Immersive Teleconferencing : A New Algorithm to Generate Seamless Panoramic Video Imagery

Aditi Majumder M.Gopi
W. Brent Seales Henry Fuchs

University of North Carolina at Chapel Hill
{majumder,gopi,seales,fuchs}@cs.unc.edu

Abstract

This paper presents a new algorithm for immersive teleconferencing, which addresses the problem of registering and blending multiple images together to create a single seamless panorama. In the immersive teleconference paradigm, one frame of the teleconference is a panorama that is constructed from a compound-image sensing device. These frames are rendered at the remote site on a projection surface that surrounds the user, creating an immersive feeling of presence and participation in the teleconference. Our algorithm efficiently creates panoramic frames for a teleconference session that are both geometrically registered and intensity blended. We demonstrate a prototype that is able to capture images from a compound-image sensor, register them into a seamless panoramic frame, and render those panoramic frames on a projection surface at 30 frames per second.

1 Introduction

A limiting factor most commonly associated with current teleconferencing technology is network bandwidth and its impact on image size and frame rate. Another, less commonly noted, constraint that limits the usefulness of teleconferencing systems is the field of view represented by the imagery. The field of view of a typical camera is small and of low resolution. When this is used as the basis for a teleconference, it fails to produce a convincing feeling of presence and immersion. Even when the network bandwidth problem is completely solved, one is left with relatively low spatial resolution and a narrow field of view. In this paper we present an algorithm that is designed for teleconferencing systems that attempt to be *immersive*, where the imagery is meant to be a piecewise collection of images that together form a panorama or wide-area image (Fig. 1). Without addressing the issue of excessive network bandwidth, we move forward from the narrow field of view system and open greater possibilities for sessions that engender a sense of presence between users.

The decision to capture, transmit and display a wide-area imagery comes with a price in several areas. In this paper we focus on the particular problem of making the wide-field-of-view imagery seamless, without introducing unacceptable visual artifacts, and without incurring a computational overhead that becomes impractical for real-time conferencing. Single-sensor, high-resolution wide-field-of-view solutions are very expensive and not commonly available. As a result most researchers have proposed that immersive imagery should be composited from compound imaging devices [13], built from a set of cameras, arranged with or without mirrors so that their collective field of view generates the desired panorama. These devices do not produce seamless imagery because their arrangement, even when carefully calibrated, is inexact and leads to visible artifacts at image boundaries.

This paper presents a new algorithm for building a seamless wide-area image from an underlying set of images, under two explicit assumptions that the sensors are positioned only approximately and the algorithm must run in real-time. The algorithm makes use of simple geometric constraints that apply when cameras are clustered together to have an approximate common virtual center of projection (COP).

The geometric registration algorithm presented here produces a texture map, which encodes a warp that is obtained in an initialization step. The warp is created in such a way as to register all the individual images from the compound imaging device into a single panorama. The texture map, which needs only to be computed once, is then delivered to the on-line, ren-

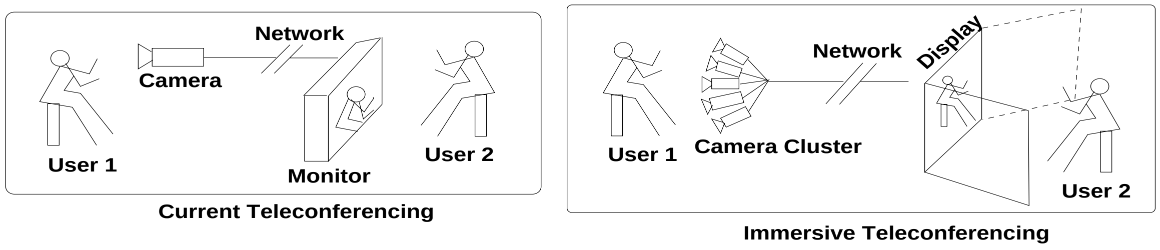

Figure 1: Difference between current and immersive teleconferencing

dering component of the system for the real-time display of imagery from the cameras onto a target display surface. The photometric algorithm that we have proposed here also has two components. The first step uses a multi-resolution spline technique [12] to generate a *pixel-based-weight map*. The second step is to encode the map for use on-line to achieve the photometric blending across the camera boundaries. By applying the off-line algorithm to obtain the warp and the pixel-based-weight-map, and then using the result in an on-line rendering algorithm, we are able to capture, register and render high-resolution panoramic imagery at 30 frames per second.

