Spherical Mosaics with Quaternions and Dense Correlation

First Progress Report

Submitted in partial fulfillment of the requirements
for the degree of

Ph.D.

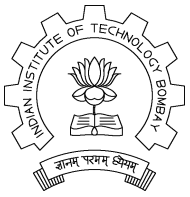
by

Rhushabh Goradia

Roll No: 04405002

under the guidance of

Prof. Sharat Chandran



Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Mumbai

August 28, 2005

Acknowledgments

I would like to thank my guide, **Prof. Sharat Chandran**, for his valuable guidance, consistent motivation and directions he has fed into my work.

—Rhushabh Goradia

Contents

List of Figures

Abstract

The need for mosaicing arises when we want to stitch two or more images together so as to view them as a single continuous image. There are various ways to construct mosaics of images, one of them being *Spherical Mosaics*. As the name suggests, 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.

In this report we focus on an algorithm for constructing spherical mosaics from a collection of images taken from a common optical center. 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 form the inputs for the algorithm. The algorithm's output is a rotation relating each image to the base image and revised estimates of the camera's internal parameters. We also compare two different optimization techniques (local and global) and show why global optimization technique is much more superior to the local one.

We study the details of this algorithm and also provide the details of our work in this area which overcomes some limitations of the algorithm as described in [1].

Chapter 1

Introduction

1.1 What is Spherical Mosaicing ?

The need for mosaicing arises when we want to stitch two or more images together so as to view them as a single continuous image. There are various ways to construct mosaics of images, one of them being *Spherical Mosaics* [1]. As the name suggests, 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. Images shown here describes a somewhat hemispherical point of view but it can be extended to full spherical arrangements of images.

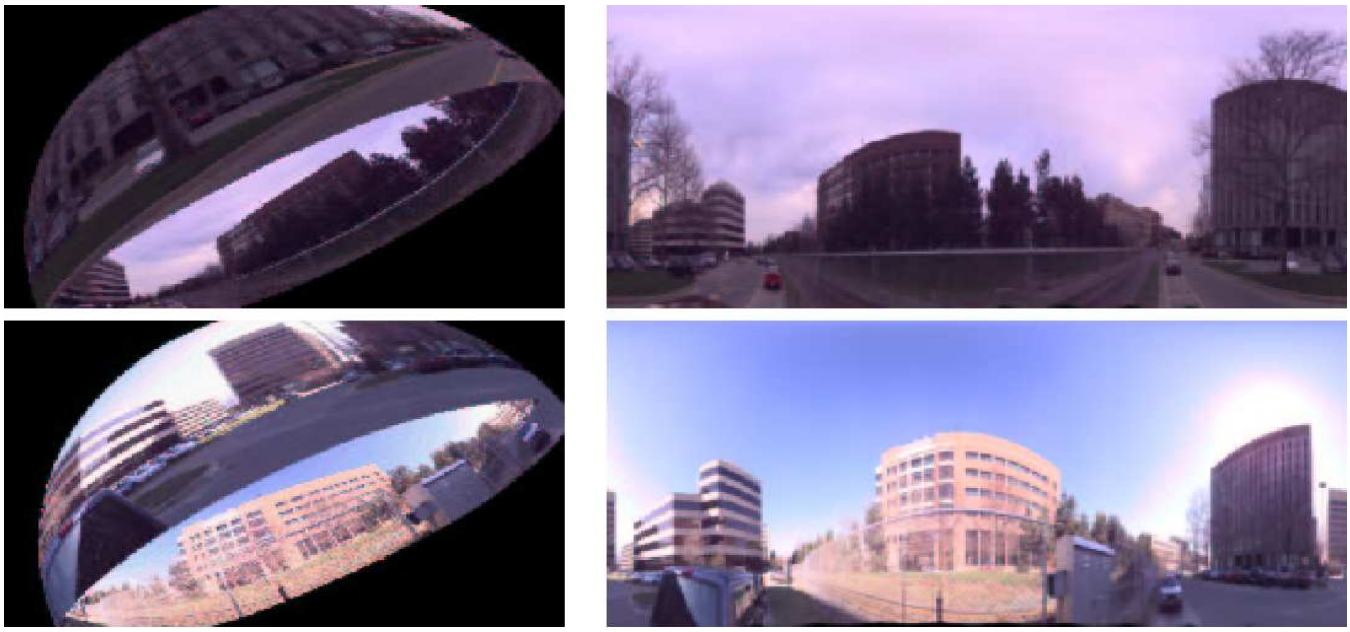


Figure 1.1: Two typical mosaics shown as sphere and cylinder

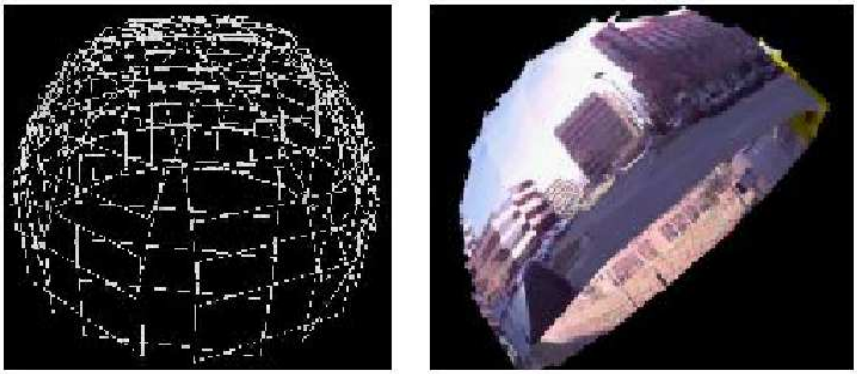


Figure 1.2: The roughly hemispherical tiling for a node of the dataset

1.2 A Brief Review of the Basic Concepts

Let us review some of the basic concepts useful in making us understand the spherical mosaicing technique.

1.2.1 Image formation by perspective projection

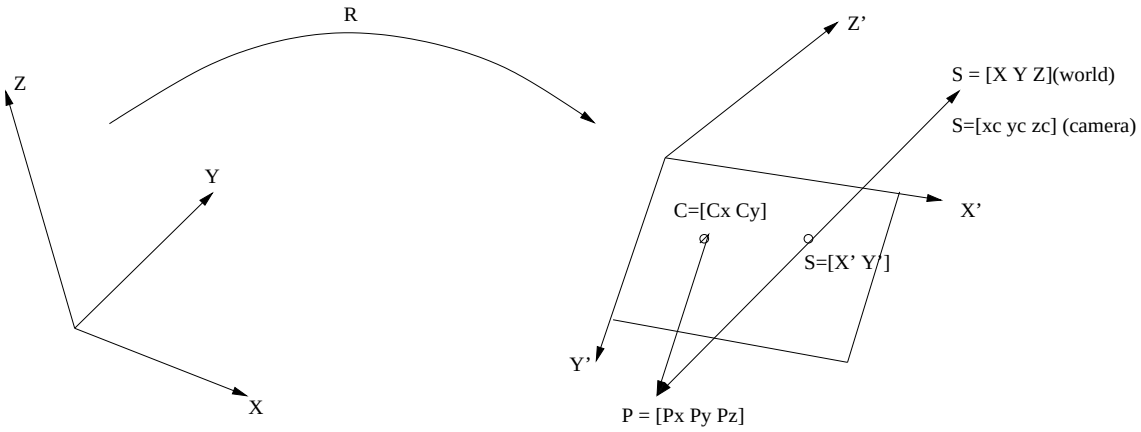


Figure 1.3: Overview of Perspective Projection

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

Now,

$$s_c = R(s - p)$$

where $s_c = [x\ y\ z]$ are the coordinates of s in the camera's coordinate system. Now we scale the x and y co-ordinates by depth as

$$x_I = \frac{x_c}{z_c}\ y_I = \frac{y_c}{z_c}$$

After scaling it, we now apply the intrinsic camera parameters to complete the transformation. The final values are:

