-perspective images. We have also found it important to be able to specify the parameterization of the picture surface. Vallance and Calder [2001] provide an in-depth analysis of the previous literature in multi-perspective imaging. They also describe an API to facilitate rendering of multi-perspective images.

Most of the papers described so far use synthetic 3D data for illustrating their results. Zheng [2003] generates route panoramas from a moving video by taking the central column of pixels from each frame and abutting them together. We will demonstrate how non-central columns can be used for to create more effective visualizations.

Seitz and Kim [2003] investigate how to generate multi-perspective images from a moving video camera. They treat the captured video as a stack of frames forming a 3D volume and then allow arbitrary 2D slices through this volume. While this method allows generation of almost any multi-perspective image that is possible given the video volume, it is not clear what perspectives the resulting images represent except in special cases. For example, a slice through the volume parallel to the first frame (essentially extracting a frame from the volume) is a perspective image, a slice straight down the volume is a pushbroom image, and a diagonal slice is a cross-slits image. A non-linear slice through the volume will create a multi-perspective image such as we create in this paper. However, it is difficult to associate any general non-linear slice with its perspective structure in 3D. This in turn makes it hard to design a slice to accomplish a particular task, such as displaying city blocks with their varying facade depths. It is precisely this problem that we address in this paper.

In addition to making multi-perspective imaging practical, there has also been much theoretical work on multi-perspective imaging. Gupta and Hartley [1997] derive a projection model for pushbroom cameras. Zomet et al. [2003] extend this in their work to model cross-slits cameras. More recently, Yu and McMillan [2004] provide a General Linear Camera (GLC) model, which unifies the perspective, pushbroom and cross-slits cameras along with five other camera models under one framework. Although GLC's encompass eight cameras, we currently restrict our system to these three, which seem most useful for our task. Specifically, the subset of GLCs we allow is that which can be created from a camera traveling in a path,

Figure 3: A hand-crafted multi-perspective image excerpted from Michael Koller's Seamless City. Notice the distinct vanishing points at A and C, even though these two streets are parallel, and the distortions in the depiction of building B. This image was constructed by aligning and inter-cutting several ordinary perspective images. Artful placement of the cuts between images yields a composite image without evident seams. However, this process undoubtedly requires great care and labor. Our paper introduces a novel user interface for semi-automatically constructing images like this one (used with permission).

since the camera path naturally defines one slit of a cross-slits camera.

The work that most closely resembles our own is that of an artist Michael Koller [2004]. An excerpt from his work is shown in figure 3. Koller synthesizes a continuous visual image of the city of San Francisco made by sequential photos of a walk through the city. In a manual process he aligns, cuts and pastes these images next to each other to form a final image. By making his cuts follow architectural features in the scene, he produces perspectives that look correct along the alleys and street intersections. Our work can be thought of as a way to automate his approach, however our approach is not as flexible as his. Unlike him, we only allow vertical cuts in input images.

3 Multi-Perspective Representation

3.1 Multi-Perspective Paradigm

Multi-perspective images can be specified as a 2D manifold of rays and the mapping from this manifold to a rectangular, regularly sampled image as described in figure 5. The user specifies the manifold of rays as a sequence of cross-slits cameras in the 3D scene, and specifies the mapping to an image by placing a regularly sampled picture surface in the scene. We find that distinguishing clearly in our user interface between specifying the ray manifold and specifying the mapping to the output image improves the intuitiveness of our system.

The manifold of rays can be specified in several ways. For a pinhole camera, the manifold is the set of all rays passing through a point. In an orthographic camera, the manifold describes all rays in a single direction. Similarly, for a cross-slits camera the manifold is described by all rays passing through two lines (or slits). These are three of the eight 2D linear manifolds described by Yu and McMillan [2004].

For our application, we choose to constrain the allowable set of manifolds based on their applicability to urban landscapes and the ease of specification for the user. Specifically, we restrict the space of ray manifolds to cross-slits images. Doing so allows us to include perspective and pushbroom images while at the same time enabling the interpolation scheme described in section 3.2. The result is that our final image can be represented as a mosaic of multiple cross-slits images.

The picture surface defines the sampling of the manifold of rays as well as their mapping to the final image, as shown in figure 5. Traditionally, in a single perspective image, the picture surface is a plane. However, we can reduce distortion in the output image by allowing the picture surface to change orientation to accommodate the storefronts.

Lastly, every point on the picture surface must be associated with a ray from exactly one ray manifold. This ensures that the resulting final output image has no missing regions.