1.1 Capturing Panoramic Imagery

Advances in digital photography have made the creation of panoramic scenes quite commonplace [5]. Several sensors have also been developed which form a panoramic image on a single image plane through the use of continuous mirrored surfaces [6, 7, 8]. In these cases, no image registration or blending is required. The drawback of that approach is the limited resolution of a single sensor. Our approach requires merging multiple images, but the final image resolution is much higher.

Our approach is to allow a compound imaging device to be constructed from any "cluster" of cameras that approximates a system with a common COP. We do not require exactness, as it is the goal of the algorithm to compensate for misalignments and aberrations in positioning. The particular camera cluster used in this work as the input device is built from multiple cameras to simulate a virtual camera with 360 degree horizontal field of view and a 90 degree vertical field of view. Since an image from any such cluster will only appear correct if the physical cameras all share the same COP, we have taken care to approximately align the cameras' optical centers. Since multiple cameras cannot occupy the same physical space, mirrors are used to reflect the optical center of the camera to a different physical location. Two cameras are used for each 60 degree horizontal "slice" of the field of view: one camera for the lower tier of the cluster and another camera for the upper tier. The entire mechanical assembly is approximately 460 mm long, 450 mm high and 450 mm wide with 12 identical cameras (JVC Model TK-1270U) one for each mirror. Figure 2 shows a CAD design of the geometry next to a photograph of the implemented system.

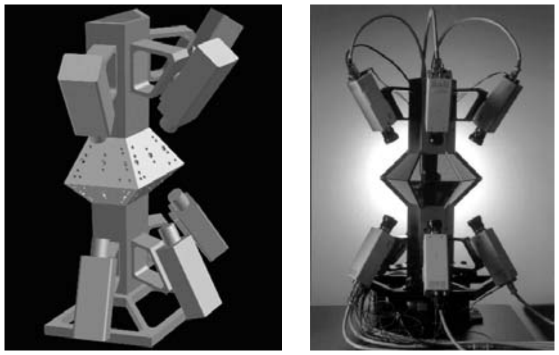

Figure 2: One side of the Camera Cluster

Although the particular camera cluster we used in this paper was manufactured to close tolerance by our collaborators at University of Utah, it illustrates the need for the algorithms we present. The geometric registration of the images is difficult because of the mechanical misalignments in the front surface mirror placements and differences between the internal parameters of the cameras. Hence the panoramic imagery cannot be generated by direct alignment of the images. Further the large variation in the image intensity and color balance between the cameras motivates our use of a feature based technique, which performs more robustly than a correlation based method for geometric registration.

1.2 Camera Cluster Images

The design of the camera cluster ensures that there is a small overlap region between the adjacent camera images, because of the non-zero aperture size of the cameras. In an ideal pin-hole camera model, there would be no overlap.

A typical image from a camera of the camera cluster is shown in the top left figure of color plate 1. Each camera sees a trapezoidal mirror from which the world space is reflected. The trapezoidal region of interest in each image, is geometrically and photometrically registered with its neighbors to construct a panoramic image of the world from a common COP.

1.3 Analysis on Camera Cluster Images

Consider the six image planes from the camera cluster on the top half of the cluster (Fig. 1). Let the COP be at the origin. The virtual image planes form a dihedral angle of approximately 127 degrees with its neighbors, the optical center being equidistant from the COP. In this configuration, each image plane will be facing the common COP and will be clipped at the sides by its neighbor's image planes to form a trapezoid. Figure 3 shows two virtual camera image planes in this configuration. These planes are extended beyond the image extent, for clarity. Within the image extent, the projection of one plane on the other is shown by a shaded region.