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

Thus, the entire transformation can be represented in a matrix form using projective geometry [2] as :

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \cong KM \begin{bmatrix} R & -R_p \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{1.1}$$

where **K**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:

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

1.2.2 Introduction to Quaternions

Let us have a quick review of what quaternions are, why are they needed and their advantages [9] and [5].

1.2.2.1 Why Quaternions ?

A common problem in computer vision is solving for rigid body motions or poses consisting of a rotation and translation in 3D space. For example, given a set of points x_i and correspondences p_i, it is often of interest to compute the 3×3 rotation matrix R and 3-vector translation t such that

$$Rx_i + t = p_i$$

Although this system of equations is essentially linear, a number of problems arise when formulating solutions that account for the non-linear constraints on the components of R. The constraints arise from using nine values of rotation matrix R to represent three independent variables of 3D rotation. The rotation matrix is constrained to be orthogonal which is satisfied when $R^T R = I$ (i.e., the rows and columns are orthonormal). Also, the rotation must not be a reflection; this is satisfied when the determinant is 1 ($\mid R \mid = 1$).

A number of techniques have been developed to deal with this added complexity. One of the most convenient is the quaternions representation.

Here are some of the advantages and mathematical niceties of the quaternion representation of rotation.

- Quaternions can be composed/multiplied in a straightforward manner to accumulate the effects of composed rotations.
- The inverse of a quaternion (specifying the inverse rotation) is obtained by simply negating 3 components of the quaternion vector.
- The rotation between two rotations can be computed by multiplying one quaternion with the inverse of the other.
- One can easily transform a quaternion into an axis-and-angle representation. Using this and the previous item, one can compute a rotational distance metric between two rotations — the angle of rotation between them.
- Quaternions can be easily transformed to a 3×3 rotation matrix for efficient computation when rotating vectors.
- With the quaternion representation, the rotation can be solved for in closed form when correspondences between three-dimensional point sets are available.
- Maintaining the constraints (orthogonal with unit determinant) of rotation is made simple with quaternions by standard vector normalization.
- The unit quaternion representing the best rotation is the eigenvector associated with the most positive eigenvalue of a symmetric 4×4 matrix. The elements of this matrix are combinations of sums of products of corresponding coordinates of the points.
- Suppose that we are given the coordinates of a number of points as measured in two different Cartesian coordinate systems. The photogrammetric problem of recovering the transformation between the two systems from these measurements is referred

to as that of *absolute orientation.* Let us call the two coordinate systems "left" and "right." A desirable property of a solution method is that, when applied to the problem of finding the best transformation from the right to the left system, it gives the exact inverse of the best transformation from the left to the right system. Symmetry is guaranteed when one uses unit quaternions to represent rotation.

- It is much simpler to enforce the constraint that a quaternion have unit magnitude than it is to ensure that a matrix is orthonormal.

1.2.2.2 What are Quaternions ?

The quaternion $\mathbf{q}$ is a four vector $[q_x,\ q_y,\ q_z,\ q_0]^T$ which is often considered as a three-vector $\boldsymbol{u} = [q_x,\ q_y,\ q_z]^T$ and a scalar $s = q_0$. Also it has the property that ${q_0}^2 + {q_x}^2 + {q_y}^2 + {q_z}^2 = 1$. Quaternion $\mathbf{q}$ is generally referred to as $[u, s]^T$ for notational simplicity.

The dot product and vector norm for quaternions is defined as usual

$$q_1 \cdot q_2 = u_1 \cdot u_2 + s_1 s_2$$
$$\mid q \mid = (q \cdot q)^{-\frac{1}{2}}$$

Multiplication is defined over quaternions as

$$q_1 q_2 = [[s_1 u_2 \ + \ s_2 u_1 \ + \ u_1 \times u_2],\ s_1 s_2 \ - \ u_1 \cdot u_2]$$

The complex conjugate of a quaternion is defined by negating the vector component and is denoted $\overline{q} = [-u, s]^T$. The complex conjugate of a unit quaternion, $|q| = 1$, is the inverse of the quaternion with respect to multiplication, i.e., $q\overline{q} = q_I$, where $q_I = [0, 0, 0, 1]^T$. Also $qq_I = q_I q = q$ which is why q_I is referred as the identity quaternion.

1.2.2.3 Rotations as Quaternions

A unit quaternion $\mathbf{q}$ can be used to perform a rigid rotation of a vector $\mathbf{x} = [x, y, z]^T$ by two quaternion multiplications

$$x' = q \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix} \overline{q},$$

where the scalar component of $\mathbf{x}$ is simply set to zero. Observe that quaternion multiplication is not commutative; this is consistent with the fact that general three-dimensional rotations do not commute; however, quaternion multiplication is associative and distributive.

Working from this definition of quaternion rotation, one can derive a formula for the corresponding orthogonal (Euclidean) 3×3 rotation matrix from a unit quaternion

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

The subscript u in R_u is used to denote that this is the rotation matrix when given a unit quaternion. Given an arbitrary quaternion, R_u would no longer be unitary but rather a scaled rotation matrix.

1.2.3 2-D Projective Transformations

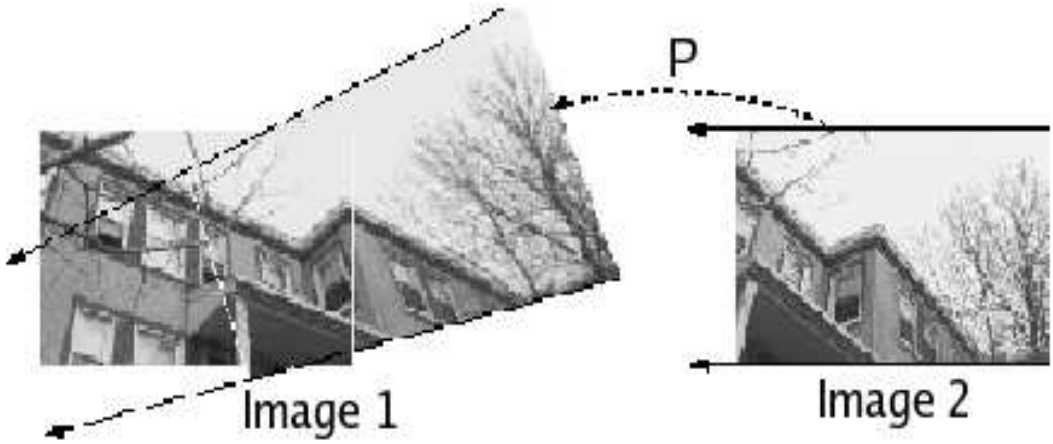


Figure 1.4: Transforming pixels from image 1 to space of image 2

Figure 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 [4]. As stated in [4], *Given a pair of images taken by cameras with the same internal parameters from the same location, then there is a projective transformation P taking one image from the other. Furthermore, P is of the form $P = KRK^{-1}$ where R is a rotation matrix and K is the calibration matrix. Unlike simpler 2-D transformations (translation, rotation, affine), the projective transformation does not preserve parallel lines. This is evident in the above figure, where the lines bounding image 1 intersect after transformation.*