3.2 Interpolating Two Cross-Slits Images

Associating regions of the picture surface with cross-slits cameras naturally leads to the problem of how to handle unassigned regions between them. Our system adds an additional cross-slits camera between every two adjacent user-specified cross-slits cameras, resulting in smooth interpolation. The process of computing the location of the interpolating slit is described in figure 1. As shown in the figure, the location of the interpolating slit is uniquely defined by the geometry of the adjacent slits.

It should be clear from the figure that the insertion of these interpolating slits is independent of the picture surface chosen by the user. Also, the camera path must be continuous but need not be straight—the interpolating slit will depend only on the points X and Y along the camera path. Note that this interpolation scheme is not dependent on using the camera path as one of the slits. In fact, this scheme will correctly interpolate any pair of cross-slits cameras provided that the slit joining them is continuous.

4 Interactive Multi-Perspective Image Design Tool

In this section, we describe our system for interactive design and rendering of multi-perspective images. Our system takes a set of video frames (captured with a sideways-looking video camera) with known camera pose as input and produces one composite multi-perspective image as output. The system consists of a user interface and a rendering engine.

4.1 Data Acquisition and Preprocessing

In our current implementation, data is captured using a Basler A504kc high-speed camera which captures at 300 frames per second. The high speed ensures dense image data while still maintaining a reasonable driving speed (approximately 8-10 mph) for the videos of commercial city blocks. Camera pose is then estimated with a structure-from-motion (SFM) algorithm using Boujou [http://www.2d3.com], a commercial software package. The SFM also outputs partial 3D scene structure, which helps the user in choosing cross-slits locations. We assume that the X-Z plane that camera pose is estimated in corresponds to the real-world ground plane.

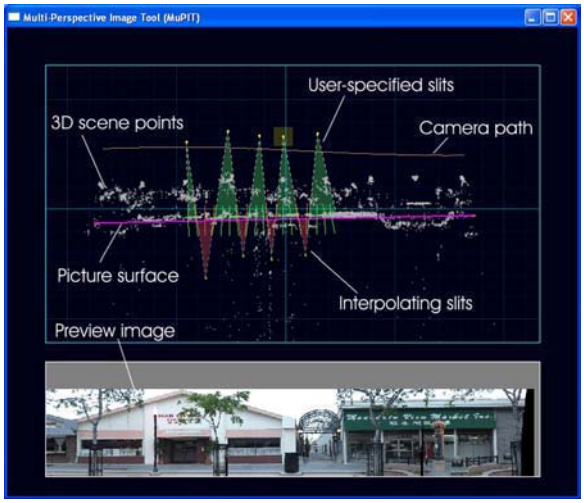


Figure 4: This is a snapshot of our interface. The diagram at top is a plan view of the scene. You can see the partial 3D scene structure, picture surface, camera path, user-specified slits, interpolating slits and a low-resolution preview image.

4.2 Design Choices

There are a number of restrictions that we place on our system. We assume that the camera path in the input video lies on a plane parallel to the ground plane. For each cross-slits image, the user specifies one slit, and the camera path is assumed to be the second slit. We assume that the user-specified slit is vertical (perpendicular to the ground plane). This allows a simplified plan view to be used in our user interface where the vertical slits are projected as points.

As shown in figure 5, the picture surface is a parametric surface in 3D. We choose our picture surface to have vertical sides. The restriction of vertical sides on the picture surface applies naturally to urban facades on flat terrain. To aid the user in specifying non-vertical sides of the picture surface, we provide piecewise-continuous lines and quadratic splines. Quadratic splines are approximated as piecewise-linear segments to simplify rendering. This allows our surface to be represented as a series of planar facets. We constrain the sampling of the picture surface to be regular.

4.3 User Interface

Shown in figure 4, our interface provides both a design section in which the user specifies the multi-perspective image and a preview section that can provide rapid, low-resolution previews of the final image. Once satisfied with the design, the system can output the full resolution image.

To create a multi-perspective image, the user must define the picture surface, place all desired cross-slits cameras, and associate the cameras with regions of the picture surface. Because of the lack of natural intuition concerning these types of images, the interface strives to present the design in terms of familiar concepts. With this in mind, the user is shown the camera trajectory in plan view along with any estimated scene structure in the form of a point cloud as output by our SFM algorithm. The camera trajectory, as explained in section 4.2, defines one of the two slits required for any cross-slits image.

To define the picture surface in our interface, the user needs to only draw a set of connected line segments in plan view. This is possible because we restrict the picture surface to be vertical. To help fit the picture surface to curved facades, segments of the picture surface can also be toggled between straight lines and quadratic splines.