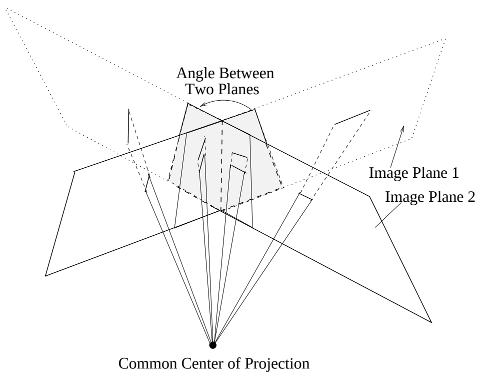

Figure 3: The Overlapping Region in 3D

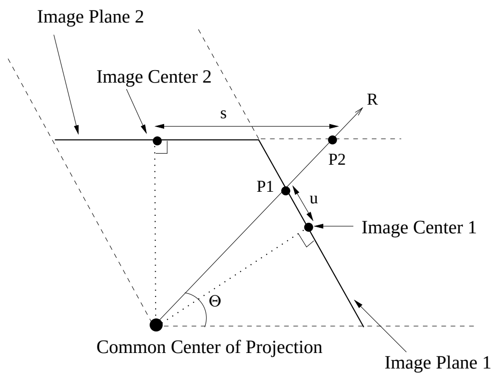

Figure 4: Analysis of the Overlapping Region

For simplicity, consider just two of the above six image planes. We are interested in the relationship between the coordinate systems (u, v) and (s, t) of these two images. The 2D equivalent of this configuration, illustrated in Fig. 4, is formed from the base edges of these two image planes at the equator. They form an angle of 120 degrees and subtend 60 degrees at the COP. In 2D we are interested in the relationship between u and s. This is easily extended to 3D involving v and t also. A ray, R, from the COP intersects the image planes (lines) at points, say, $P1$ and $P2$. In Figure 5, the distance of these projected points, $P1$ and $P2$, from the optical center of their respective image planes is plotted against the angle, θ, the optical ray makes with the horizontal line. These distances are directly related to the u and s coordinates of the image planes. At the region in the vicinity of the intersection of these two image lines(55 to 65 degrees), the change in both u and s is very small, and for a small change in u, there is approximately the same change in s, showing a linear relationship between these two quantities, at the overlap region. The coordinates v and t are also linear in their relationship. When the dihedral angle of the image planes is closer to 180 degrees, the linearity in the overlap region extends.

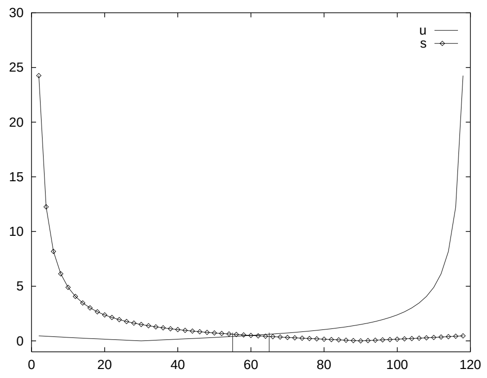

Figure 5: Relationship between u and s

This locally linear relationship between coordinates of the two image planes is an important geometric property that we exploit in our method to register the images into a panorama. Since the region of overlap is narrow and the angle between the image planes is large, the linear assumption is very accurate.

The next section elaborates on the geometry that our algorithms require of the compound sensor device. Section 2 presents the details of the algorithm for performing the geometric registration, including finding correspondences and compensating for lens distortion effects. Section 2.4 shows examples and results from the implementation, which has been used to render panoramic imagery from the cluster into a four-wall CAVE in

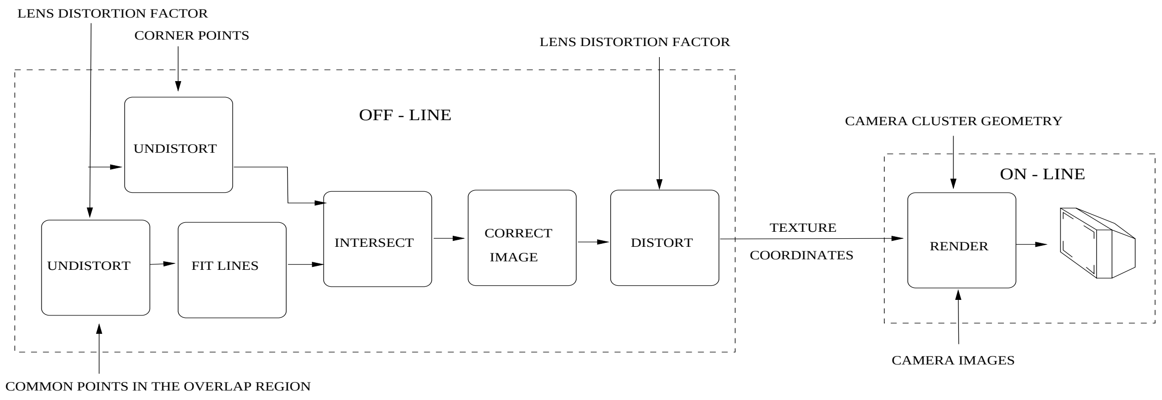

Figure 6: Sequence of Steps

real-time. Finally, section 4 proposes an algorithm to achieve photometric blending across the camera boundaries, and section 5 concludes with a summary and discuss the advantages of the approach.

