Spherical Mosaics with Quaternions and Dense Correlation

Satyan Coorg Seth Teller
MIT Computer Graphics Group

Abstract

We describe an algorithm for generating spherical mosaics from a collection of images acquired from a common optical center. The algorithm takes as input an arbitrary number of partially overlapping images, an adjacency map relating the images, initial estimates of the rotations relating each image to a specified base image, and approximate internal calibration information for the camera. The algorithm's output is a rotation relating each image to the base image, and revised estimates of the camera's internal parameters.

Our algorithm is novel in the following respects. First, it requires no user input. (Our image capture instrumentation provides both an adjacency map for the mosaic, and an initial rotation estimate for each image.) Second, it optimizes an objective function based on a global correlation of overlapping image regions. Third, our representation of rotations significantly increases the accuracy of the optimization. Finally, our representation and use of adjacency information guarantees globally consistent rotation estimates.

The algorithm has proved effective on a collection of nearly four thousand images acquired from more than eighty distinct optical centers. The experimental results demonstrate that the described global optimization strategy is superior to non-global aggregation of pairwise correlation terms, and that it successfully generates high-quality mosaics despite significant error in initial rotation estimates.

1 Automatic Spherical Mosaicing

This paper gives an algorithm for robust estimation of the rotations relating images taken from a common optical center[1]. We call this a "spherical mosaicing" algorithm because it allows any number of images to be merged into a single seamless view, simulating the image that would be acquired by a camera with a spherical field of view. Our data represents a field of view that is somewhat more than a hemisphere, but our technique extends straightforwardly to full spherical arrangements of images. The resulting mosaics are useful as "first-class" data objects for 3-D reconstruction, and as a compact means for visualizing spatially extended image datasets.

This work is motivated by a system under development for automatic reconstruction of textured 3D CAD models representing urban environments [Tel97, CMT98, CT99]. The system requires acquisition and processing of a suitable dataset, in this case a large number of digital images of the region of interest. Our instrumentation annotates each acquired image with an estimate of absolute 6-DOF *pose* (also called exterior orientation) – 3 DOF of position, and 3 DOF of orientation for the acquiring camera. Thus our acquisition system provides both an adjacency map for images in the mosaic, and an initial estimate of the rotations relating each image to its neighbors.

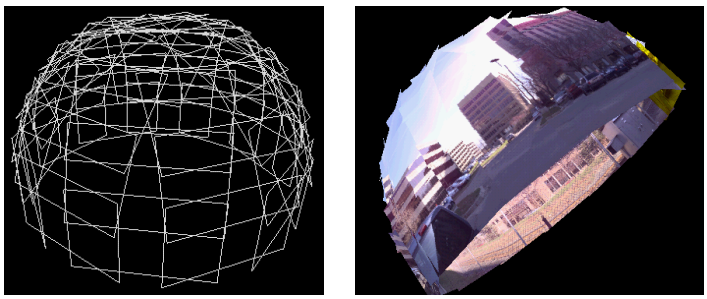

Figure 1: The roughly hemispherical tiling for a node of the dataset.

Our dataset consists of photographs acquired by a Kodak DCS 420 digital camera mounted with fixed optical center on an indexed pan-tilt head, itself attached to a tripod base. The tripod was manually positioned at eighty-one locations among the buildings of an office complex. At each position, the camera was rotated through a predetermined "tiling" of 50-70 orientations,

[1]An earlier version of this paper was presented in CVPR 1998 [CMT98].

yielding a roughly hemispherical field of view (Figure 1). We call each set of images obtained from a common optical center a "node". The tiling defines an adjacency map over the images, in which two images are adjacent if they have significant overlap along a shared vertical or horizontal edge.

Physical instrumentation alone does not produce pose estimates sufficiently accurate for direct incorporation into 3-D reconstruction algorithms. For example, our actuated pan/tilt head produces orientation estimates accurate to about one degree. This is one to two orders of magnitude less accurate than the pixel or sub-pixel alignment required for typical reconstruction algorithms. It is thus necessary to design *pose refinement* algorithms that recover accurate camera pose from (approximate) initial estimates.

This paper describes our method for refining estimates of camera orientation for images taken around a single optical center. Recovering relative translations and orientations between mosaics acquired at different optical centers is addressed elsewhere [Coo98, ATar, ATon].

Related Work

Of fundamental interest in mosaic computations is the warp relating a pair of overlapping images. The simplest method to compute this warp uses four point correspondences between the two images [Hec89, Fau93]. Several algorithms (e.g., [ZFD97]) use this approach. However, identifying suitable features and correspondences is a difficult problem, and known methods yield good results only for images with significant overlap and minimal projective distortion.

An alternative method would use correlation of color or luminance information present in images to compute the warp by nonlinear optimization (e.g., [Sze96]). Although such techniques avoid the need for feature detection and correspondence, they do not guarantee that a series of pairwise warps will produce a globally consistent set of relative orientations. We show an example of this problem in Section 4.

The choice of representation for rotations is an important practical issue in the development of algorithms for mosaic generation. For estimation of small rotations, the axis-angle representation has been used [Sze96]. However, that algorithm also uses matrices to avoid instability due to the non-uniqueness of the axis-angle representation. We have adopted the quaternion representation because of its convenience and compactness [Hor87, WI95].

Cylindrical panoramas have been computed by McMillan [MB95], who

solves for rotation angles for images taken under rotation around a single (vertical) axis. His algorithm enforces the constraint that angles computed for a cylindrical panorama should sum to 2π. A similar constraint is employed by Szeliski and Shum [SS97] to "close the gap" between the first and last images. These methods do not generalize to arbitrary (e.g., spherical) image adjacency maps.

Shum and Szeliski in their recent paper [SS98] also compute full-view spherical panoramas. However, their global alignment algorithm requires a combination of both correlation-based and feature-based optimization. In contrast, we optimize correlation directly to perform global alignment, avoiding both the need to identify and correspond suitable features.

In a different application domain, robust algorithms have been described for generating mosaics of video frames taken from a slowly rotating camera [SHK98, HS98]. These algorithms assume a single continuous image sequence, and exploit similarity between successive images due to the small camera rotation between video frames. In contrast, our technique handles general arrangements of images on the sphere.

1.1 Overview

(a) (b) (c)

Figure 2: Part (a) shows one image of a hemispherical tiling blended with its adjacent images. Part (b) illustrates blurring due to incorrect pose estimates. Part (c) shows the same view after optimization.

Figure 2 illustrates the basic idea behind the optimization technique. As depth and parallax effects do not occur across images taken from a single optical center [Har97], the apparent pixel motion between such images can be

explained by a 2-D *projective* transformation that depends on camera orientations and internal parameters (described below). As shown in Figure 2-(a), this transformation maps pixels from adjacent images into a common 2-D space. Incorrect transformations arising from inaccurate estimates of camera pose result in mismatches between pixels, causing the *blurring* and *ghosting* shown in Figure 2-(b). The optimization techniques described here uses these pixel differences to refine pose estimates and eliminate the blurring artifacts, as shown in Figure 2-(c).

The rest of the paper is organized as follows. Section 2 briefly describes the process of image formation via perspective projection, and our representation of rotations by quaternions. Section 3 reviews 2-D projective transformations and methods to compute them. Section 4 presents a closed-form method to decouple projective transformations into two parts, one describing the intrinsic parameters of the camera, and another describing the pure rotations to which the camera has been subjected. While theoretically elegant, this technique is sensitive to errors in image formation and generally yields poor results for real imagery. We address this problem in Section 5 with a global optimization technique that computes revised rotations and camera internal parameters directly from correlations among images. Constraining the optimization to manipulate pure rotations produces significantly more accurate mosaics, as shown in Section 6.