In plan view, the task of positioning user-specified slits involves simply placing the slits as points and specifying their field of view. The intersection of the field of view with the picture surface defines the region of the picture surface associated with that slit. If any segment of the picture surface is associated with more than one user-specified slit, such as if two fields of view overlap, there no longer exists a unique ray direction for points in that segment, and therefore that is not a valid specification for a multi-perspective image. As long as the fields of view do not overlap on the picture surface, however, any number of user slits may be described. The specified camera slits can also be toggled between slits located at finite positions and slits located at infinity. Slits at infinity are represented by a directional line next to the selected point. Placing a slit at infinity produces a pushbroom image. Similarly, placing the slit directly on the camera path (thus intersecting both slits) produces an ordinary perspective image.

Once any valid multi-perspective image is specified, the interface immediately shows a set of example ray directions at several points along the picture surface. The program also automatically displays the interpolating cross-slits camera between any two adjacent user-specified cameras as explained in section 3.2.

4.4 Rendering Engine

Using an Intel 2.8 GHz Pentium4 machine, the low-resolution preview image can typically be rendered in under a second. An efficient rendering engine is necessary to provide this fast feedback to the user.

The constraints imposed by our design choices allow us to simplify and accelerate the rendering. Since each point on the picture surface is associated with only one cross-slits camera, the final output image is composed from several distinct, abutting cross-slits images. Each cross-slits camera can span multiple facets of the picture surface. Also, each facet may contain multiple cross-slits cameras. In both cases, we render only a single cross-slits image onto a single planar segment at a time. Each planar segment is parameterized by (u,v). Finally, the restriction of vertical sides for the picture surface and of vertical user-specified slits allows us to assign entire columns of the final output image from a single input video frame. For each cross-slits image on a planar segment, there is a mapping between the (u,v) coordinates of the planar segment to the (s,t) coordinates of the final output image, defining the sampling of the picture surface. This can be seen in figure 5. The sampling on each planar segment is regular in the u and v parameter directions.

A fast algorithm to render a single cross-slits image onto a single planar surface is therefore the basic building block used in rendering. For each cross-slits image, we compute the mapping between each input frame and the final output image. We then use this mapping to compute which pixels will actually contribute to the final output image and warp only those pixels.

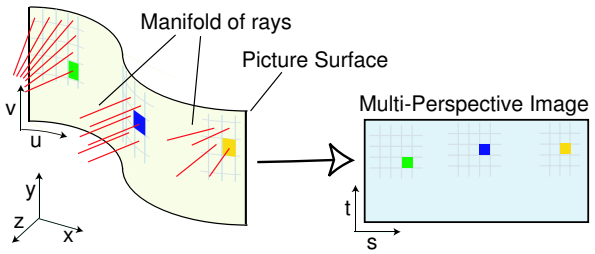


Figure 5: The ray manifold is the set of rays in space. The sampling of these rays and the mapping to the output image can be specified by a picture surface with a 2D parameterization of its 3D surface.