2 Geometric Registration Algorithm: An Overview

Below are the assumptions we make to achieve accurate geometric registration of images captured by the camera cluster:

- The COP of all cameras in the cluster are approximately the same.
- The image region of overlap between the adjacent camera images is narrow, about 2 to 10 percent of the image dimension. For example, for a 500×500 image, the overlap is about 10 to 50 pixels.
- Fields of view (focal lengths) of the cameras can be unknown and are *not* needed by the algorithm.
- Radial lens distortion factors of the cameras are used if known.
- The color balance between images is assumed to vary.
- Camera cluster geometry is given.

The pipeline of our geometric registration process is shown in Figure 6. The algorithm proceeds in the following sequence of steps. Depending on the lens system, images from the cameras may show considerable radial lens distortion. Lens distortion estimates for each camera can be estimated using standard procedures [10, 11]. The four corner points of the trapezoidal region of all the camera images are extracted automatically using simple image processing (localized edge detection). These corner points of the trapezoid are corrected for lens distortions, and the resulting quadrilateral is our region of interest. This is followed by sampling the corresponding points between two adjacent cameras in the narrow overlap region. We describe a method in the next section to uniformly sample these common corresponding points. Since the overlap region is narrow, and the sampling of corresponding points is uniform, a least-squares line of these sampled points on each of these image planes *independently* would be the image of each other on these two image planes. We call these lines on the image planes *clipping lines*. The parameterization of these lines will be different up to a scale factor, depending on the field-of-view (focal length) of the cameras. We solve this problem without explicitly knowing the field of view by using texture mapping, as described later in this paper. In the geometry of our camera cluster, each camera has three adjacent neighbors, and hence has three clipping lines. These three clipping lines define an internal quadrilateral region in the initial trapezoidal region defined by the corner points. This defines a subregion in the trapezoid with *no overlap*. This subregion is then texture mapped on to a computer model of the corresponding mirror of the camera cluster geometry. When all the camera images are texture mapped, the panoramic view can be rendered from the center of the model of the camera cluster geometry.

As we are assuming a common COP, the geometric registration is an initialization process that needs to be done only once. Once the texture coordinates are found, captured images can be registered in real-time.

2.1 Capturing Corresponding Points

We find corresponding points on adjacent cameras by using a projector to emit structured patterns on a wall, which the adjacent cameras in the camera cluster can sense. The image of a camera i is processed, to obtain the mapping between projector coordinates (x, y), and the camera coordinates (u_i, v_i). When two cameras, say i and j, see the projected light from the same projector pixel (x, y), at say, (u_i, v_i) and (u_j, v_j) respectively, then $((u_i, v_i), (u_j, v_j))$ are the corresponding pixels for this camera pair. Since we assume a common COP, we need not know the distance of the wall from the camera cluster. Further, because of the same assumption, the overlapping region is constant, irrespective of the depth of object seen by the cameras. If the common COP assumption is invalid, the distance to an object becomes a factor in merging adjacent images of that object. We repeat the above procedure for different overlapping regions by rotating the camera cluster so that different pairs of adjacent cameras can see the wall without worrying about how far away the wall is. The projector is used just for the convenience of the procedure, and for automation.

2.2 Finding the Clipping Lines

The corresponding points collected for the camera pair (i, j), are represented as $((u_i, v_i), (u_j, v_j))$. A least square line, say L_i, is fit for (u_i, v_i), and another least square line, say L_j, is fit for (u_j, v_j). The image of L_i on the camera j is L_j, because of the linear relationship between the $u_i(v_i)$ and $u_j(v_j)$ in the overlapping region. The constant of proportionality in this linear relationship is attributed to the differences in the fields of view. If the fields of view of the cameras are same, then the transformation from $u_i(v_i)$ to $u_j(v_j)$ can be brought about by just a translation and rotation [2].

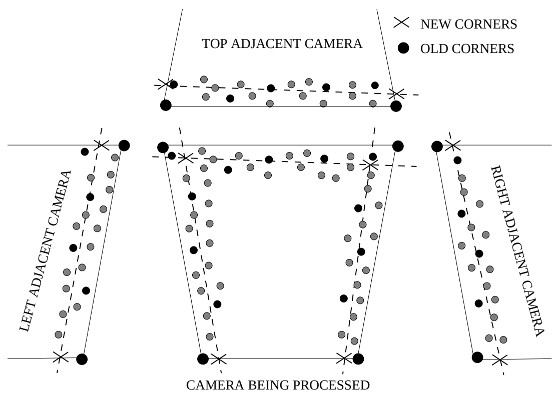

Figure 7: Clipping Lines and Corner Points

Figure 7 shows how this portion of the algorithm works for one camera. The corner points extracted from the trapezoid of the camera image are shown as large circular dots. The common corresponding points between the adjacent cameras are shown as small grey dots. The corresponding points shown as small black dots will be explained in the next section. The solid lines show the trapezoid formed by the corner points, and the dashed lines are the least-square-fit lines. The interesting intersection points of these lines are marked by crosses. These crosses are the new corner points, and the quadrilateral formed by these is the *adjusted* region