2 Perspective Projection

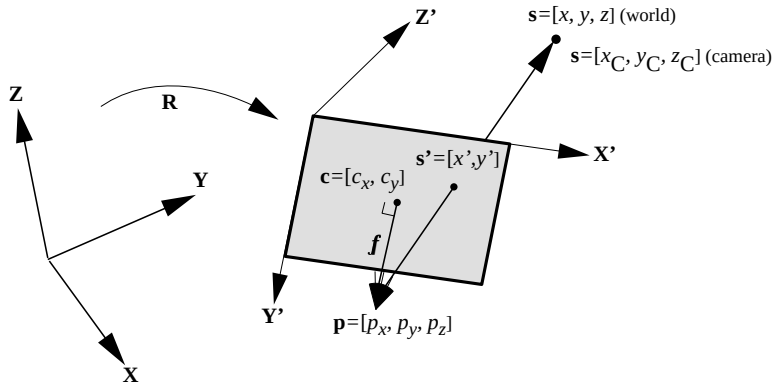

Figure 3: Overview of perspective projection.

Figure 3 shows the process of image formation by perspective projection, illustrated for a point $\mathbf{s} = [x, y, z]$ as viewed by a camera at world-space position $\mathbf{p} = [p_x, p_y, p_z]$. The rotation from the global coordinate system **XYZ** to the camera coordinate system $\mathbf{X'Y'Z'}$ is specified by a 3×3 rotation matrix **R**.

The first step is to compute $\mathbf{s_C} = [x_C, y_C, z_C]$, the coordinates of **s** in the camera's coordinate system:

$$\mathbf{s_C} = \mathbf{R}(\mathbf{s} - \mathbf{p})$$

Next, perspective projection scales the x and y coordinates by depth to yield normalized image coordinates $[x_I, y_I]$:

$$x_I = \frac{x_C}{z_C} \quad y_I = \frac{y_C}{z_C}$$

The normalized image coordinates are converted to pixel values based on the focal length f (expressed in pixels) and the coordinates (c_x, c_y) of the principal point:

$$x' = fx_I + c_x \quad y' = fy_I + c_y$$

Note that it is not necessary that the principal point coincide with the image center; in practice, it is usually offset by a few pixels. It is convenient to represent the entire transformation as a chain of matrix transformations. This is done using projective geometry [Fau93]:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \cong \mathbf{KM} \begin{bmatrix} \mathbf{R} & -\mathbf{Rp} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{1}$$

where **K** is the 3×3 (upper-triangular) internal camera parameter matrix:

$$\begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

and **M** is the 3×4 canonical perspective projection matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

The projective equality ($\cong$) is valid up to scaling.

Note that this discussion restricts the camera to be described by a three parameter model (instead of the standard five parameter model [Fau93]) by assuming that the pixels are square (i.e., the focal lengths are equal) and non-linear distortion is negligible. These assumptions are valid for the digital camera used in our system; radial distortion is less than a pixel even near the edge of the image. It is straightforward to incorporate non-square pixels in the algorithms by using different focal lengths. However, any non-linear distortion must be measured and corrected (e.g., with Stein's method [Ste95]) before using the images as input to the algorithms. Alternatively, estimation and correction for non-linear distortion can be incorporated directly into the mosaic algorithm [SK99].

2.1 Rotations as Quaternions

There are several choices for expressing camera rotations: 3×3 orthonormal matrices, Euler angles, quaternions etc. Horn [Hor87, Hor91] demonstrates that quaternions are a convenient representation, especially for problems involving numerical optimization. Quaternions represent rotations as four-dimensional unit vectors $\mathbf{q} = [q_0, q_x, q_y, q_z]$ where $q_0^2 + q_x^2 + q_y^2 + q_z^2 = 1$.

The quaternion representation is compact in comparison to the orthonormal representation (only four parameters instead of nine). Unlike the even more compact Euler angle representation (three angles expressing rotations about the $\mathbf{X}, \mathbf{Y}, and \mathbf{Z}$ axes), quaternions are stable while representing large rotations, and exhibit no singularities.

Differentiating a Rotation Matrix

The derivative of the rotation matrix with respect to the quaternion parameters is an important quantity used by our algorithm to update the estimate of each camera's associated rotation.

The following expression represents a quaternion as a 3×3 orthonormal matrix:

$$\begin{bmatrix} (q_0^2 + q_x^2 - q_y^2 - q_z^2) & 2(q_x q_y - q_0 q_z) & 2(q_x q_z + q_0 q_y) \\ 2(q_y q_x + q_0 q_z) & (q_0^2 - q_x^2 + q_y^2 - q_z^2) & 2(q_y q_z - q_0 q_x) \\ 2(q_z q_x - q_0 q_y) & 2(q_z q_y + q_0 q_x) & (q_0^2 - q_x^2 - q_y^2 + q_z^2) \end{bmatrix}$$

The derivative of a rotation matrix $\mathbf{R}^{-1}$ with respect to a quaternion

$\mathbf{q} = [q_0, q_x, q_y, q_z]^T$ is a tensor of dimension $3 \times 4 \times 3$. Here, the derivative is typically multiplied by a vector $\mathbf{v} = [v_x, v_y, v_z]^T$; only its value at $\mathbf{v}$ is required:

$$
\begin{aligned}
(\frac{\partial \mathbf{R}^{-1}}{\partial \mathbf{q}})_{\mathbf{v}} &= \begin{bmatrix} a & d & -c & b \\ b & c & d & -a \\ c & -b & a & d \end{bmatrix} \\
\text{where:} \quad a &= +q_0 v_x + q_z v_y - q_y v_z \\
b &= -q_z v_x + q_0 v_y + q_x v_z \\
c &= +q_y v_x - q_x v_y + q_0 v_z \\
d &= +q_x v_x + q_y v_y + q_z v_z
\end{aligned}
$$

For representing rotations, quaternions hold a significant computational advantage over other representations (Euler angles, axis-angle). In these other representations, derivative expressions contain sine and cosine terms, increasing the computational cost of estimating derivatives.

3 2-D Projective Transformations

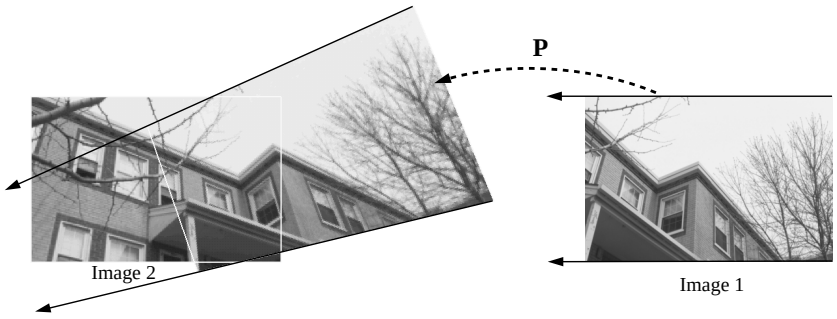

Figure 4: Transforming pixels from image 1 to the space of image 2.

Figure 4 illustrates the relationship between two images taken from a fixed optical center, but with differing orientations. In such cases, pixels in one image can be mapped to the other image by a 2-D *projective* transformation [Har97]. Unlike simpler 2-D transformations (translation, rotation, affine), the projective transformation *does not* preserve parallel lines. This is evident in Figure 4, where the lines bounding image 1 intersect after transformation.

As depth effects do not occur across two images taken from the same optical center [Har97, Sze96], the general perspective projection (Equation 1) simplifies to:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \cong \mathbf{KR} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2}$$

Inverting Equation 2 yields:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \cong \mathbf{R}^{-1}\mathbf{K}^{-1} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \tag{3}$$

Equation 3 provides a method to convert pixel positions in one image (say, image 1) to 3-D rays. Thus pixel coordinates in another image (say, image 2) can be obtained by projecting back into image 2's space using Equation 2:

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \cong \mathbf{K}\mathbf{R}_2\mathbf{R}_1^{-1}\mathbf{K}^{-1} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \tag{4}$$

Thus the 2-D *projective* transformation that maps pixel (x_1, y_1) of image 1 to pixel (x_2, y_2) of image 2 is:

$$\mathbf{P} = \mathbf{K}\mathbf{R}_2\mathbf{R}_1^{-1}\mathbf{K}^{-1} \tag{5}$$

Note that, as $\cong$ denotes projective equality, Equation 4 is valid only up to a linear scaling of $[x_2, y_2, 1]^T$. As a consequence, only eight parameters are needed to describe the matrix $\mathbf{P}$. Thus 2-D projective transformations are also known as *8-parameter warps*. Typically, the unknown scale factor of the transformation is determined by fixing either $\det \mathbf{P} = 1$ or $\mathbf{P}_{33} = 1$.