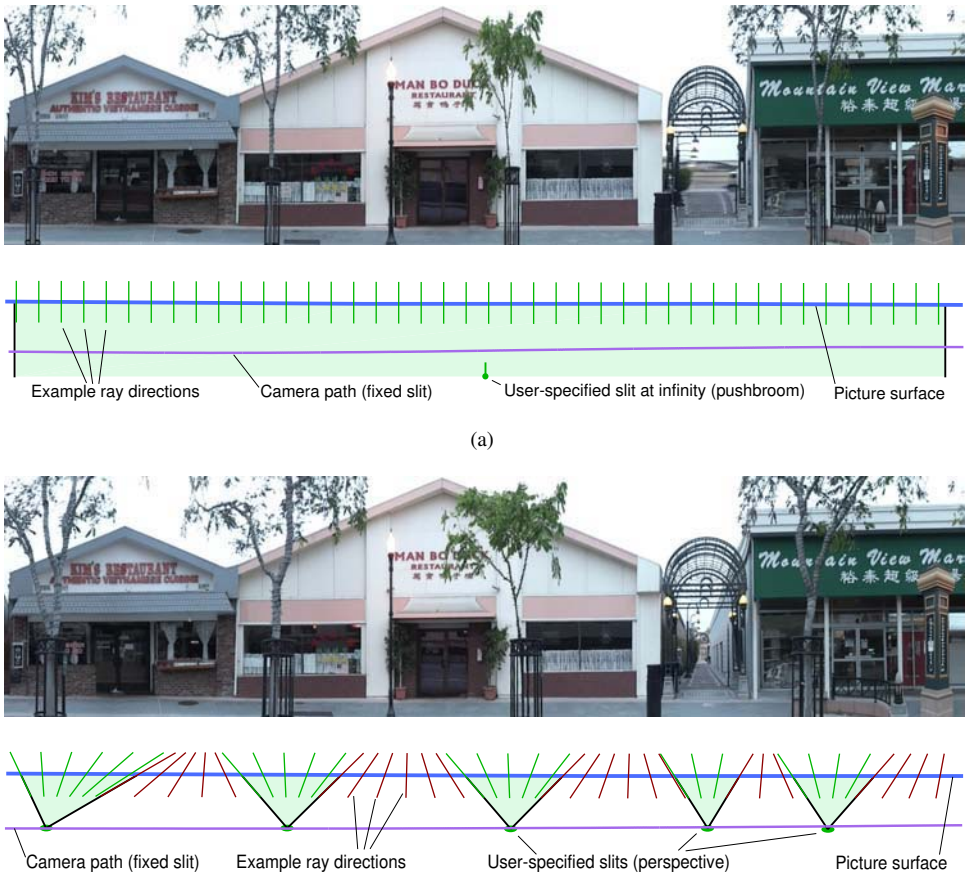


Figure 6: This example shows how manipulation of the perspective structure of the image can be used to generate a multi-perspective image with reduced distortion. The diagram below each is a plan view of the scene, with the input video camera moving along the indicated path and looking upwards. The picture surface in both (a) and (b) is fixed at the facade of the storefronts. (a) is a traditional pushbroom image generated by specifying a vertical slit at infinity with all the ray directions being parallel. The resulting image has the familiar distortions associated with a pushbroom image: objects in front of the picture surface (e.g. trees) are compressed horizontally, and objects behind the picture surface are expanded (the view down the alleyway). (b) has been generated using multiple cross-slits. By placing selected user-specified slits atop the camera path, ordinary perspective views are generated in the vicinity of the trees and alleyway. This enhances the realism of the image while still maintaining the extended field of view of the pushbroom.

The main point is that we do not pre-warp the input images. Instead, we calculate the necessary warp and warp only the small region from each input image. This speeds up rendering to the point that if all images can be preloaded into RAM, previews can be rendered at interactive rates even for thousands of input images.

We now describe the process of computing the homography from an input video frame to the output image for a single planar segment. We use the convention of using bold for vectors (lowercase) and matrices (uppercase). Points in different coordinates systems are

$\mathbf{x} = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T$	3D point in world coordinate system in which camera pose is estimated
$\mathbf{u} = \begin{bmatrix} u & v & 1 \end{bmatrix}^T$	2D point in the coordinate system of the planar segment of the picture surface
$\mathbf{p} = \begin{bmatrix} p & q & 1 \end{bmatrix}^T$	2D pixel location of a single input video frame
$\mathbf{s} = \begin{bmatrix} s & t & 1 \end{bmatrix}^T$	2D pixel location of the final multi-perspective image

Let the origin of the planar segment in world 3D homogeneous coordinates be the 4x1 vector $\mathbf{o}$. Let $\mathbf{w}$ and $\mathbf{h}$ be 4x1 vectors defining the width and height extent of the segment. Then, the mapping from a point $\mathbf{u}$ (for all $0 \le u, v \le 1$) on the planar segment of the picture surface to a corresponding world point $\mathbf{x}$ is given by

$$\begin{aligned} \mathbf{x} &= u\mathbf{w} + v\mathbf{h} + \mathbf{o} \\ &= \begin{bmatrix} \mathbf{w} & \mathbf{h} & \mathbf{o} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \\ &= \mathbf{Q}\,\mathbf{u} \end{aligned}$$

In general, a point in the world coordinate system $\mathbf{x}$ is mapped into a particular input video frame i by the relationship:

$$\mathbf{p} = \mathbf{K}\,[\mathbf{R}_i \,|\, \mathbf{t}_i]\,\mathbf{x}$$

given the camera intrinsics $\mathbf{K}$, the rotation matrix $\mathbf{R}_i$, and the translation vector $\mathbf{t}_i$. Substituting for $\mathbf{x}$, we obtain the relationship between the planar segment and the input video frame:

$$\mathbf{p} = \mathbf{K}\,[\mathbf{R}_i \,|\, \mathbf{t}_i]\,\mathbf{Q}\,\mathbf{u}$$