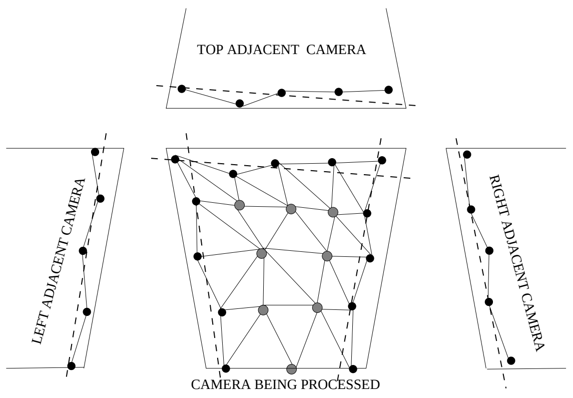

Figure 8: Correspondence based Image Correction

of interest. The interior of the adjusted region is the maximal image region that is devoid of any overlaps.

When this processing is done for all the cameras, these adjusted regions are texture-mapped onto trapezoidal faces (polygons) forming a model that resembles the camera cluster geometry. Though this texture map of the adjusted region should produce seamless stitching of images when viewed from the COP of the model, due to various inaccuracies and errors in computation, the images generated in the above method invariably will have some overlapping region which are visible in both of the adjacent cameras, or some clipped region which is not visible in any of these cameras. Finer corrections to the error generated in this line fitting is done by a correspondence based image correction.

2.3 Correspondence-Based Image Correction

This method performs a small pixel-based image warp on two adjacent images, so that a few corresponding points coincide at the common image boundary to create a seamless geometric registration.

Let us assume that the clipping lines between cameras i and j are $L_{ij}(\alpha)$ on camera i's image plane, and its corresponding line $L_{ji}(\beta)$ on the camera j's image plane. The corresponding corner points on the two image planes are given by $(L_{ij}(0), L_{ji}(0))$, and $(L_{ij}(1), L_{ji}(1))$. As the end points are corresponding points and as the relationship between the coordinates is linear, the image of $L_{ij}(\alpha)$, $0 \leq \alpha \leq 1$, on camera i is $L_{ji}(\alpha)$, $0 \leq \alpha \leq 1$, on camera j. This parameterization eliminates the need for knowing the scale factor brought in by the differing fields of view of different cameras.

As the first step in the image correction process, we select n points from the set of corresponding points $((u_i, v_i), (u_j, v_j))$, which are very close to the clipping lines L_{ij} and L_{ji}. Let these points be $((u_{ik}, v_{ik}), (u_{jk}, v_{jk}))$, $1 \leq k \leq n$. The lines joining these chosen points $(u_{ik}, v_{ik})((u_{jk}, v_{jk}))$ forms a piecewise linear approximation $M_{ij}(M_{ji})$ of $L_{ij}(L_{ji})$. Considering M_{ij} and M_{ji} as the new clipping boundaries, the next step is to parameterize M_{ij} and M_{ji}. Let the closest point in L_{ij} to (u_{ik}, v_{ik}) be $L_{ij}(\alpha_k)$. We assign $M_{ij}(\alpha_k)$ to be (u_{ik}, v_{ik}) and $M_{ji}(\alpha_k)$ to be (u_{jk}, v_{jk}). If there are no points assigned to $M_{ij}(0)(M_{ji}(0))$ and $M_{ij}(1)(M_{ji}(1))$ then we assign $M_{ij}(0) = L_{ij}(0)(M_{ji}(0) = L_{ji}(0))$ and $M_{ij}(1) = L_{ij}(1)(M_{ji}(1) = L_{ji}(1))$. All other values of $M_{ij}(M_{ji})$ are interpolated between the assigned values. This piecewise linear approximation is considered to be the new clipping boundary as shown in Fig. 8. This ensures registration at least at the assigned points, as they are actually the corresponding points. Thus, M_{ij} and M_{ji} give the texture coordinates for the boundary line between cameras i and j in our rendering process.

In Figs. 7 and 8, the points that are chosen as close to the clipping boundaries are marked with small dark dots on both the adjacent image planes. The distance of these points from the clipping line is exaggerated for the sake of this example.