As depth effects do not occur across two images taken from the same optical center [4, 7], the general perspective projection (Equation 3.1) simplifies to:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \cong KR \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{1.2}$$

Inverting Equation 4.1 yields:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \cong R^{-1}K^{-1} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \tag{1.3}$$

Above equation converts image co-ordinates to 3-D. Thus pixel coordinates in image 2 can be obtained by projecting back into image 2's space using Equation 4.1:

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \cong KR_2R_1^{-1}K^{-1} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \tag{1.4}$$

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 :

$$P = KR_2R_1^{-1}K^{-1} \tag{1.5}$$

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*.

1.3 Organization of Report

In chapter one, we had a brief overview of what does *Spherical Mosaicing* stand for. We also reviewed some of the basic concepts like image formation via perspective projection, use of quaternions as a form of representation for rotation along with some of its nice properties[9] and [5], and 2-D projective transformations and methods to compute them, which are required to understand the Spherical Mosaicing optimization algorithm. In chapter two, we will describe in detail the Spherical Mosaicing algorithm along with a couple of optimization techniques, viz. Local Optimization technique and Global Optimization Technique. The former although being theoratically elegant, yields qualitatively inaccurate results. We will see how the latter solves this problem by computing rotations and internal parameters directly from the image-space correlation using Global Optimization. In chapter three, we will have a look at the implementation details of the algorithm along with some modifications in terms of getting the initial rotation estimates as well as the optimization technique applied. In short, it tells us what our contributions and results are. Chapter four gives a brief idea of the future work to be carried out.

Chapter 2

Spherical Mosaicing

2.1 Introduction

Spherical Mosaicing algorithm 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. 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 form the inputs for the algorithm. The algorithm's output is a rotation relating each image to the base image and revised estimates of the camera's internal parameters, thereby computing a spherical mosaic, a composite of all images corresponding to a single node.

[1] uses the camera instrument to annotate each acquired image with an estimate of absolute 6-DOF pose (exterior orientation) — 3 DOF of position, and 3 DOF of orientation for the acquiring camera. Thus the 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. But these estimates are not accurate and calls for some *pose-refinement* algorithm. How to optimize and refine these estimates is described in the following sections.

2.2 Local Optimization Technique and its Demerits

2.2.1 The Technique

This section presents a novel idea to compute the warps [7]. The idea is to compute a warp that (locally) minimizes image-space error by using nonlinear optimization after having the initial estimates. 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 **L1** and **L2** of images 1 and 2, respectively:

$$E_{12} = \sum_{x_1,y_1}(L_1(x_1, y_1) - L_2(P(x_1, y_1)))$$

The SSD form is well suited for numerical optimization, as only first order derivatives are required to compute updated values for the iteration.

A single error term is defined as:

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

The optimization consists of analytically determining derivatives of the above term with respect to **P**. It use Levenberg-Marquardt (LM) non-linear optimization technique for the same. The derivative $\frac{\partial e_{x_1,y_1}}{\partial P}$ is expressed as an 8-component vector consisting of derivatives w.r.t each entry of P.

The following figures shows images in which the first image represents image after the initial estimates of the warp.Note that incorrect transformations arising from inaccurate estimates of camera pose result in mismatches between pixels, causing the blurring and ghosting as seen. The second image is the image after optimization.

Figure 2.1: 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.

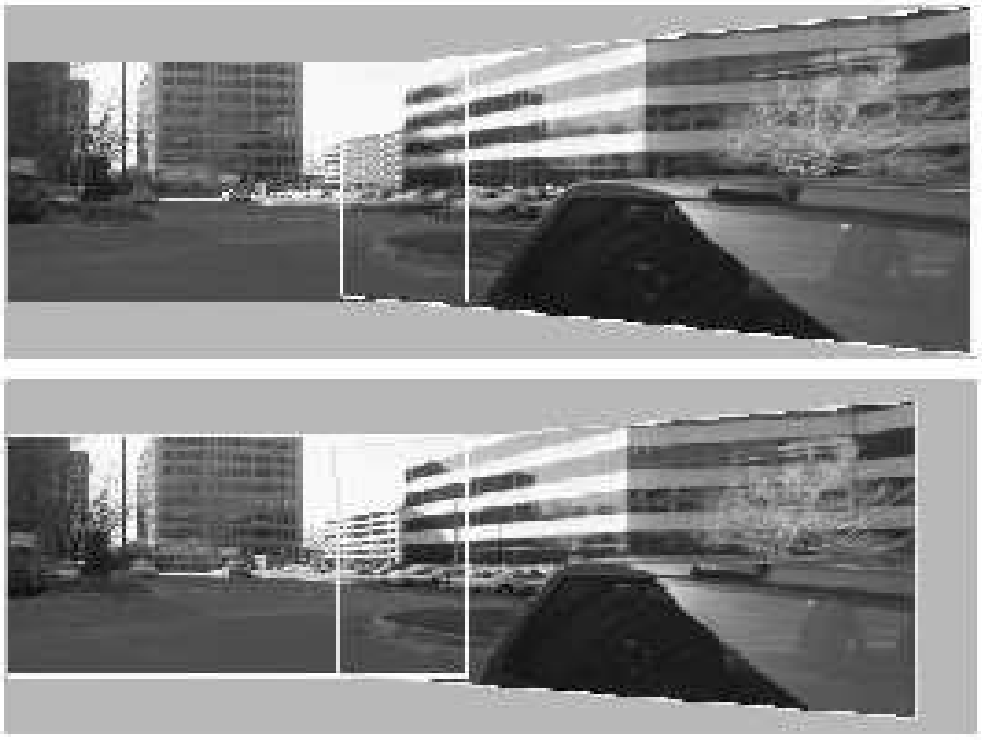


Figure 2.2: The projective warp between two images before and after optimization

2.2.2 Demerits of above discussed Technique

From empirical evidences it has been seen that this technique of estimating rotations seems to be inaccurate. This can be observed in form of large gaps between image borders although within the region of overlap, the two warps appear iden tical.

Figure 2.3: Image after local optimization

This **problem is inherent in the warp solution itself.** This calls for imposing a rotational constraint during the op timization to obtain quantitatively correct results and remove the gaps found in the image resulting from the above discusse d technique. Also, more fundamentally, relying on local pairwise warps to compute global quantities can lead to inconsistenc ies in the computed internal parameters and rotations, and more gaps in the mosaiced images. Thus, we need to go for a globa l optimization procedure.