3.1 Computing Warps

Below, we briefly review an optimization due to Szeliski [Sze96] that computes 8-parameter warps using this technique. This discussion also introduces the Levenberg-Marquardt (LM) optimization [PTVF92], which is used in several places here.

The idea is to compute a warp that (locally) minimizes image-space error by using nonlinear optimization. The error function for this optimization

simply measures the difference in brightness between two images 1 and 2 (in the overlap region), after pixels in image 1 are mapped to image 2's space. The difference in brightness is measured by a sum-of-squared difference (SSD) error metric using the luminances L_1 and L_2 of images 1 and 2, respectively:

$$E_{12} = \sum_{x_1,y_1} (L_1(x_1, y_1) - L_2(\mathbf{P}(x_1, y_1)))^2 \tag{6}$$

This error term is "one-way", in the sense that image 1 is used as a sampling reference for collection of pixel differences. However, we collect error symmetrically, by evaluating both E_{ij} and E_{ji} for every adjacent image pair i and j (Section 5.1). The SSD form is well suited for numerical optimization, as only first order derivatives are required to compute update values for the iteration [PTVF92].

The optimization consists of analytically determining derivatives of a single error term of the form:

$$e^2_{x_1,y_1} = (L_1(x_1, y_1) - L_2(x_2, y_2))^2$$

with respect to $\mathbf{P}$. The derivative $\frac{\partial e_{x_1,y_1}}{\partial \mathbf{P}}$ is expressed as an 8-component vector consisting of derivatives with respect to each entry of $\mathbf{P}$.

In LM optimization, the overall gradient term $\mathbf{G}$ is computed by accumulating over all error terms [PTVF92]:

$$\mathbf{G} = -\sum_{x_1,y_1} e_{x_1,y_1} \frac{\partial e_{x_1,y_1}}{\partial \mathbf{P}}$$

Similarly, the (linearized) Hessian term corresponding to two adjacent images 1 and 2 is:

$$\mathbf{H} = -\sum_{x_1,y_1} \frac{\partial e_{x_1,y_1}}{\partial \mathbf{P}} (\frac{\partial e_{x_1,y_1}}{\partial \mathbf{P}})^T$$

where we sum over all pixels (x_1, y_1) in the overlap region of the images.

The optimization proceeds by incrementing the value of $\mathbf{P}$ by

$$\mathbf{\Delta P} = -(\mathbf{H} + \lambda \mathbf{I})^{-1} \mathbf{G}$$

where $\mathbf{I}$ is the identity matrix, and λ is a stabilization parameter set initially to a high value, and reduced to 0 as the optimization converges [PTVF92].

Initialization

As in any nonlinear optimization, it is important to initialize the warp with a value that is close to the optimum. Several techniques have been proposed for this step. McMillan [McM97] computes a 2-D translation with minimum error, and uses it to initialize the optimization. However, good initial translations are difficult to determine, or may not exist at all, if the effects of perspective are large. Szeliski and Shum [SS97] perform the initialization interactively, with the help of a human operator.

In our application, initial rotation estimates are provided by our acquisition instrumentation, and approximate camera calibration. It is thus straightforward to compute a good initial estimate using Equation 5 with the internal parameter matrix determined by camera calibration and rotations supplied by physical instrumentation.

Warp Example

Figure 5 shows the results of this optimization for two images (sized 762×506 pixels). Note that the blurring, initially present, disappears after optimization. The projective matrix $\mathbf{P}$ computed (normalized such that its determinant is 1) is:

$$\mathbf{P} = \begin{bmatrix} 0.7423 & 0.0162 & 584.7688 \\ -0.0994 & 0.9983 & 14.2725 \\ -0.0003 & 0.0000 & 1.0463 \end{bmatrix} \tag{7}$$

We next describe a technique to recover camera internal parameters and relative rotations from such warps.

4 Orientation Estimates from Warps

It is straightforward to recover rotation from warps using Equation 5 if the camera calibration is known accurately. If not, the following closed-form solution can be used to derive camera calibration from the warp itself.

4.1 Closed-Form Solution for Internal Parameters

The technique proposed in this section is similar in spirit to closed-form solutions presented by Hartley [Har97] and McMillan [McM97], but much simpler to express. The basic idea is the use of Equation 5 to enforce the

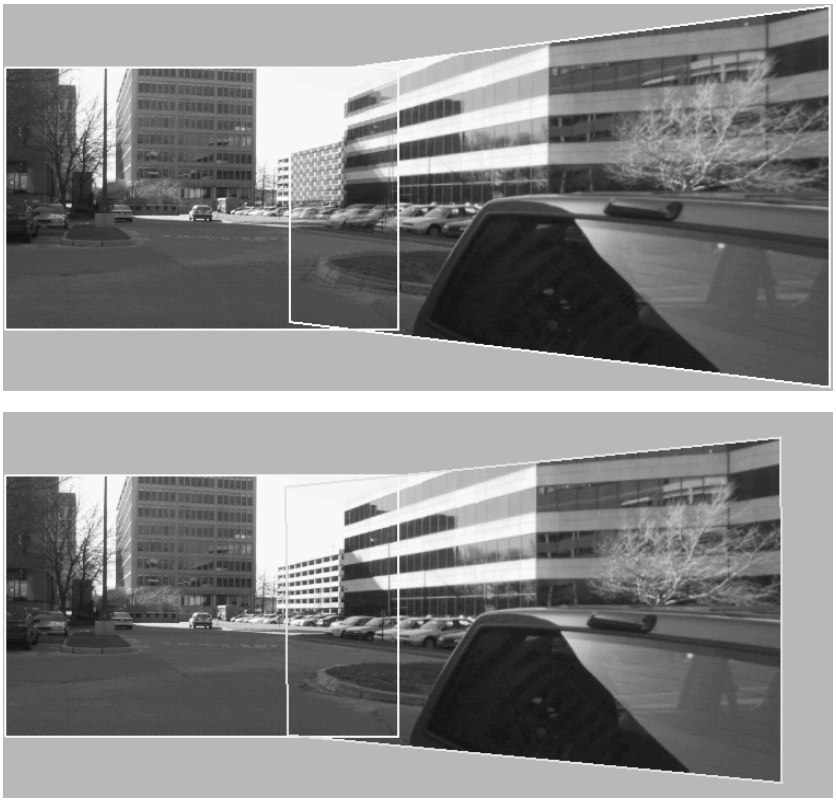

Figure 5: The projective warp between two images, before and after optimization.

orthonormality constraint for valid 3×3 rotational matrices, and the use of eigen-vectors of the warp matrix to define the solution.