To extend the above process to correct the lens distortion on the interior of the image also, multiple sample points are taken on the interior of the trapezoidal boundary. These sample points (grey dots in Fig. 8) are assigned the texture coordinates by interpolating between the undistorted boundary point texture coordinates. These interior sample points are denoted by grey dots in Fig. 8. This piecewise *non-linear* warp in the form of a texture map is a very powerful way to compensate for many non-linear effects that cause the imagery to be distorted, overlapping and far from seamless.

2.4 Rendering

Rendering of the images to achieve real-time geometric video registration is an important component of our algorithm. The undistorted image is clipped to the clipping lines and refined by the image correction methods. This resulting image is viewed as a texture map, and is mapped onto a triangulated trapezoidal planar face representing the particular geometry of the camera cluster (in our case a mirror face of the camera cluster). All the images are mapped on to the same sized planar geometry, which is defined by the physical geometry of the camera cluster. Thus the scale factor in the relationship between the coordinates of the two adjacent image planes due to the differing fields of view, is eliminated. Since the actual image after image correction is not exactly a trapezoid, as shown in Fig. 8, the texture mapping onto a trapezoidal face would introduce distortion. But in practice, this deviation of the texture from a trapezoid is not more than a couple of pixels, and the distortion effect is not perceptually significant.

It should be noted that all the above computed texture coordinates for the camera cluster geometry are from the undistorted image. Since we are texture-mapping distorted images from the live video stream from the cameras, the distorted texture coordinate corresponding to each undistorted texture coordinate is found by applying the inverse lens distortion transform. In particular, for the sake of simplicity, assume a square image $I'(U', V')$, which is the raw image from camera live feed. This image is corrected for lens distortion to $I(U, V)$. The corrected image is texture-mapped onto a plane $P(U, V)$, where a point $P(u, v)$ is assigned the color value of $I(u, v)$. In order to texture map directly from the live feed, we assign the color value of $I'(u', v')$ to $P(u, v)$, which is the distorted coordinate of (u, v). This process also corrects lens distortion, as the physical location of $P(u, v)$ pulls the pixel $I'(u', v')$ to the appropriate location, which is equivalent to lens undistortion.

3 Illustrations and Results

In color plate 1, we show the results after each of the steps of the algorithm for registering the top image with its adjacent images. The color plate 2 shows snapshots taken from our real-time rendering of the live images from a camera cluster of 10 cameras. Once we generated the texture file, we were able to move the camera cluster to various places and get the geometrically registered panoramic view of the environment around the camera cluster. The photometric differences in the colors of the different cameras make the seam across the camera visible in the first two snapshots in color plate 2. Currently, we are working on the problem of making the panorama photometrically seamless across the camera boundaries. In the following section, we give an overview of our approach. The seamless panorama as shown in the third snapshot in color plate 2 shows the initial results of this approach. Further, since there is some small mismatch in the alignment of the COP of the different cameras at a single point, we can see minor deviations from the registrations at a few places. The process of collecting the corresponding points takes about $20s$ for a cluster of 10 cameras. The off line process of generating the texture files, once the corresponding pixels are available, takes approximately $5ms$ for each camera. The process of generating the texture files for the whole cluster of cameras therefore takes about $60 - 70ms$. Rendering and timings were done on an SGI O2, with an R10000 CPU.

We have set up the first prototype of an immersive teleconferencing environment with our collaborators at

University of Illinois at Chicago. We capture the images from the camera cluster at University of North Carolina (UNC) and send it to Electronic Visualization Laboratory (EVL) at University of Illinois at Chicago (UIC) via the Internet. The precomputed texture map is sent from UNC to EVL-UIC. They get the images from the network, and using the pre-computed texture map render a geometrically registered video panorama on their CAVE system.

4 Photometric Seamlessness

As mentioned before, the camera boundaries are visible despite correct geometric registration because of the photometric difference between two cameras. Our approach to solve this photometric problem is similar to the way we solved the geometric registration problem. We are designing an algorithm that has an off-line component and an on-line component. The off-line algorithm will be used only once to determine a *pixel-based-weight-map*. Once this map is created, the blending will be done on-line using the graphics pipeline. For creating this weight map, we are using a classical algorithm for blending two images [12].

We use the *image spline* [12] technique to attain a smooth transition in the vicinity of the boundary between two images. A good image spline will make the transition smooth and yet will preserve most of the original image information. Using a weighted average splining technique (as in [12]), an image is first decomposed into a set of bandpass components. A different spline is used for each bandpass component to blend the transitions across the color boundary. Finally, the bandpass components are summed together to generate the desired blended image. The following subsections give an overview of the algorithm used in [12] and ways to extend it for real-time blending.