2.3 Global Optimization Technique

2.3.1 Introduction

The optimization described in this section 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 this representation have the property that the aggregate rotation along any cycle in the image adjacency map is the identity. In this manner, this representation avoids the possibility of gaps arising from inconsistent pairwise estimates. This method is as described in [1].

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.

2.3.2 Optimization of Rotational Parameters

The algorithm minimizes a **Global Error Function**:

$$\boxed{E = \sum_{i,j} E_{ij} + E_{ji}}$$

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

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

and 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. As in the pairwise warp estimation, this function is minimized by computing *derivatives* w.r.t each orientation and using **LM Nonlinear Optimization** starting from the initial orientations.

The various steps of computation are:

- The single error term for images i and j is given by:

$$e^2_{x,y} = (L_i(x,y) - L_j(x",y"))^2 \quad \text{with} \quad x" = \frac{x'}{z'} \;\; y" = \frac{y'}{z'}$$

and

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

- The derivatives for the above error term are calculated as follows:

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

where

$$v = K^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Then, the *derivative* of the term $e_{x,y}$ w.r.t the quaternion $\mathbf{q}$ is given by:

$$\frac{\partial x"}{\partial q} = \frac{\frac{\partial x'}{\partial q} - x" \frac{\partial z'}{\partial q}}{z'} \quad \frac{\partial y"}{\partial q} = \frac{\frac{\partial y'}{\partial q} - y" \frac{\partial z'}{\partial q}}{z'}$$

- and finally we have ...

$$\frac{\partial e_{x,y}}{\partial q} = \frac{\partial L_j}{\partial x"} \frac{\partial x"}{\partial q} + \frac{\partial L_j}{\partial y"} \frac{\partial y"}{\partial q}$$

- These *derivatives* $\frac{\partial L_j}{\partial x"}$ and $\frac{\partial L_j}{\partial y"}$ are approximated using the following convolution matrices applied at $(x", y")$

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

The **Gradient** term corresponding to the quaternion q_i is computed over all terms that depend on q_i:

$$G_i = \sum_{x_i,y_i} e_{x_i,y_i} \frac{\partial e_{x_i,y_i}}{\partial q_i}$$

It is computed in the coordinates of image i, w.r.t the quaternion q_i.Similarly, the **Hessian** term corresponding to two adjacent images i and j is:

$$H_{ij} = \sum_{x_i,y_i} \frac{\partial e_{x_i,y_i}}{\partial q_i} \left(\frac{\partial e_{x_i,y_i}}{\partial q_i}\right)^T$$

Now, in an unconstrained optimization, the increments would be computed as $-H^{-1}G$. Applying these increments directly to the 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

$$\delta q_i \; : \; \forall i : q_i \cdot \delta q_i = 0$$

Using **Lagrange multipliers** λ_i to enforce these constraints, the equation for computing the increments becomes:

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

where

$$Q = \begin{bmatrix} q_1 & 0 & \dots & 0 \\ 0 & q_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & q_n \end{bmatrix}, \Delta Q = \begin{bmatrix} \delta q_1 \\ \delta q_2 \\ \vdots \\ \delta q_n \end{bmatrix}, \Lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_{4n} \end{bmatrix}$$

The optimization solves the above equation for ΔQ and Λ. Convergence is detected when the value of the objective function changes by less than some threshold.

2.3.3 Optimization of Internal Camera Parameters

In addition to estimating orientations, the algorithm also performs an optimization on the internal camera parameters [1]. The overall optimization alternates between a step that updates all rotations, and a step that updates internal parameters of the camera. The new parameters are computed using derivatives of v' w.r.t the camera focal length $\mathbf{f}$ and image principal point (c_x, c_y) as shown:

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

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

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

And here is the optimized image...

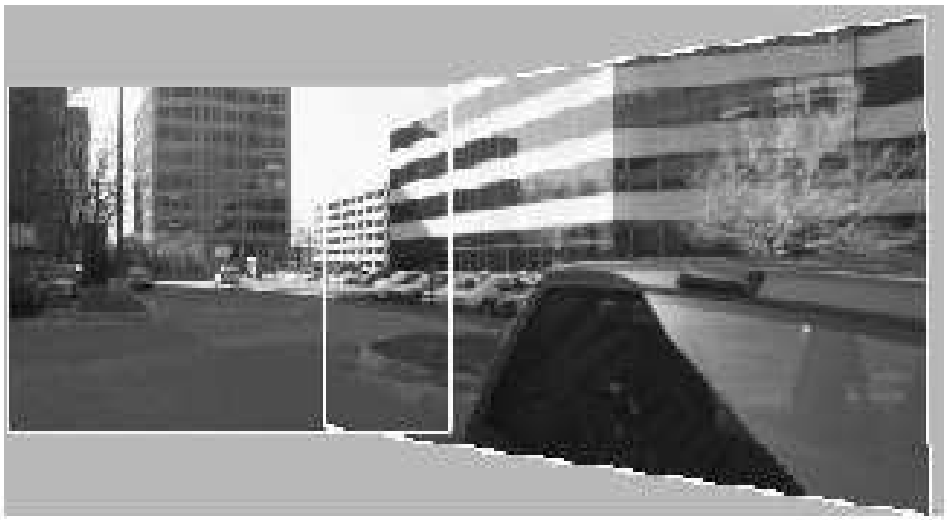


Figure 2.4: Projective warp between two images after direct optimization

We see that the large gaps between image borders, which were initially present, have reduced considerably.

2.4 Relative Importance of Camera Parameters

Are all the internal parameters equally important for this optimization? A simplified analysis shown below tells us that determining the **focal length** accurately is more important than determining the coordinates of the image principal point [1] and [8].

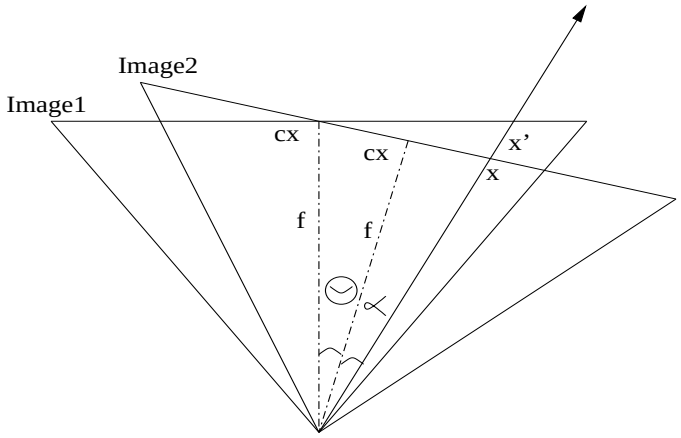


Figure 2.5: Rotation and camera parameters in 2-D

Figure shows two 1-D images rotated by an angle θ.Here the transformation between pixel x (with offset angle α from the center) in *image 2* to pixel x' in *image 1* is given by:

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

Now $\tan\theta\tan\alpha \ll 1$ for small angles.Thus

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

Thus we see, to 1^{st} order, the mapping is **insensitive to the principal point (c_x, c_y) and more sensitive to the focal length f**. Thus the image center can be used as an initial value for optimization !!!