Rewriting Equation 5 in terms of $\mathbf{R} = \mathbf{R}_2\mathbf{R}_1^{-1}$, the relative rotation between the two images, and solving for $\mathbf{R}$, we have:

$$\mathbf{R} = \mathbf{K}^{-1}\mathbf{P}\mathbf{K} \tag{8}$$

Since $\mathbf{R} = \mathbf{R}^{-T}$ (the orthonormality condition for rotations), we have

$$\mathbf{K}^{-1}\mathbf{P}\mathbf{K} = \mathbf{K}^T\mathbf{P}^{-T}\mathbf{K}^{-T}$$

Rearranging terms yields:

$$\mathbf{P}\mathbf{C} = \mathbf{C}\mathbf{P}^{-T} \tag{9}$$

where $\mathbf{C} = \mathbf{K}\mathbf{K}^T$. The matrix $\mathbf{C}$ is a symmetric 3×3 matrix whose components are formed from the focal length f and image principal point $[c_x, c_y]$:

$$\mathbf{C} = \begin{bmatrix} f^2 + c_x^2 & c_x c_y & c_x \\ c_x c_y & f^2 + c_y^2 & c_y \\ c_x & c_y & 1 \end{bmatrix} \tag{10}$$

The solution for $\mathbf{C}$ in Equation 9 can be expressed in terms of eigen-vectors of the projective matrix $\mathbf{P}$. The eigen-values of $\mathbf{P}$ are the same as that of $\mathbf{R}$, since they are related by a *similarity transform* [Str88], i.e., left and right multiplication by a matrix and its inverse (Equation 8). The eigen-values of the rotation $\mathbf{R}$ are 1, $e^{i\theta}$, and $e^{-i\theta}$, where θ is the angle of rotation effected by $\mathbf{R}$. Let $\mathbf{e}_0$, $\mathbf{e}_1$ and $\mathbf{e}_2$ be the eigen-vectors of $\mathbf{P}$ corresponding to these three eigen-values, respectively.

Note that $\mathbf{C} = \mathbf{e}_0\mathbf{e}_0^T$ is a solution to Equation 9:

$$\mathbf{P}\mathbf{C} = \mathbf{P}\mathbf{e}_0\mathbf{e}_0^T = \mathbf{e}_0\mathbf{e}_0^T = \mathbf{e}_0\mathbf{e}_0^T\mathbf{P}^{-T} = \mathbf{C}\mathbf{P}^{-T}$$

This derivation uses the fact that if $\mathbf{e}$ is a right eigen-vector of $\mathbf{P}$, then $\mathbf{e}^T$ is a left eigen-vector of $\mathbf{P}^{-T}$.

Similarly, as the eigen-values corresponding to $\mathbf{e}_1$ and $\mathbf{e}_2$ are reciprocals of each other, it can be shown that $\mathbf{e}_1\mathbf{e}_2^T$ and $\mathbf{e}_2\mathbf{e}_1^T$ are also possible solutions of $\mathbf{C}$. The most general symmetric solution to $\mathbf{C}$ is therefore a linear combination of the three solutions, where the second two solutions are weighted equally:

$$\mathbf{C} = c_0\mathbf{e}_0\mathbf{e}_0^T + c_1(\mathbf{e}_1\mathbf{e}_2^T + \mathbf{e}_2\mathbf{e}_1^T)$$

The coefficients c_0 and c_1 are solved by enforcing the constraints that the $\mathbf{C}_{33} = 1$ and $\mathbf{C}_{12} = \mathbf{C}_{13}\mathbf{C}_{23}$ (again using the three-parameter camera model). The matrix $\mathbf{K}$ can be recovered by Cholesky Decomposition of $\mathbf{C}$ [PTVF92].

Using this technique for the warp described in Equation 7 gives values of $f = 1141$, $c_x = 397$, $c_y = 249$ (all in pixels). The rotation $\mathbf{R}$ corresponding to these parameters is:

$$\begin{bmatrix} 0.8897 & 0.0090 & 0.4603 \\ -0.0004 & 0.9961 & 0.0010 \\ -0.4599 & 0.0080 & 0.8901 \end{bmatrix}$$

The rotation determined appears qualitatively reasonable, i.e., it is (close to) a rotation about the image's y axis; this agrees with Figure 5. However, as described in the next section, there is a significant quantitative difference between this and the physically correct rotation.

4.2 Warps and Rotations

The rotation computed in the previous section is approximately 27° (= $\cos^{-1} 0.89$). However, the physical rotation was nearly 30° (= $\frac{360}{12}$ as twelve images formed a full circle). Using a 27° rotation for this image sequence would therefore leave a gap between the last and the first image.

At first glance, it might appear that the problem is due to incorrectly recovered internal parameters. The computed focal length differs significantly from the focal length determined by Tsai's calibration algorithm (by about 150 pixels). However, even though a different set of internal parameters might yield other rotations, the *angle* of rotation is completely determined by the eigen-values of the projective warp $\mathbf{P}$. As this is unaffected by the method by which internal parameters are computed, this problem is inherent in the warp solution itself.

Figure 6: The projective warp between two images, after direct optimization.

Figure 6 illustrates the problem by showing the warp computed by direct optimization with respect to the internal parameters $\mathbf{K}$ and the relative rotation $\mathbf{R}$ (using the method described below). The internal parameters were computed to be $f = 1016$, $c_x = 393$, $c_y = 253$ (all in pixels), and the relative rotation and warp between the two images are:

$$\mathbf{R} = \begin{bmatrix} 0.8637 & 0.0047 & 0.5040 \\ -0.0054 & 1.0001 & -0.0000 \\ -0.5043 & -0.0031 & 0.8638 \end{bmatrix} \quad \mathbf{P} = \begin{bmatrix} 0.669371 & 0.003488 & 592.614624 \\ -0.114616 & 0.999421 & 15.121033 \\ -0.000492 & -0.000003 & 1.058760 \end{bmatrix}$$

The rotation (of $cos^{-1}0.8637 = 30.27°$) and internal parameters agree well with initial estimates. Also, visually, the overall warp computed differs from

that described in Equation 7. This can be observed in the large "gap" between the right edge of the image and the border in Figure 5 (which is smaller in Figure 6). However, within the region of overlap, the two warps appear identical; there are no blurring artifacts visible in either case. Thus, there are several possible warps that yield low SSD error, but not all of them correspond to physically correct rotations. It is therefore essential to impose the rotational constraint *during* the optimization to obtain quantitatively correct results.

Similar effects were also observed for warps corresponding to other image pairs. Thus, while 8-parameter warps generate visually consistent blending between adjacent images, they appear to be inadequate for recovering quantitative 3-D parameters, such as relative rotations. Also, more fundamentally, relying on local *pairwise* warps to compute global quantities can lead to inconsistencies in the computed internal parameters and rotations. The next section presents a global optimization method that addresses these problems.

5 Spherical Mosaicing

The optimization described in this section directly produces the "best" possible rotations for *each* image, given initial estimates. The advantage of this approach is that global consistency is guaranteed by computing a *unique* rotation for each image. That is, the pairwise rotations inferred from our representation have the property that the aggregate rotation along any cycle in the image adjacency map is the identity. In this manner, our representation avoids the possibility of "gaps" arising from inconsistent pairwise estimates.

The approach followed is to optimize a global correlation function defined for adjacent images with respect to all orientations (represented as quaternions). As a by-product, the algorithm computes a *spherical mosaic*, a composite of all images corresponding to a single node.

5.1 Optimization

The algorithm minimizes a global error function:

$$E = \sum_{i,j \text{ are adjacent}} E_{ij} + E_{ji}$$

where E_{ij} is the SSD error between luminance values of images i and j:

$$E_{ij} = \sum_{x_i, y_i} (L_i(x_i, y_i) - L_j(\mathbf{P}_{ij}(x_i, y_i)))^2$$

and $\mathbf{P}_{ij}$ maps coordinates of image i to those of image j. This correlation function is computed only for pairs of adjacent images in the spherical tiling, and only for pixels of image i that map to a valid pixel of image j. Even though the SSD error function could be minimized by forcing all images to become non-overlapping, in practice this does not happen, for two reasons. First, the initial estimates are sufficiently good that gradient descent moves toward the true optimum. Second, there is not enough "room" on the sphere for so many rigid quadrangles to become non-overlapping. In practice, our algorithm converges to a value close to the initial estimates, with significant overlap between adjacent images.

As in the pairwise warp estimation, this function is minimized by computing derivatives with respect to each orientation and using LM nonlinear optimization starting from the initial orientations. The various steps in the computation are described in detail below.