With our restriction of regular sampling on the planar facets, we map the segment to a rectangular region of the final output image using only translation and scaling. Note that this choice of mapping is arbitrary. We can now define the relationship between this planar segment of the picture surface and the final output image as:

$$\mathbf{s} = \begin{bmatrix} a & 0 & c \\ 0 & b & d \\ 0 & 0 & 1 \end{bmatrix} \mathbf{u} = \mathbf{M}\,\mathbf{u}$$

or equivalently

$$\mathbf{u} = \mathbf{M}^{-1}\,\mathbf{s}$$

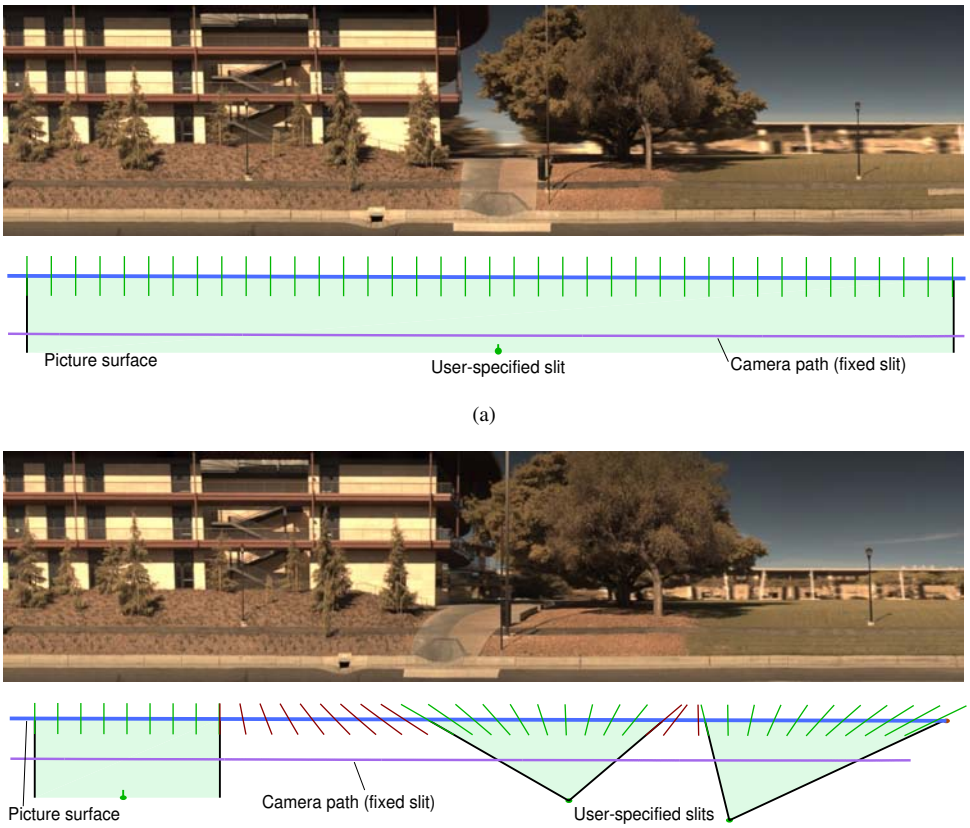


Figure 7: This example shows how our system can be used to generate effective multi-perspective images of deep plazas. The picture surface in both (a) and (b) is fixed at the facade of the building on the left. (a) is a traditional pushbroom image generated by specifying a vertical slit at infinity. This causes the deep plaza on the right to stretch horizontally, leading to apparent smearing due to limited resolution. In (b), by blending between two almost perspective views on the right (one for the tree and one for the building) and the same pushbroom view on the left results in a better visualization. In our multi-perspective image, some unwanted curving is introduced into the walkway at the center of the image. This walkway is in fact straight.

where c and d define the origin of the rectangular region on the final output image and a and b define the width and height of the region respectively.

We can then obtain a direct relationship between each input video frame i and the final multi-perspective image as

$$\mathbf{p} = \mathbf{K}[\mathbf{R}_i|\mathbf{t}_i]\mathbf{Q}\mathbf{M}^{-1}\mathbf{s}$$

Using this formulation, all of the intermediate matrices reduce to a single invertible 3x3 matrix that we call $\mathbf{H}$, giving

$$\mathbf{p} = \mathbf{H}\mathbf{s}$$

$$\mathbf{s} = \mathbf{H}^{-1}\mathbf{p}$$

The matrix $\mathbf{H}$ defines the homography between an input video frame and a portion of the final multi-perspective image corresponding to one planar segment of the picture surface.

With the mapping from each input video frame to the final output image, each pixel on the final output image will have multiple input frames overlapped onto it. We must now choose which input frame to select for each pixel. For our cross-slits images, one of the slits is fixed—the camera path. The second, vertical slit is either a user-specified or interpolated slit. We project a point on this vertical slit through each input video frame onto the final output image.