So we have seen how a global optimization function and representing rotations as quaternions helps in reducing the gaps between images and gives a very much acceptable composite image as its output.

2.5 Summary of Above Discussion

This chapter described two methods to recover relative rotations and internal camera parameters for the set of images acquired from a common optical center.

The first method is a closed-form solution using eigen-vectors of 8-parameter warps. This method is theoretically elegant, but yields quantitatively inaccurate results as it just performs *local optimization*. The second method solves this problem by computing rotations and internal parameters directly from image-space correlation using a *global optimization* technique.We saw that this method gives better and accurate results than the former one.

Overall, spherical mosaicing allows the resulting mosaic to be treated as a rigid, composite image and provides a huge field of view. If proper optimization of the estimates is performed, better results are delivered.

Chapter 3

Our Approach

3.1 Contributions and Results

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 form the inputs for the algorithm described in [1]. It uses the camera instrument to annotate each acquired image with an estimate of absolute 6-DOF pose (exterior orientation) — 3 DOF of position, and 3 DOF of orientation for the acquiring camera. Thus the 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.

In our algorithm implementation, we use the camera instrument only to provide the partially overlapping images. The adjacency map relating the images is given as a manual input while the initial estimates of rotation and internal camera parameters are automatically calculated from input images.

3.1.1 Calculation of initial estimates for rotation relating the input images

Instead of having the initial estimates for rotation relating the images provided by the camera instrument, we calculated it from the partially overlapping images taken as input.

It is a 4 step procedure as described below:

1. A corner detector is applied to each image to extract the high curvature points, defined as 'Corners', as described in [3].

2. A classical corelation technique is then used to establish matching candidates between the two adjacent images, as described in [10].

3. We then compute the initial warp matrix using the 4 point correspondence method [7].

4. Finally we recover the initial rotation estimates from the computed warp [1].

Let us see each of these steps in more detail.

3.1.1.1 Corner Point Detection

This section gives you a brief overview of how to extract the corners from the input image.

1. Compute the image gradient over the entire image I

2. For each image point p:

   (a) form the matrix C, where,

$$C = \begin{bmatrix} \sum E_x^2 & \sum E_x E_y \\ \sum E_x E_y & \sum E_y^2 \end{bmatrix}$$

   over a $(2N+1) \times (2N+1)$ neighborhood Q of p.

   (b) compute λ_2, the smaller eigenvalue of C.

   (c) if $\lambda_2 > \Upsilon$, where Υ is the threshold on λ_2, save the coordinates of p into a list, L

3. Sort L in decreasing order of λ_2.

4. Scanning the sorted list top to bottom: for each current point, p, delete all points appearing further on in the list which belong to the neighborhood of p.

The output is a list of feature points for which $\lambda_2 > \Upsilon$ and whose neighborhoods do not overlap.

Thus we saw, the algorithm has two main parameters: the threshold Υ, and the size of the neighborhood, $(2N+1)$. The threshold can be estimated from the histogram of λ_2, as the latter has an obvious valley near zero. A sample histogram is as shown,

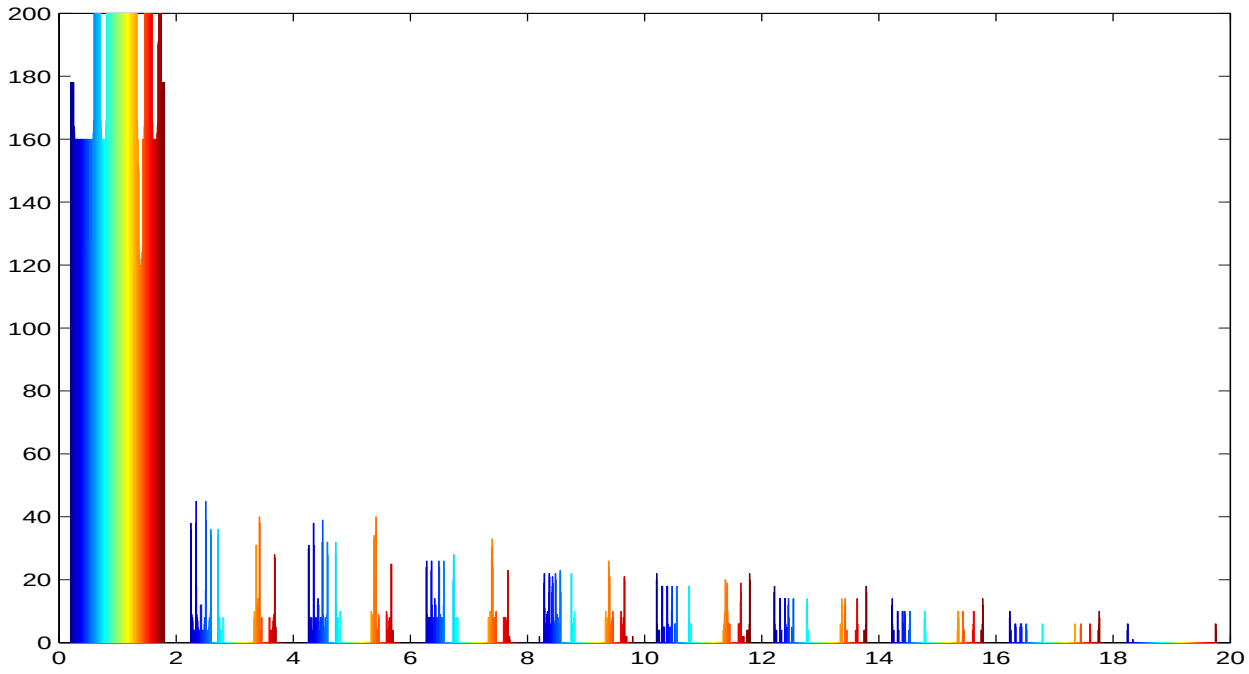


Figure 3.1: Histogram of λ_2 values across a sample image

Unfortunately, there is no simple criterion for the estimation of the optimal size of the neighborhood. In our implementation we have chosen $N = 10$. A sample output of our implemented code is shown below:

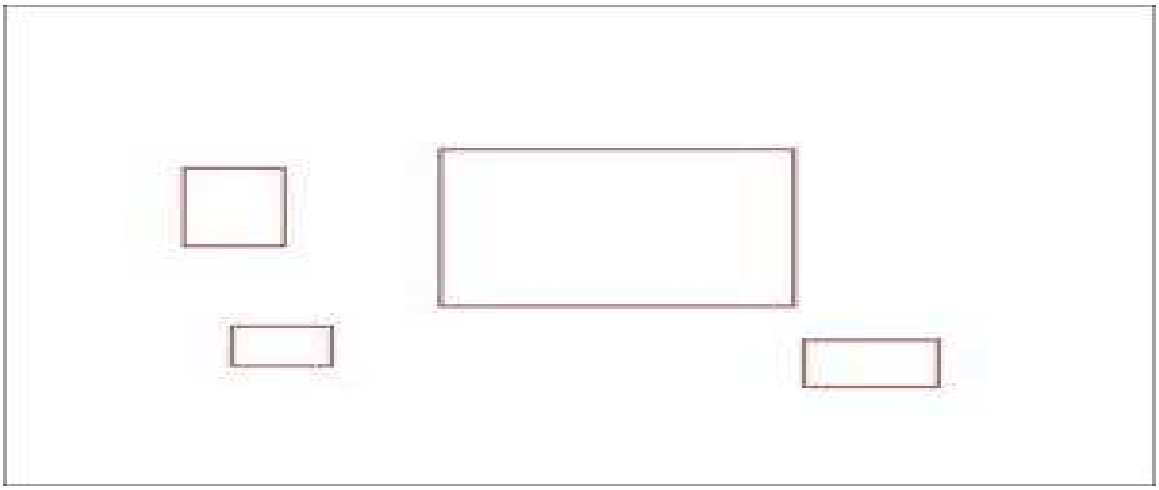


Figure 3.2: Input sample image

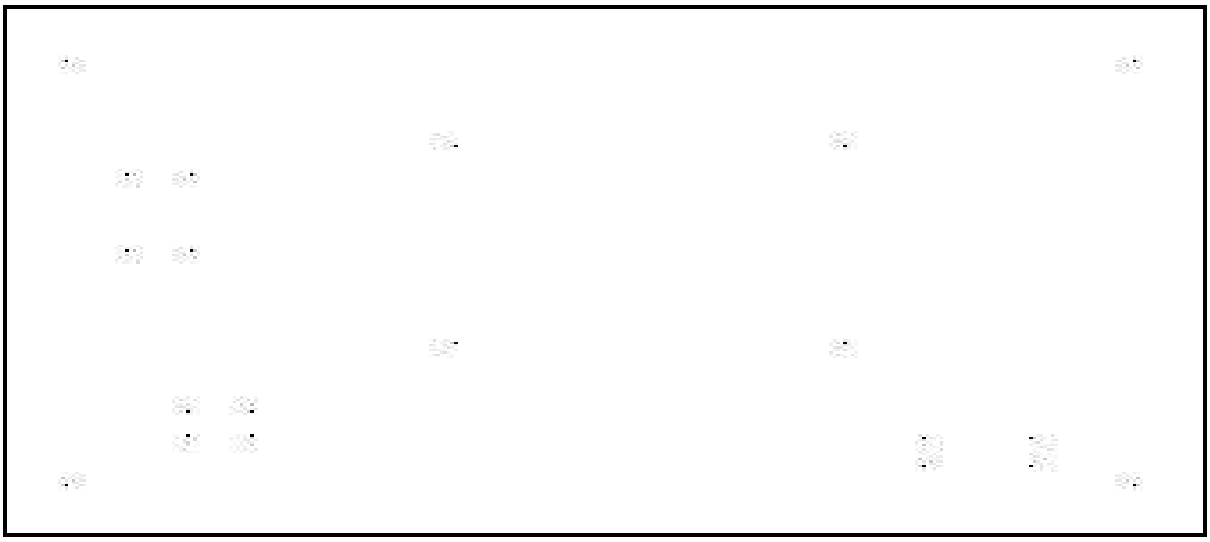


Figure 3.3: Output of application of the Corner-Detector algorithm

3.1.1.2 Point Correspondence using Correlation-Relaxation

- **Matching Through Correlation**

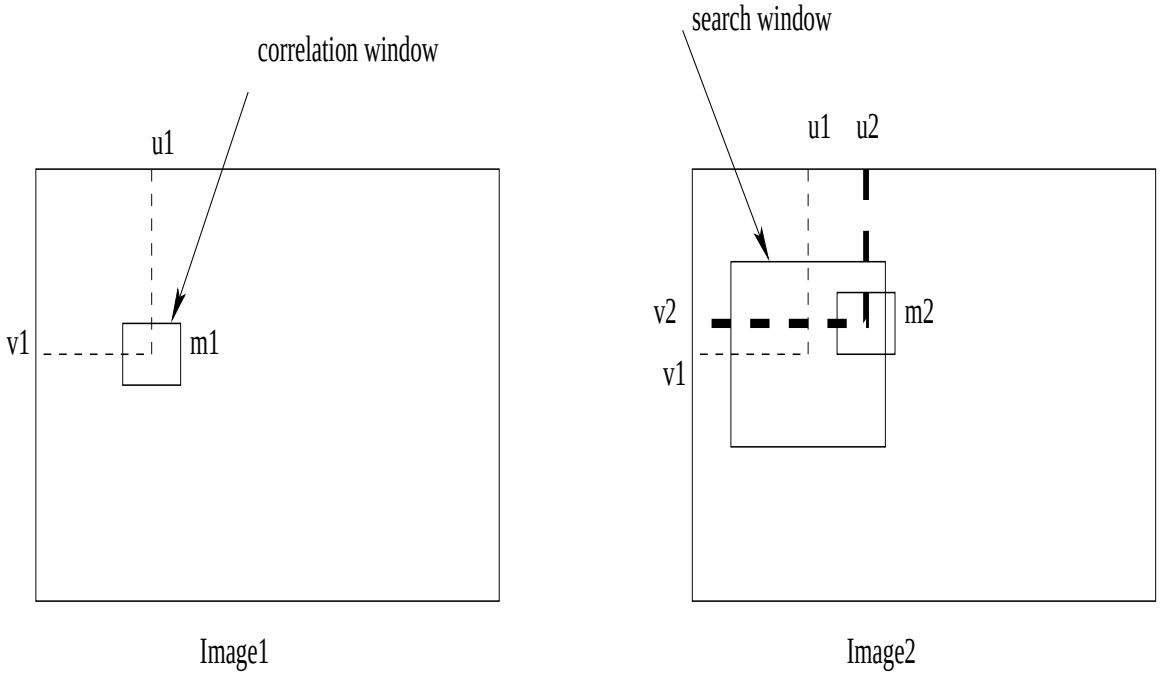


Figure 3.4: Correlation technique

Given a high curvature point **m1** in image1, it uses a correlation window of size $(2n+1) \times (2m+1)$ centered at this point. It then selects a rectangular search area

of size $(2d_u + 1) \times (2d_v + 1)$ around this point in the second image, and performs a correlation operation on a given window between point **m1** in the first image and all high curvature points **m2** lying within the search area in the second image. The search window reflects some *a priori* knowledge about the disparities between the matched points. This is equivalent to reducing the search area for a corresponding point from the whole image to a given window. The correlation score is defined as

$$Score(\mathbf{m1}, \mathbf{m2}) = \frac{\sum_{i=-n}^{n}\sum_{j=-m}^{m}[I_1(u_1+i,v_1+j)-\overline{I_1(u_1,v_1)}]\times[I_2(u_2+i,v_2+j)-\overline{I_2(u_2,v_2)}]}{(2n+1)(2m+1)\sqrt{\sigma^2(I_1)\times\sigma^2(I_2)}}$$

where

$$\overline{I_k(u,v)} = \frac{\sum_{i=-n}^{n}\sum_{j=-m}^{m} I_k(u+i,v+j)}{[(2n+1)(2m+1)]}$$

is the average at point (u, v) of $I_k(k = 1, 2)$, and $\sigma(I_k)$ is the standard deviation of the image I_k in the neighbourhood $(2n + 1) \times (2m + 1)$ of (u, v), which is given by:

$$\sigma(I_k) = \sqrt{\frac{\sum_{i=-n}^{n}\sum_{j=-m}^{m} I_k^2(u,v)}{(2n+1)(2m+1)} - \overline{I_k(u,v)}}$$

The score ranges from -1, for two correlation windows which are not similar at all, to 1, for two correlation windows which are identical.