For a single error term for images i and j of the form:

$$e_{x,y}^2 = (L_i(x, y) - L_j(x'', y''))^2$$

with

$$x'' = \frac{x'}{z'} \quad y'' = \frac{y'}{z'}$$

and

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \mathbf{v}' = \mathbf{P} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{K}\mathbf{R}'\mathbf{R}^{-1}\mathbf{K}^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{11}$$

the derivatives are computed as follows (using the rotation-matrix derivative given in Section 2.1):

$$\frac{\partial \mathbf{v}'}{\partial \mathbf{q}} = \mathbf{K}\mathbf{R}'(\frac{\partial \mathbf{R}^{-1}}{\partial \mathbf{q}})_{\mathbf{v}}$$

where $\mathbf{v} = \mathbf{K}^{-1}[x, y, 1]^T$. Then, the derivative of the term $e_{x,y}$ with respect to the quaternion $\mathbf{q}$ is given by:

$$\frac{\partial x''}{\partial \mathbf{q}} = \frac{\frac{\partial x'}{\partial \mathbf{q}} - x'' \frac{\partial z'}{\partial \mathbf{q}}}{z'} \quad \frac{\partial y''}{\partial \mathbf{q}} = \frac{\frac{\partial y'}{\partial \mathbf{q}} - y'' \frac{\partial z'}{\partial \mathbf{q}}}{z'} \tag{12}$$

$$\frac{\partial e_{x,y}}{\partial \mathbf{q}} = \frac{\partial L_j}{\partial x''}\frac{\partial x''}{\partial \mathbf{q}} + \frac{\partial L_j}{\partial y''}\frac{\partial y''}{\partial \mathbf{q}} \tag{13}$$

Equation 13 involves image derivatives $\frac{\partial L_j}{\partial x''}$ and $\frac{\partial L_j}{\partial y''}$; these are approximated using the following convolution matrices applied at (x'', y'') [Hor86]:

$$\underbrace{\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}}_{8} \quad \underbrace{\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}}_{8}$$

The gradient term corresponding to the quaternion $\mathbf{q}_i$ is computed by accumulating over all terms that depend on $\mathbf{q}_i$:

$$\mathbf{G}_i = \sum_{x_i, y_i} e_{x_i, y_i} \frac{\partial e_{x_i, y_i}}{\partial \mathbf{q_i}}$$

The gradient is computed in the coordinates of image i, with respect to the quaternion $\mathbf{q}_i$ associated with image i. Similarly, the Hessian term corresponding to two adjacent images i and j is:

$$\mathbf{H}_{ij} = \sum_{x_i, y_i} \frac{\partial e_{x_i, y_i}}{\partial \mathbf{q}_i} (\frac{\partial e_{x_i, y_i}}{\partial \mathbf{q}_j})^T$$

The Hessian is computed in the coordinates of image i, with respect to the quaternions associated with images i and j, respectively. For a spherical tiling consisting of n images, the n values of $\mathbf{G}_i$ and the n^2 values of $\mathbf{H}_{ij}$ are concatenated to yield the global $1 \times 4n$ gradient $\mathbf{G}$ and the global $4n \times 4n$ Hessian $\mathbf{H}$, respectively.

In an unconstrained optimization, the increments would be computed as $-\mathbf{H}^{-1}\mathbf{G}$. Applying these increments directly to the $\mathbf{q}_i$, however, would produce non-unit quaternions which do not correspond to pure rotations. To constrain the updated quaternions to be unit vectors, we enforce the following additional constraints on the increments $\delta\mathbf{q}_i$:

$$\forall i : \mathbf{q}_i \cdot \delta\mathbf{q}_i = 0$$

Applying these $\delta\mathbf{q}_i$ moves the $\mathbf{q}_i$ tangentially to the unit four-sphere. Using Lagrange multipliers λ_i to enforce these constraints, the equation for computing the increments becomes:

$$\begin{bmatrix} \mathbf{H} & \mathbf{Q} \\ \mathbf{Q}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{Q} \\ \mathbf{\Lambda} \end{bmatrix} = - \begin{bmatrix} \mathbf{G} \\ \mathbf{0} \end{bmatrix} \tag{14}$$

where

$$\mathbf{Q} = \begin{bmatrix} \mathbf{q}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{q}_2 & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{q}_n \end{bmatrix}, \; \Delta\mathbf{Q} = \begin{bmatrix} \delta\mathbf{q}_1 \\ \delta\mathbf{q}_2 \\ \vdots \\ \delta\mathbf{q}_n \end{bmatrix}, \; \mathbf{\Lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_{4n} \end{bmatrix}$$

The optimization proceeds by solving Equation 14 for $\Delta\mathbf{Q}$ and $\mathbf{\Lambda}$ (both of dimension $1 \times 4n$), then normalizing $\mathbf{q}_i$:

$$\mathbf{q'}_i = \frac{\mathbf{q}_i + \delta\mathbf{q}_i}{\|\mathbf{q}_i + \delta\mathbf{q}_i\|}$$

To avoid matrix inversion, which is ill-conditioned when $\mathbf{H}$ is nearly singular, i.e. when the error surface is nearly flat along one or more dimensions, we use the SVD and pseudo-inverse instead [PTVF92]. Convergence is detected when the value of the objective function changes by less than some threshold (e.g., 0.1%) over one iteration.

5.2 Internal Camera Parameters

In addition to estimating orientations, the algorithm also performs an optimization on the internal camera parameters. Even though the camera was calibrated offline, in practice, there could be small variations while actually collecting images. These variations would result in incorrect warps between adjacent images, yielding misalignment artifacts similar to those arising from errors in rotation measurements.

The overall optimization alternates between a step that updates all rotations, and a step that updates internal parameters. The new parameters are computed using derivatives of $\mathbf{v}'$ in Equation 11 with respect to the camera focal length f and image principal point (c_x, c_y):

$$\frac{\partial \mathbf{v}'}{\partial f} = \left(\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{R}'\mathbf{R}^{-1}\mathbf{K}^{-1} + \mathbf{K}\mathbf{R}'\mathbf{R}^{-1} \begin{bmatrix} -\frac{1}{f^2} & 1 & \frac{c_x}{f^2} \\ 0 & -\frac{1}{f^2} & \frac{c_y}{f^2} \\ 0 & 0 & 0 \end{bmatrix} \right) \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\frac{\partial \mathbf{v}'}{\partial c_x} = \left(\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{R}'\mathbf{R}^{-1}\mathbf{K}^{-1} + \mathbf{K}\mathbf{R}'\mathbf{R}^{-1} \begin{bmatrix} 0 & 0 & -\frac{1}{f} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right) \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

The derivative with respect to c_y is similarly determined. These are used to generate derivatives of the error term $e_{x,y}$ as in Equations 12 and 13.

5.2.1 Relative Importance of Camera Parameters

Are all the internal parameters equally important for this optimization? The following simplified analysis shows that determining the focal length accurately is more important than determining the coordinates of the image principal point. This result is also confirmed empirically in Section 6. This enables the simpler calibration technique of Tsai [Tsa87] (which assumes that the image center is the same as the principal point) to be used instead of a more complex technique (e.g., [LT87]).

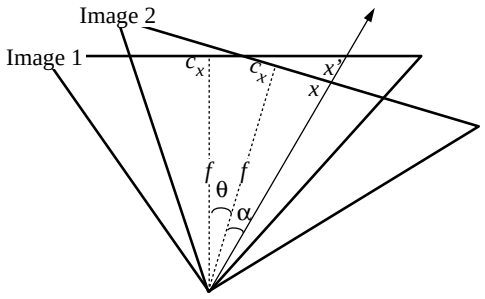

Figure 7: Rotation and camera parameters in 2-D.