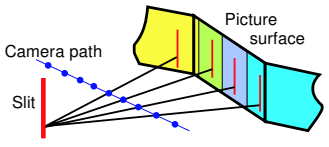


Figure 8: The slit is projected through each camera onto the picture surface. The region closest to each projection is assigned to the corresponding camera.

To compute this, we take the 3D point on the second slit that intersects the ground plane, $\mathbf{x}_g$, project it onto the input video frame ($\mathbf{p}_g$), and then use the homography derived above to project the point onto the final output image ($\mathbf{s}_g$). For a particular input video frame i, the image of $\mathbf{x}_g$ through that camera's center of projection maps to

$$\mathbf{p}_g = \mathbf{K}[\mathbf{R}_i|\mathbf{t}_i]\mathbf{x}_g$$

$$\mathbf{s}_g = \mathbf{H}^{-1}\mathbf{p}_g$$

As shown in figure 8, for each column of the final output image, we then select the input frame with the closest projected point $\mathbf{s}_g$ and warp the pixels from that frame that correspond to the output image column.

5 Results

We tested our system on several videos of city blocks taken under different challenging scenarios. We first examine a scene with a relatively flat facade and straight camera path. Figure 6(a) shows a short section of a pushbroom representation of this scene. A pushbroom image is perspective in one direction (vertical in our case) and orthographic in the other direction (horizontal). Under this projection, only objects lying on the picture surface can be rendered faithfully. In our example, the picture surface is placed at the store facade. As expected with such a parameterization, trees (which are closer than the facade) are horizontally compressed while the view down the alleyway (which is farther) is expanded. By interactively manipulating the perspective structure of the image, we can reduce these distortions as shown in figure 6(b). Using our user interface, we achieve this by specifying ordinary perspective cameras near

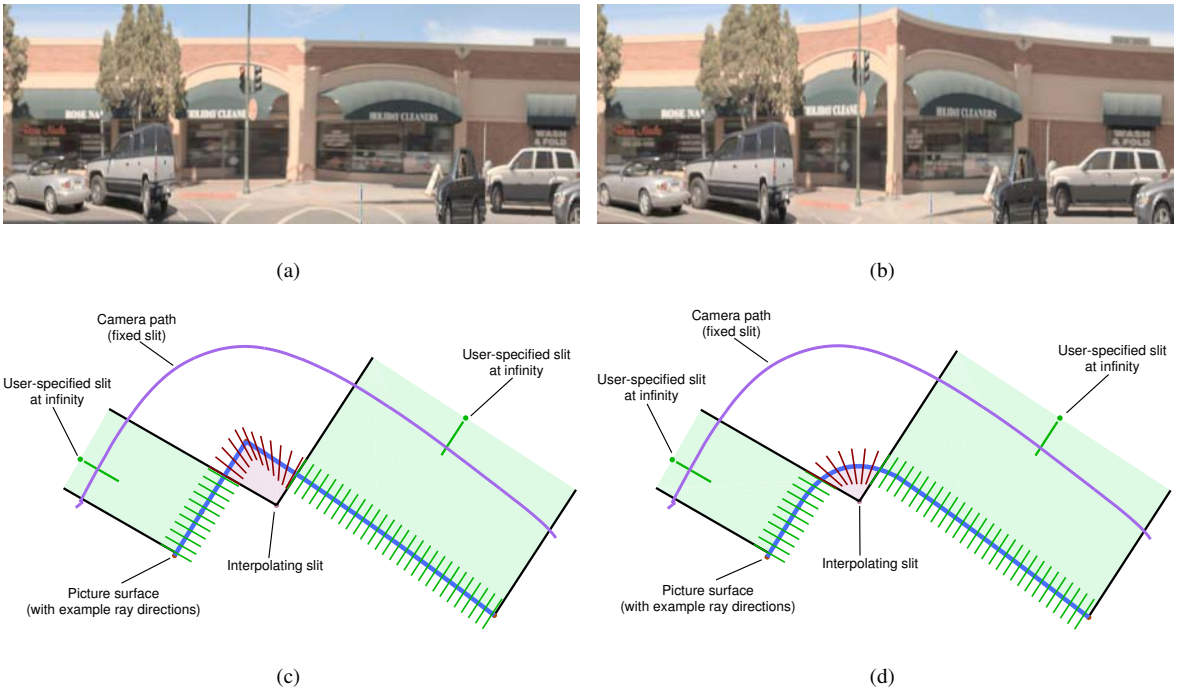


Figure 9: Our visualization of a street corner with perpendicular storefronts. (a) shows the multi-perspective image generated by our system for the choice of picture surface shown in (c). (b) shows the multi-perspective image generated by our system for the choice of picture surface shown in (d). The picture surface in (c) conforms to the actual storefront whereas it has been artificially curved in (d) using our interactive system. Note that the slits are at the same location in both the setups. (a) gives the impression that the storefront is continuous and there is no corner. By altering our picture surface we introduce an artificial pinching effect in the multi-perspective image shown in (b). This helps emphasize that it is a corner without causing severe distortions.