A constraint on the correlation score is then applied in order to select the most consistent matches: For a given couple of points to be considered as a candidate match, the correlation score must be higher than a given threhold. If the above constraint is fulfilled, we say that the pair of points considered is self-consistent and forms a *candidate match.*

In our implementation, n=m=7 for the correlation window, and a threshold of 0.8 on the correlation score is used. For the search window, d_u and d_v are set to a quater of the image width and height, respectively.

- **Disambiguating Matches Through Relaxation**

Using the correlation technique described above, a point in the first image may be paired to several points in the second image (which are called *candidate matches*). The technique used here for resolving the matching ambiguities falls into a class of *relaxation techniques.* The idea is to allow the candidate matches to reorganize themselves by propagating some constraints, such as continuity and uniqueness, through the neighbourhood.

We use the term **Measure of Support for a Candidate Match** for removing the ambiguity. Consider a candidate match (m_{1i}, m_{2j}) where m_{1i} is a point in the first image and m_{2j} is a point in the second image. Let $N(m_{1i})$ and $N(m_{2j})$ be, respectively, the neighbors of m_{1i} and m_{2j} within a disc radius R. If (m_{1i}, m_{2j}) is a good match, we will expect to see many matches (n_{1k}, n_{2l}), where $n_{1k} \epsilon N(m_{1i})$ and $n_{2l} \epsilon N(m_{2j})$, such that the position of n_{1k} relative to m_{1i} is similar to that of n_{2l} relative to m_{2j}. On the other hand, if (m_{1i}, m_{2j}) is a bad match, we will expect to see only few matches, or even not any at all, in their neighborhood.

More formally, we define a measure of support for a match, which we call the *strength of the match* (SM for abbreviation), as

$$S_M(\mathbf{m_{1i}}, \mathbf{m_{2j}}) = c_{ij} \sum_{n_{1k} \epsilon N(m_{1i})} [\max_{n_{2l} \epsilon N(m_{2j})} \frac{c_{kl}\, \delta(m_{1i}, m_{2j}; n_{1k}, n_{2l})}{1 + dist(m_{1i}, m_{2j}; n_{1k}, n_{2l})}]$$

where c_{ij} and c_{kl} are the goodness of the candidate matches (m_{1i}, m_{2j}) and (n_{1k}, n_{2l}), which can be the correlation scores given in the last subsection, $dist(m_{1i}, m_{2j}; n_{1k}, n_{2l})$ is the average distance of the two pairings, i.e.,

$$dist(m_{1i}, m_{2j}; n_{1k}, n_{2l}) = \frac{[d(m_{1i}, n_{1k}) + d(m_{2j}, n_{2l})]}{2}$$

with $d(m, n) = \parallel m - n \parallel$, the Euclidean distance between $\mathbf{m}$ and $\mathbf{n}$, and

$$\begin{aligned} \delta(m_{1i}, m_{2j}; n_{1k}, n_{2l}) &= e^{-r/\varepsilon_r} \text{ if } (n_{1k}, n_{2l}) \text{ is a candidate match and } r < \varepsilon_r \\ &= 0, \text{ otherwise} \end{aligned}$$

where r is a relative distance difference given by

$$r = \frac{|\, d(m_{1i}, n_{1k}) - d(m_{2j}, n_{2l})|}{dist(m_{1i}, m_{2j}; n_{1k}, n_{2l})}$$

and ε_r is a threshold on the relative distance difference.

Several remarks can be made regarding the measure of matching support.

- Firstly, the SM actually counts the number of candidate matches found in the neighborhoods, but only those whose positions relative to the considered match are similar are counted.
- Secondly, the test of similarity in relative positions is based on the relative distance (the value of r). Indeed, the similarity in relative positions is justified by the hypothesis that an affine transformation can approximate the change between the neighborhoods of the candidate match beign considered. This assumption is reasonable only for a small neighborhood. Thus we should allow larger tolerance in distance differences for distat points, and this is exactly what the criterion specified here does.

- Thirdly, the contribution of a candidate match (n_{1k}, n_{2l}) to the SM (m_{1i}, m_{2j}) is the exponential of the negative relative error r, which is strictly monotonically decreasing function of r. When r is very big, then $\exp(-r/\varepsilon_r) \to 0$, and the candidate match can be ignored. When $r \to 0$, i.e., the difference is very small, then $\exp(-r/\varepsilon_r) \to 1$, and the candidate will largely contribute to the match (m_{1i}, m_{2j}).
- Fourthly, if a point in the left image has several candidate matches in the right image, only the one which has smallest distance difference is accounted for, which is done by the "max" operator.
- Lastly, the contribution of each candidate match in the neighborhood is weighted by its distance to the match. The addition of '1' is only to prevent the over weight for very close points. In other words, a close candidate match gives more support to the match being considered than a distant one.

- **Relaxation Process**

  If we define the energy function as the sum of the strengths of all candidate matches, i.e.,

  $$\tau = \sum_{m_{1i}, m_{2j}} S_M(m_{1i}, m_{2j})$$

  then the problem of disambiguating matches is equivalent to minimizing the energy function τ. The relaxation scheme is one approach to it. It is an iterative procedure, and can be formulated as follows:

  iterate {

  - compute the matching strength for each candidat match
  - update the matches by minimizing the total energy

  } **until the convergance of the energy**

  After the correlation procedure, for each point is the first image, we have a set of candidate matches from the second image(the set is possibly nil). There are several strategies for updating the matching in order to minimixe the total energy. The strategy used here is *winner-take-all*. In brief, the method works as follows:

  At each iteration, any matches which have the highest matching strengths for both of the two images points that formed them are immediately chosen as "correct". This method proceeds as a steepest-descent approach, and is thus fast.

3.1.1.3 Estimating initial warp solution using four point correspondence

After having establishing the matching candidates between 2 adjacent images, we now find the initial warp matrix using the 4-point correspondence method. Let us see how we find the warp matrix from the matching points of two images.