Figure 7 shows two 1-D images rotated formed by rotating a "line" camera by an angle θ. In this figure, the transformation between pixel x (with offset angle α from the center) in image 2 to pixel x' in image 1 is given by:

$$x' = c_x + f\tan(\theta + \alpha) = c_x + f\frac{\tan\theta + \tan\alpha}{1 - \tan\theta\tan\alpha}$$

For small angles of rotation and small fields of view, $\tan\theta\tan\alpha \ll 1$. Thus:

$$x' \approx c_x + f\tan\theta + f\tan\alpha = c_x + f\tan\theta + x - c_x = f\tan\theta + x$$

To first order, the mapping is insensitive to the principal point. Thus the image center can be used as an initial value for optimization.

5.3 Implementation

Our implementation contains several elements that improve its performance in practice. In this section, we briefly describe them: 1) the use of both low- and high-pass filtering; 2) a modification to account for textureless images; and 3) efficient computation of the correlation and derivative terms required for the optimization.

Band-Pass Filtering

Figure 8: An image and its filtered values.

Straightforward implementation of our algorithm will fail where there are large textureless regions. High-pass filtering alone introduces many discontinuities into previously smooth image regions, corrupting the derivative computations and preventing convergence. Thus our implementation filters the images to remove textureless regions, while simultaneously preserving smoothness. We generate "band-pass" luminance values by convolving image luminance values with the derivative of a Gaussian (of 5 pixel radius).

We maintain all images at several resolutions, and optimize each resolution to completion before processing the next higher resolution image set. Each optimization phase is initialized with the rotation estimates produced by the previous phase. In practice, the chosen filter radius is comparable to the error in the initial rotation estimates, which come either from our acquisition instrumentation or from the previous (lower-resolution) optimization phase. (An alternative strategy might use the entire image pyramid simultaneously, with a filter radius proportional to the corresponding image resolution.)

Our implementation actually optimizes twice for each resolution, each time until convergence: first, on the filtered luminance values (see Figure 8), and second on the original luminance values. The first stage produces good estimates of rotation, which are used as initial estimates for the second stage, making it converge reliably. Note that the second phase is necessary for two reasons. First, since the filtered values are not invariant under planar projective transformation, the second optimization phase must use the original image values to avoid geometric bias when computing the correlation scores. Second, the second phase incorporates a correlation term, arising from matching of the textureless regions, which was suppressed in the first

phase. Section 6 demonstrates that use of band-pass values results in both faster convergence and increased avoidance of false minima.

Textureless Images

Several images in our data lack sufficient texture to determine their orientation accurately (e.g., those corresponding to cameras oriented toward the sky). As the optimization is global, such textureless images may "corrupt" the orientation estimates of other images. We avoid this problem by excluding from the optimization any images with less than a threshold fraction (80%) of textured pixels. For these images, the initial rotation estimates suffice for incorporation into subsequent processing.

Correlation and Derivative Terms

The dominant computational costs in our optimization arise from traversing (and mapping) pixels in each image, and accumulating global derivatives. Our implementation mitigates these costs with several techniques. First, only pixels that actually map to a valid pixel in the adjacent image are traversed. This is achieved by computing a boundary for the overlap region from the warp, and traversing pixels only inside the boundary. Second, warped image coordinates and their derivatives are computed incrementally as each image is traversed.

6 Mosaicing Results

This section presents both quantitative and qualitative results obtained on the dataset using the spherical optimization technique. Section 6.1 explores, using pairs of images, the sensitivity of the technique to initial values supplied to the optimization procedure. Section 6.2 presents results for full nodes in the form of spherical mosaics produced by the algorithm.

6.1 Image Pairs

In these experiments, optimal calibration parameters and the rotation between two images were obtained by optimizing initial values provided by camera calibration and physical instrumentation. Then, each of these parameters was perturbed by some amount and the optimization was rerun.

The metric used to measure the sensitivity is the number of iterations required to converge to the optimal (unperturbed) values. Optimizations that converged to a different minimum were not considered. The results tabulated below are for data obtained from two different adjacent image pairs in our dataset.

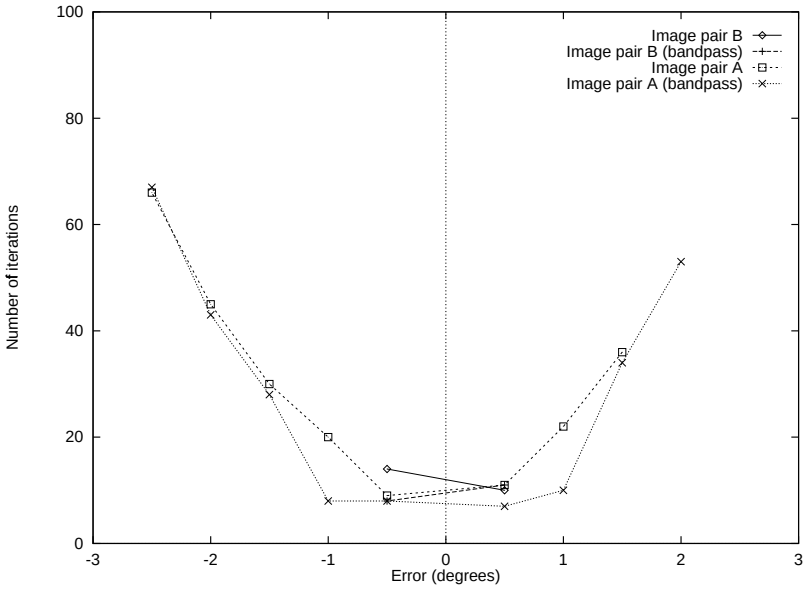

Figure 9: Convergence for different rotation errors.

Figure 9 shows results obtained by perturbing the rotation angle (about the rotation axis) by a few degrees. Note that the optimization converges within a few iterations if the perturbation is on the order of a degree (the optimization fails to converge for larger angles with image pair B). In addition, using band-pass images reduces the number of iterations required for convergence.

Figure 10 shows results obtained by perturbing the focal length. The perturbation is expressed in terms of the fraction of the correct value. Note that the algorithm is robust up to a few percent of error in the focal length. However, it fails to converge for larger errors; fairly good camera calibration is thus required to provide initial focal length estimates. Note that a focal length estimate can also be provided by enforcing the 2π constraint [McM97].

Figure 11 shows convergence behavior under perturbation of the principal point (only results for c_x are shown here). The optimization is quite robust,

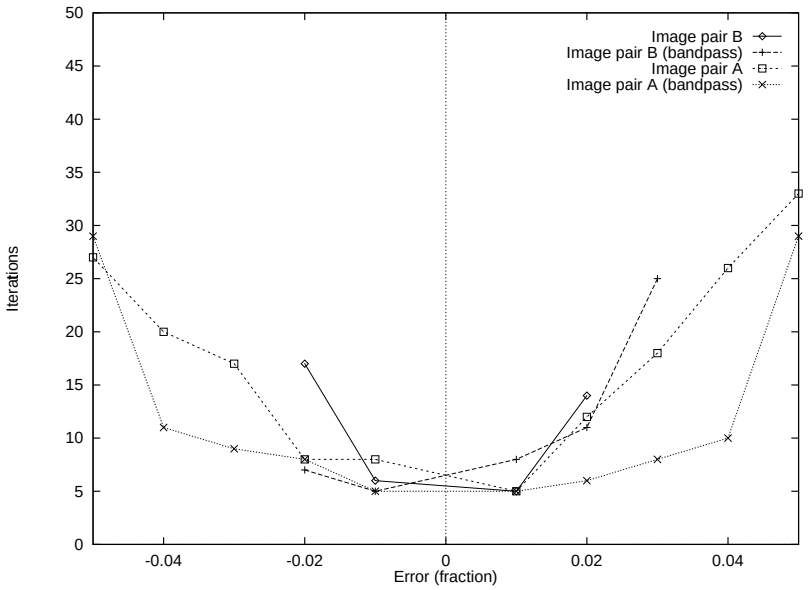

Figure 10: Convergence for different focal length errors.

converging to the optimal values even with initial errors of ten or twenty percent. This confirms the validity of the simple theoretical model in Section 5.2.1. Also, note that the convergence behavior need not be symmetric between positive and negative errors: images are asymmetric in general, and shifts in the transformation by a few pixels to the "right" can yield very different results than shifts to the "left".