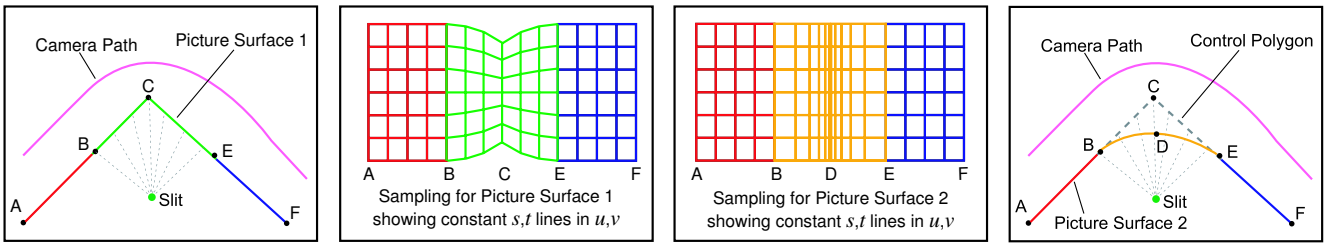


Figure 10: This figure explains how our choice of a curved surface and a non-uniform sampling strategy achieves a pinching effect at the corners seen in figure 9(b). In (a), picture surface 1 conforms to the storefront. To achieve the pinching effect in this case we would need to stretch the image at C. This would mean non-linearly mapping the picture surface uv coordinates to the final output image st coordinates as shown in (b) to maintain continuity in the image. We achieve the same effect by instead using a curved surface as shown in (d) but sampling according to the strategy in (c). This solves the problem of nonlinear vertical sampling, but now requires adjusting the horizontal sampling. This can be easily done by using the control polygon to define the horizontal mapping from 3D xyz to u but still mapping to s linearly. If the control polygon corresponds to the storefront, then a brick on the wall will not appear to be stretched horizontally.

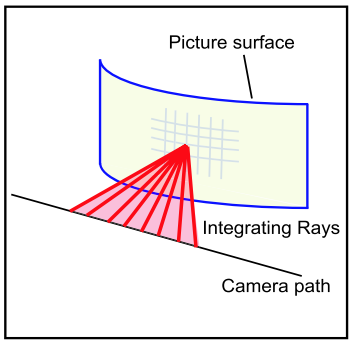
regions of significant depth variation such as the trees and the alleyway. This creates an image that is more natural looking in these areas. To keep the image continuous, the system inserts interpolating cameras between the user-specified cameras.

Another example that illustrates the benefit of manipulating ray directions based on scene geometry is shown in figure 7. The scene consists of a building with flat facade on the left and a very deep plaza on the right. By choosing multiple cross-slits as shown in figure 7(b) we can get a more recognizable image than figure 7(a).

The ability to specify curved picture surfaces allows us to conform the picture surface to the natural architecture of a corner. An example of this is shown in figure 9(a). This type of image is impossible to create with a traditional single, planar picture surface.

Although this allows us a summary view of both sides of the corner, the apparent size of facade is constant throughout the image. We can more naturally represent what a motorist or pedestrian would see if we stretch the image near the corner, producing a pinching effect as shown in figure 9(b). This is a nonlinear effect and would require sampling of the picture surface as shown in figure 10(b). Instead, we achieve this effect by curving the picture surface as shown in figure 9(d) and adjusting the horizontal sample density

(a) Perspective image



(b) Integration on picture surface

(c) Synthetic focus image

Figure 11: Here we show a preliminary extension of our algorithm to generate synthetically focused images. By integrating over an angular arc for each point on the picture surface as shown in (b), we can simulate a large 1D aperture focused at the picture surface. (c) is an example of such a synthetically focused image with the focus fixed at the plane of the facade. Note that the tree present in (a) is blurred out.

as shown in figure 10(c) by using the control polygon. This is a departure from the earlier imposed regular sampling on the picture surface. This sampling adjustment is performed automatically by our program and is explained in figure 10.

6 Conclusions and Future Work

While most people generally have a good idea of what a photograph will look like when shown a diagram of the camera position and orientation relative to a scene, this intuition does not exist for multi-perspective images. The choice of 2D manifold of rays, the placement of the picture surface and the sampling of the surface constitute a design problem. We provide a user interface which helps develop an intuition for the perspective structure of multi-perspective images as well as generates effective visualizations of urban landscapes.