Say, we have a 3D point $P[X, Y, Z]$. We now take a snapshot of the scene containing point P. Let us call the snapshot as *image1*. We now rotate the camera by some angle and take another snapshot of the scene containing P. Let it be called *image2*. The above procedure is understood more clearly in the following diagram:

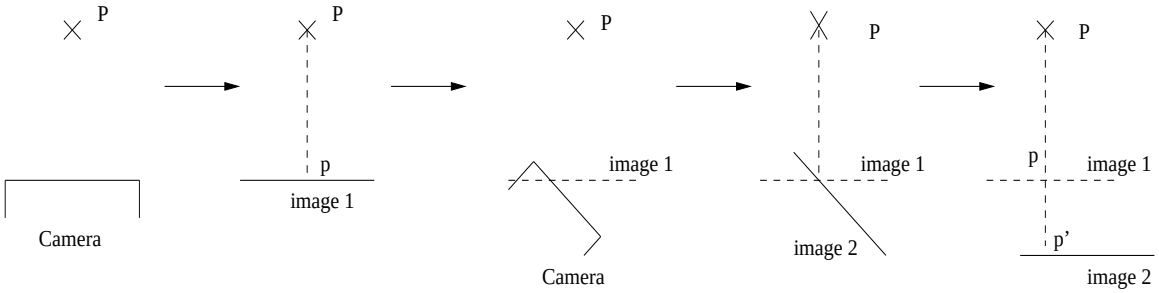


Figure 3.5: Computing warp matrix using 4 point correspondence

Let $p[x, y]$, in *image1*, be the corresponding 2D point of 3D point P. Let $p^{'}[x^{'}, y^{'}]$, in *image2*, be the corresponding 2D point of 3D point P.

We now have the relation,

$$\lambda p = MP \quad and \quad \lambda^{'} p^{'} = M^{'} P \tag{3.1}$$

where, M and M' are 3×4 camera parameter matrix of the form $A[R|T]$ where A is a 3×3 internal camera parameter matrix and $[R|T]$ is a 3×4 external camera parameter matrix. Now,

$$P = \lambda M^{*} p + sN \tag{3.2}$$

where,

- $M^{*}_{4\times3}$ is a left inverse of M i.e. $MM^{*} = I$
- $N_{4\times1}$ lies in null-space of M i.e. $MN = 0$

Also, P lies on a plane. Therefore,

$$C^T P = 0 \tag{3.3}$$

Therefore, using Equation 3.2 and Equation 3.3 we have,

$$C^T P = \lambda C^T M^* p + s C^T N$$

Therefore,

$$s = -\frac{\lambda C^T M^* p}{C^T N} \tag{3.4}$$

Put values of Equation 3.4 in Equation 3.2. Therefore,

$$P = \lambda M^* p \ - \ \frac{\lambda N C^T M^* p}{C^T N}$$

Therefore,

$$P = \lambda \ [\ I \ - \ \frac{N C^T}{C^T N} \ ] \ M^* \ p$$

Thus,

$$P = \lambda \ M'' \ p$$

where,

$$M'' = [\ I \ - \ \frac{N C^T}{C^T N} \ ] \ M^*$$

Now, put $P = \lambda \ M \ \ p$ in Equation $\lambda' p' = M'' P$. Therefore,

$$\lambda' p' = \lambda M' \ M'' p$$

Thus, finally we have,

$$\boxed{\mu \ p' = \mathbf{H} \ p}$$

where, $\mathbf{H} = M' \ M''$
The final equation defining the relationship is:

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

where

- H is the homography matrix (warp matrix) relating both the images. It is a 3×3 matrix and is valid only upto a linear scaling of $p^{'}$. Hence it has 8 parameters to be determined and hence 8 linear equations. Thus, we need atleast 4 point correspondences between 2 images.

The figure below shows as to how the output may look like after stiching together images with their initial estimates. The output here shows one composite image formed by stiching together five partially overlapped images.

Figure 3.6: Composite image formed of mosaicing five partially overlapping images

3.1.1.4 Recovering initial rotation estimates from the computed warp

We can now get the initial rotation estimates from the computed warp using Equation 1.5 if the camera calibration is known accurately. If not, a closed-form solution can be used to derive camera calibration from the warp itself, which is explained in the following subsection.

3.1.2 Calculations for initial estimates of the internal camera parameters from the warp

We have the following equation with us:

$$P = KR_2R_1^{-1}K^{-1}$$

Let $R = R_2R_1^{-1}$. Therefore the equation becomes

$$R = K^{-1}PK$$

if we solve for R. Since $R = R^{-T}$ (the orthonormality condition for rotations), we have,

$$K^{-1}PK = K^TP^{-T}K^{-T}$$

Let $C = KK^T$. Therefore we get,

$$\boxed{PC = CP^{-T}}$$

were

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

But how do we obtain matrix C ?

Now, the solution for **C** can be expressed in terms of eigen-vectors of the projective matrix **P**. The eigen-values of **P** are the same as that of **R**, since they are related by a *similarity transform*. We know the eigen-values of the rotation **R** are 1, $e^{i\theta}$, *and* $e^{-i\theta}$, where θ is the angel of rotation. Let e_0, e_1 *and* e_2 be the eigen-vectors of **P** corresponding to these three eigen-values, respectively. Therefore the most general symmetric solution to **C** is

$$C = c_0e_0e_0^T + c_1(e_1e_2^T + e_2e_1^T)$$

The coefficients c_0 *and* c_1 are solved by enforcing the constraints that the $C_{33} = 1$ and $C_{12} = C_{13}C_{23}$.

After having known **C**, we can now easily estimate internal camera parameter matrix **K** using Cholesky Decomposition of **C**.

3.1.3 Optimization Technique

After having got the initial estimates, we can now go for the *Global Optimization* technique, as discussed in section 2.3, to get optimized rotation and internal camera matrix. This technique uses a gradient based approach for optimization and may get stuck with the problem of local minima. Other optimization techniques like *Simulated Annealing* might just provide better results.

3.2 Final Remarks

We thus saw, in our algorithm implementation, we use the camera instrument only to provide partially overlapping images. The adjacency map relating the images is given as a manual input while the initial estimates of rotation and internal camera parameters are automatically calculated from input images. Global Optimization technique may get stuck with the problem of local minima and hence we may go for other optimization techniques like *Simulated Annealing*.

Chapter 4

Future Work

For the past one year, other than my courses, I have been increasing my awareness in the area of computer graphics and animation. I have started some literature survey on Computer Graphics and Animation. But as of now, I am yet to decide upon *the problem* I would like to solve for my Ph.D.dissertation. So I have intentions to increase my level of awareness in the general area of computer graphics and animation, and go ahead for an extensive and exhaustive literature survey to know the "state-of-the-art". Then I plan to venture beyond by taking up a problem that warrants more explanation and holds promise for being an exciting and worthy Ph.D. research topic.

Bibliography

[1] Satyan Coorg and Seth Teller. Spherical mosaics with quaternions and dense correlation. In *IJCV, 37(3)*, pages 259–273, June 2000.

[2] Oliver Faugeras. Three-dimensional computer vision. In *MIT Press*, 1993.

[3] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Plessey Research Roke Manor, United Kingdom, The Plessey Company*, 1988.

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

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

[6] 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.

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

[8] R. Tsai. A versatile camera calibration technique for high accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. In *IEEE Journal of Robotics and Automation, RA-3(4)*, August 1987.

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

[10] Zhengyou Zhang, Rachid Deriche, Olivier Faugeras, and Quang-Tuan Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. In *INRIA, Programme 4, Robotique, image et vision, Project Robotvis*, May 1994.