6.2 Node Optimization

Figure 12 shows convergence rates for images from two nodes. The graph plots total pixel luminance SSD error across all images in a node as the optimization proceeds. Note that the error decreases faster away from the optimum; this is consistent with the *quadratic* convergence rates guaranteed by LM optimization near the minimum [Sca85]. Also, due to slight errors in image formation and use of discrete sampling to estimate the error function, the optimization converges to a non-zero value. Despite this, the mosaics generated using the optimal values do not exhibit any visible misalignment artifacts (see below), providing evidence that the images are registered accurately with respect to each other.

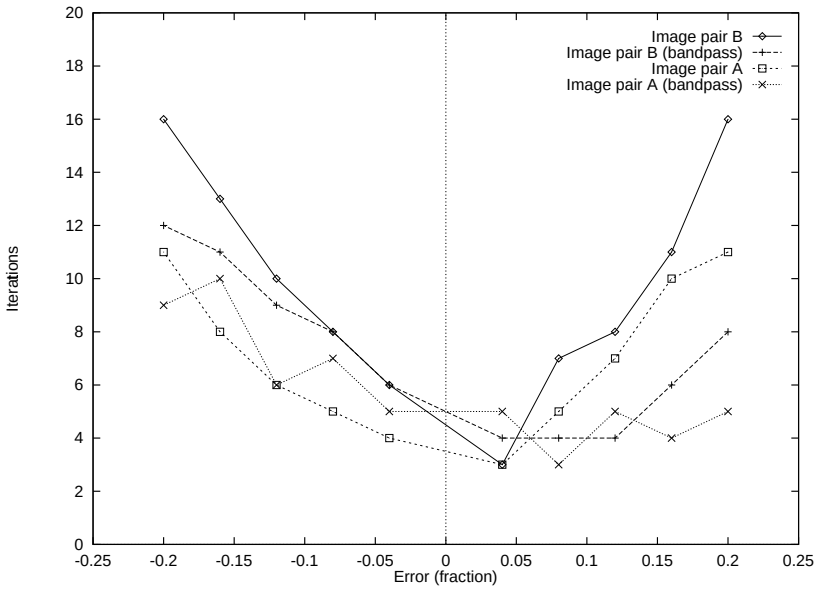

Figure 11: Convergence rates under perturbation of the principal point.

In a batch process, the algorithm successfully processed all (close to four thousand) input images[2] from eighty-one nodes, requiring about twenty minutes of processing per node for 381×253 pixel images and about two hours of processing per node for 762×506 pixel images (both on a 150MHz R10000 SGI O2 workstation). Internal camera parameters converged to values that differed by less than one percent across all nodes[3].

The relative rotations and camera parameters can be used to blend all node images into a single seamless image, without "blurring" or "ghosting" artifacts. Each pixel is blended with a weight inversely proportional to its distance from the image center. Two mosaics resulting from this procedure are shown in Figure 13.

To illustrate the advantage of global optimization, we modified the optimization to use pair-wise correlation on an adjacency tree, i.e., an adjacency map containing no cycles. (This is a generalization of omitting the correlation term between image $n-1$ and image 0 in a cylindrical panorama of n images.) The resulting mosaic has several evident defects (Figure 14).

[2]A few input images were unusable due to sun flare and/or CCD oversaturation.

[3]For 762×506 images, focal length $f = 1015$ pixels; principal point $c_x, c_y = 397, 261$.

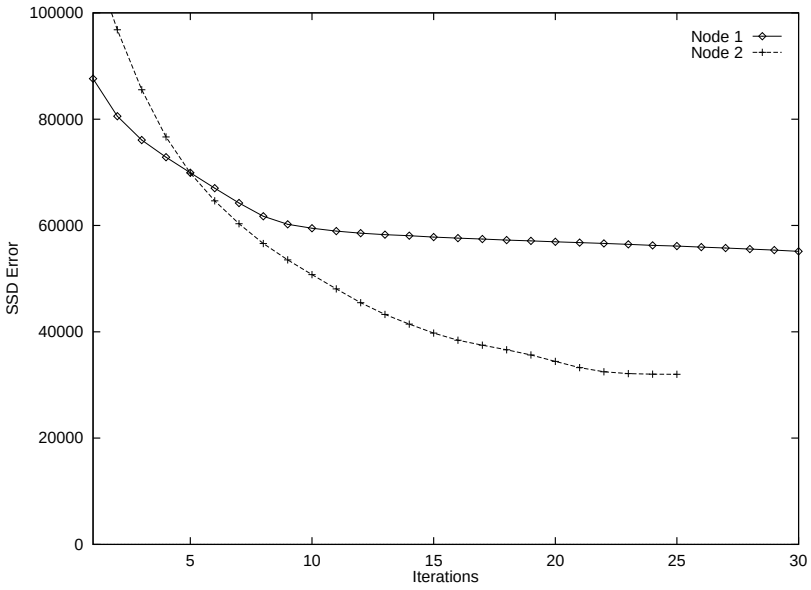

Figure 12: Convergence for different rotation errors.

6.3 Mosaic Representations

We use three alternative visual representations for mosaics in practice. Each resamples the source imagery onto a three-dimensional shape for purposes of direct display, texture mapping, etc. The first representation is the familiar sphere. The second representation (Figure 13) projects the sphere onto a cylindrical surface by projecting each point (ϕ, θ) on the sphere to $(\phi, \sin\theta)$. This cylinder is then unrolled onto a plane for display. This projection preserves area on the sphere [Pae90], so exhibits less distortion than the more commonly used (ϕ, θ) projection. However, it does distort straight edges from the source images into curves in the projected image.

Our third representation, the cubical environment map used in computer graphics (Figure 15) ameliorates this problem. Here, the mosaic consists of six images corresponding to each face of the cube. The drawback of this representation, of course, is that edges spanning two cube faces are discontinuous. However, since most edges are small relative to the 90° field-of-view provided by each cube face, this is not a problem in practice.

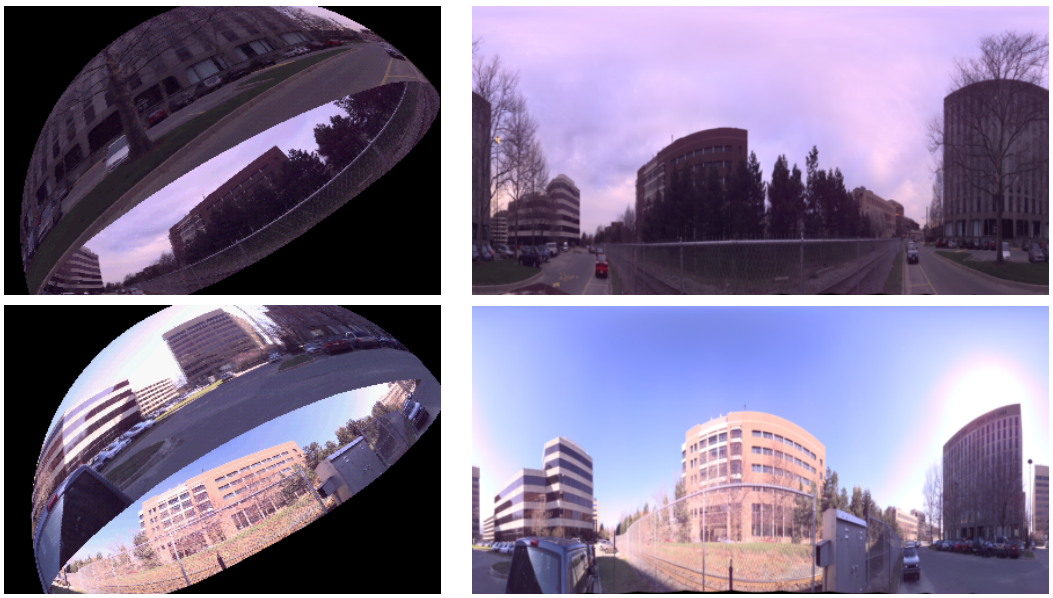

Figure 13: Two typical mosaics, shown as spheres and cylinders.