One limitation of our system is that we allow only regular sampling on the picture surface (and the automatic adjustment of horizontal sampling density illustrated in figure 10). We do not allow any user-specified sampling strategies. One can imagine sampling the image more densely in the center than toward the edges, resulting in a fish eye like effect. Also, the enforcement of vertical slit orientation, a limitation imposed by our user interface, implies that the user interface cannot accurately depict changes in elevation such as hills. Similarly, the user interface does not permit altering the orientation of the picture plane from vertical. Finally, we choose to allow representing only three of the eight GLCs from Yu and McMillan [2004]. It would be interesting to incorporate these other cameras into our design tool.

Regarding future work, all of our examples have an effectively infinite depth of field (assuming pinhole cameras for input). By averaging multiple input video frames projected onto the picture surface, we can simulate a synthetic aperture as discussed in Vaish et al. [2004]. This results in an extremely shallow depth of field image. Vaish et al. [2004] show this effect for planar picture surfaces. By specifying a curved picture surface, we extend this effect to non-planar picture surfaces. However, in our case the aperture is only 1D, whereas the apertures in [Vaish et al. 2004] are 2D. Figure 11 shows a preliminary result for a planar picture surface.

Finally, one can imagine a system that automatically places the cross-slits based on the 3D structure of the scene to obtain a result similar to that seen in figure 6(b). Further extensions could include relaxing the constraint of mosaicing the cross-slits images with only vertical cuts. Allowing arbitrary cuts can help achieve the quality found in Koller's work [Koller 2004]. Furthermore, one could define a minimum error cut similar to Efros and Freeman [2001].

Acknowledgments: We would like to thank Sarah Harriman, Neel Joshi, Dan Morris, Jeff Klingner, and Austen McDonald for proofreading various versions of the draft. This work was supported by grants from Google Inc., National Physical Science Consortium, the Alfred P. Sloan Foundation, and the Reed-Hodgson Stanford Graduate Fellowship.

References

CHEN, S. E. 1995. Quicktime VR: An image-based approach to virtual environment navigation. In *Proc. SIGGRAPH*, 29–38.

EFROS, A. A., AND FREEMAN, W. T. 2001. Image quilting for texture synthesis and transfer. In *Proc. SIGGRAPH*, 341–346.

GLASSNER, A. S. 2000. Cubism and cameras: Free-form optics for computer graphics. *MSR-TR-2000-05*.

GUPTA, R., AND HARTLEY, R. I. 1997. Linear pushbroom cameras. *IEEE PAMI 19*, 9, 963–975.

HARTLEY, R. I., AND GUPTA, R. 1994. Linear pushbroom cameras. In *Proc. ECCV*, 555–566.

KOLLER, M., 2004. Seamless city. http://www.seamlesscity.com.

NAYAR, S. K. 1997. Catadioptric omnidirectional camera. In *Proc. CVPR*, 482–488.

PELEG, S., ROUSSO, B., RAV-ACHA, A., AND ZOMET, A. 2000. Mosaicing on adaptive manifolds. *IEEE PAMI 22*, 10, 1144–1154.

RADEMACHER, P., AND BISHOP, G. 1998. Multiple-center-of-projection images. In *Proc. SIGGRAPH*, 199–206.

SEITZ, S. M., AND KIM, J. 2003. Multiperspective imaging. *IEEE Computer Graphics and Applications 23*, 6, 16–19.

SHUM, H.-Y., AND SZELISKI, R. 2000. Construction of panoramic image mosaics with global and local alignment. *IJCV 36*, 2, 101–130.

VAISH, V., WILBURN, B., AND LEVOY, M. 2004. Using plane + parallax for calibrating dense camera arrays. In *Proc. CVPR*, 2–9.

VALLANCE, S., AND CALDER, P. 2001. Multi-perspective images for visualisation. In *Proc. Visual Information Processing*, 69–76.

WOOD, D. N., FINKELSTEIN, A., HUGHES, J. F., THAYER, C. E., AND SALESIN, D. H. 1997. Multiperspective panoramas for cel animation. In *Proc. SIGGRAPH*, 243–250.

YU, J., AND MCMILLAN, L. 2004. General linear cameras. In *Proc. ECCV*, 14–27.

ZHENG, J. Y. 2003. Digital route panoramas. *IEEE MultiMedia 10*, 3, 57–67.

ZOMET, A., FELDMAN, D., PELEG, S., AND WEINSHALL, D. 2003. Mosaicing new views: The crossed-slits projection. *IEEE PAMI 25*, 6, 741–754.