4.1 Definitions

This section reviews the definitions and different image operations that are used in [12] to achieve blending across image boundaries.

Let G_0 be the original image. A sequence of low-pass filtered images $G_0, G_1, \ldots, G_N$ can be obtained by repeatedly convolving a small weighting function with an image. If we imagine these images to be stacked above one another, the result is a tapering data structure known as a *pyramid.* If G_0 is of size (2^N+1) by (2^N+1), then there will be $N+1$ levels in the pyramid. This operation on the image at level $l-1$ to generate the image at level l is called $REDUCE$. So for $0 < l < N$, $G_l = REDUCE(G_{l-1})$. A Gaussian kernel is used as a low pass filter. The Gaussian kernel is defined by weights $w(m,n)$, where $-2 \leq m,n \leq 2$ and hence the pyramid we form is called the *Gaussian pyramid.* The mathematical expression of the function $REDUCE$ is as follows.

$$G_l(i,j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) G_{l-1}(2i+m, 2j+n)$$

where $0 \leq i,j < q$ and G_l is a $q \times q$ image.

Every pixel (i,j) in G_l represents a weighted average of 5×5 sub-array of pixels G_{l-1} centered at (i,j). Each of these in turn represents the average of a sub-array of pixels in G_{l-2}. In this way we can trace the weights back to G_0 to find an *equivalent weighting function*, W_l, which when convolved directly with G_0 will generate the image at G_l. This gives us a single mathematical transformation r_l to generate G_l from G_0, $G_l = r_l(G_0), 0 < l < N$.

The Gaussian pyramid is a set of low pass filtered images. Now we define an operation $EXPAND$ on G_l which is basically a super-sampling operation. Let $G_{l,k}$ be the image obtained by expanding G_l k times. So $G_{l,k} = EXPAND(G_{l,k-1})$. The mathematical expression for $EXPAND$ is

$$G_{l,k}(i,j) = 4 \sum_{m=-2}^{2} \sum_{n=-2}^{2} G_{l,k-1}\left(\frac{2i+m}{2}, \frac{2j+n}{2}\right)$$

where $0 \leq i,j < q$ and $G_{l,k-1}$ is a $q \times q$ image. This means that $G_{l,1}$ is of the same size as G_{l-1} and $G_{l,l}$ is the same size as the original image. Let the transformation $e_l(G_l) = G_{l,l}$. We now define a sequence of bandpass images L_l, $0 < l < N$, such that $L_l = G_l - G_{l+1,1}$, and $L_N = G_N$. This is called the *Laplacian pyramid.*

Just as each G_l in the Gaussian pyramid could have been obtained directly by convolving W_l with G_0, similarly, each L_l can be obtained by directly convolving the weighting function $(W_l - W_{l+1})$ with G_0. Thus we get a single mathematical transformation h_l such that $L_l = h_l(G_0), 0 < l < N$. Again, $L_{l,l} = e_l(L_l)$.

The steps used to construct the Laplacian pyramid may be reversed to recover the original image G_0 exactly. $L_N = G_N$ is first expanded and then added to L_{N-1} to generate G_{N-1}. Then G_{N-1} is expanded and added to L_{N-2} to generate G_{N-2}. Continuing likewise we can generate G_0. This can be written alternatively as

$$G_0 = \sum_{l=0}^{N} L_{l,l} = \sum_{l=0}^{N} e_l(L_l)$$

4.2 Algorithmic Overview

This section describes the modification of the above off-line algorithm to generate the *pixel-based-weight-map* and apply it in real-time. The panorama G is created from the images $I0, I1, \ldots, I9$ coming from ten different cameras, applying the geometric registration algorithm described in Section 2. The boundaries between these images are visible because of the photometric differences. The following procedure describes the method to generate the photometrically seamless panoramic image G_B from the geometrically registered panoramic image G.

We first generate Laplacian pyramids $P0, P1, \ldots, P9$ for $I0, I1, \ldots, I9$ respectively. Each level l of $Pi, 0 \leq i \leq 9$ is generated by $Pi_l = h_l(Ii)$.