7 Conclusion

We described two methods to recover relative rotations and internal camera parameters for the set of images acquired from a common optical center. The first is a closed-form solution using eigen-vectors of 8-parameter warps. This method is theoretically elegant, but yields quantitatively inaccurate results. The second method solves this problem by computing rotations and internal parameters directly from image-space correlation using a global optimization technique. The results, demonstrated for over eighty nodes, demonstrate that this method accurately recovers relative rotations about a common optical center.

There are several benefits in performing the spherical mosaicing optimization. First, it provides robust automatic estimation of internal camera parameters as a by-product. Second, it produces an image with an effectively super-hemispherical field of view, eliminating the ambiguity between camera translation and camera rotation found in narrow field-of-view images. This image can be of any desired effective resolution, subject to the choice of optics and number of raw images that are composited. Third, spherical mosaicing allows the resulting mosaic to be treated as a rigid, composite image. Thus

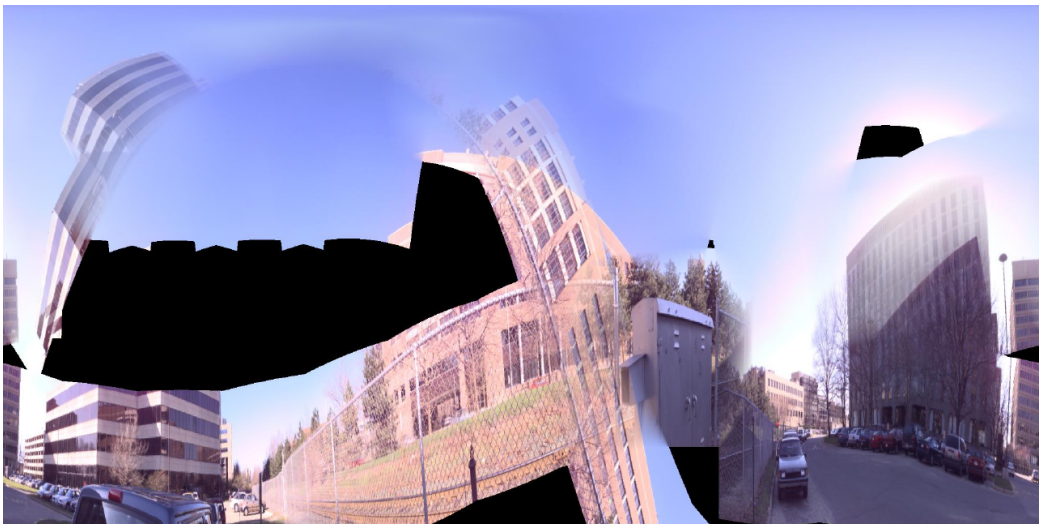

Figure 14: This mosaic, computed with pair-wise optimization over an adjacency tree, exhibits numerous defects. The mosaic computed by our method, using the same image data, is shown at the bottom of Figure 13.

Figure 15: Four faces of a cubical projection for two nodes.

it reduces, by a factor linear in the number of images per node (here, about fifty), the number of degrees of freedom when determining global position and orientation by subsequent optimization.

Acknowledgments

We are grateful to the anonymous reviewers for their suggestions, and to Neel Master for his help in preparing the figures.

References

[ATar] Matthew Antone and Seth Teller. Automatic recovery of relative camera rotations for urban scenes. In *CVPR*, 2000 (to appear).

[ATon] Matthew Antone and Seth Teller. Automatic recovery of relative camera translations in urban scenes. In *Workshop on 3D Structure from Multiple Images*, 2000 (in submission).

[CMT98] Satyan Coorg, Neel Master, and Seth Teller. Acquisition of a large pose-mosaic dataset. In *CVPR '98*, pages 872–878, 1998.

[Coo98] Satyan Coorg. *Pose Imagery and Automated Three-Dimensional Modeling of Urban Environments*. PhD thesis, MIT, 1998.

[CT99] Satyan Coorg and Seth Teller. Extracting textured vertical facades from controlled close-range imagery. In *Proceedings CVPR '99*, pages 625–632, June 1999.

[Fau93] Olivier Faugeras. *Three-Dimensional Computer Vision*. MIT Press, 1993.

[Har97] Richard Hartley. Self-calibration of stationary cameras. *IJCV*, 22(1):5–23, 1997.

[Hec89] Paul Heckbert. Fundamentals of texture mapping and image warping. Technical Report UCB/CSD 89/516, CS Division, UC Berkeley, 1989.

[Hor86] Berthold Klaus Paul Horn. *Robot Vision*. MIT Press, Cambridge, MA, 1986.

[Hor87] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4), April 1987.

[Hor91] Berthold K. P. Horn. Relative orientation revisited. *Journal of the Optical Society of America A*, 8(10):1630–1638, October 1991.

[HS98] S. Hsu and H. Sawhney. Influence of global constraints and lens distortion on pose and appearance recovery from a purely rotating camera. In *Workshop on Applications of Computer Vision*, pages 154–159, October 1998.

[LT87] R. Lenz and R. Tsai. Techniques for calibration of the scale factor and image center for high accuracy 3D machine vision metrology. In *Proc. IEEE International Conf. on Robotics and Automation*, 1987.

[MB95] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *SIGGRAPH '95 Conference Proceedings*, pages 39–46, August 1995.

[McM97] Leonard McMillan. *An Image-Based Approach to Three-Dimensional Computer Graphics*. PhD thesis, Dept. of Computer Science, Univ. of North Carolina (Chapel Hill), 1997.

[Pae90] Alan Paeth. Digital cartography for computer graphics. In Andrew Glassner, editor, *Graphics Gems*, pages 307–320. AP Professional, 1990.

[PTVF92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing (2nd ed.)*. Cambridge University Press, Cambridge, 1992.

[Sca85] L.E. Scales. *Introduction to Non-Linear Optimization*. Springer-Verlag, 1985.

[SHK98] H. Sawhney, S. Hsu, and R. Kumar. Robust video mosaicing through topology inference and local to global alignment. In *European Conference on Computer Vision*, pages 103–119, 1998.

[SK99] H.S. Sawhney and R. Kumar. True multi-image alignment and its application to mosaicing and lens distortion correction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(3):235–243, March 1999.

[SS97] Richard Szeliski and Harry Shum. Creating full-view panoramic mosaics and texture-mapped 3D models. In *SIGGRAPH '97 Conference Proceedings*, pages 251–258, August 1997.

[SS98] H.-Y. Shum and R. Szeliski. Construction and refinement of panoramic mosaics with global and local alignment. In *Proceedings of Sixth ICCV*, pages 953–958, January 1998.

[Ste95] G.P. Stein. Accurate internal camera calibration using rotation, with analysis of sources of error. In *ICCV95*, pages 230–236, 1995.

[Str88] G. Strang. *Linear algebra and its applications*. Harcourt Brace Jovanovich, 1988.

[Sze96] Richard Szeliski. Video mosaics for virtual environments. *IEEE Computer Graphics and Applications*, 16(2):22–30, March 1996.

[Tel97] Seth Teller. Automatic acquisition of hierarchical, textured 3D geometric models of urban environments: Project plan. In *Proceedings of the Image Understanding Workshop*, 1997.

[Tsa87] 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*, RA-3(4), August 1987.

[WI95] M. Wheeler and K. Ikeuchi. Iterative estimation of rotation and translation using the quaternion. Technical Report CMU-CS-95-215, Carnegie Mellon University, 1995.

[ZFD97] I. Zoghiami, O.P. Faugeras, and R. Deriche. Using geometric corners to build a 2D mosaic from a set of images. In *CVPR97*, pages 420–425, 1997.