A Laplacian pyramid BP, for the panoramic image G is created by compositing $P0, P1, \ldots, P9$ as described in [12]. Let the compositing function for level l be f_l, such that $BP_l = f_l(P1_l, P2_l, \ldots, P9_l)$, where BP_l is level l for the laplacian pyramid BP. Thus the final blended image

$$G_B = \sum_l BP_{l,l} = \sum_l e_l(BP_l)$$

$$G_B = \sum_l e_l(f_l(h_l(I1), \ldots, h_l(I9)))$$

Now, since we know the precise definitions of the family of functions e_l, f_l and h_l, by expanding the above equation, we get a closed form solution for G_B. For each $G_B(x, y)$ we can find a set of pixels $S_{x,y}$ in G and corresponding weights $w(u, v)$ for each pixel $(u, v) \in S_{x,y}$ such that

$$G_B(x, y) = \sum_{S_{x,y}} w(u, v).G(u, v)$$

Though it may seem that this will give a function for each $G_B(x, y)$, in practice, only the pixels in the neighborhood of the boundaries will have more than one pixel in $S_{x,y}$.

The result will look like Fig 3 in color plate 3, which is obtained by implementing the off-line version of the algorithm [12]. We are currently investigating the techniques necessary to use the graphics pipeline so that we can achieve real-time performance.

5 Summary and Conclusions

In this paper we have presented an algorithm for generating real-time panoramic images from the input of a common COP camera cluster. One novel aspect of our approach is to use *texture map* that encodes the image warp and the lens distortion correction to achieve the geometric registration. Thus, the algorithm decouples geometric registration, performed as an off-line initialization step, from the on-line rendering process. This decoupling allows us to use conventional graphics rendering pipelines effectively to achieve real-time rendering. These steps are instrumental in achieving *high-resolution real time panoramic image generation*, which can be used to register live images from a cluster of cameras having a common COP.

A second characteristic of our algorithm is *locality* of application. Each camera is processed independently, and the algorithm attains *global alignment* through a sequence of *local alignments*. This makes it simple to adjust individual cameras in a cluster, since the algorithm needs to be applied only to the camera to be adjusted.

Finally, in the face of cameras that exhibit large color inconsistencies, our algorithm uses a structured-light approach and concentrates on feature matches rather than correlation methods that may be impacted by the color variation. We have also proposed a way to achieve real-time photometric blending across the camera boundaries in the geometrically registered video panorama. This algorithm aims at using the conventional graphics pipeline to achieve its goals.

We have also set up the first prototype of immersive teleconferencing with our collaborators at Electronic Visualization Laboratory at University of Illinois at Chicago.

References

[1] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, H. Fuchs, "The Office of the Future : A Unified Approach to Image-Based Modeling and Spatially Immersive Displays," *SIGGRAPH*, pp. 179-188, 1998.

[2] R. Szeliski, H. Shum, "Creating Full View Panoramic Mosaics and Environment Maps," *SIGGRAPH*, pp. 251-258, 1997.

[3] Sanjib K. Ghosh, *Analytical Photogrammetry, second edition,* Pergamon, 1988.

[4] R. Hartley, G. Gupta, "Linear Pushbroom Cameras," *In J. Eklundh, editor, Third European Conf. on Computer Vision* , pp. 555-566, Stockholm, Sweden,May 1994.

[5] S. Peleg, J. Herman "Panoramic Mosaics by Manifold Projection," *In Proceedings of CVPR,* pp. 338-343, June 1997.

[6] S. Baker, S. Nayar, "A Theory of Catadioptric Image Formation," *In Proceedings of CVPR,* pp. 35-42, June 1997.

[7] S. Nayar, "Catadioptric omnidirectional camera," *In Proceedings of CVPR,* pp. 482-488, June 1997.

[8] V. Nalwa, "A true omnidirectional viewer," *Tech Report Bell Laboratories,* Holmdel NJ 07733 USA Feb 1996.

[9] S.E.Chen, "QuickTime VR - An Image Based Approach to Virtual Environment Navigation," *SIGGRAPH,* pp. 29-38, 1995.

[10] O. D. Faugeras, *Three-Dimensional Computer Vision: A Geometric Approach, first edition,* MIT Press, 1993.

[11] R. Tsai, "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses," *IEEE Journal of Robotics and Automation,* 3(4) pp. 323-344, 1987.

[12] P.J.Burt, E.H.Adelson, *A Multiresolution Spline With Application to Image Mosaics,* ACM Transaction on Graphics, Vol. 2, No. 4, Oct 1983, pp 217-236.

[13] T.Kawanishi, K.Yamazawa, H.Iwasa, H.Takemura, N.Yokoya, " Generation of high resolution stereo panoramic images by omnidirectional sensor using hexagonal pyramidal mirrors", *ICPR '98*, pp 485-489.