Chapter 8

Image Alignment and Stitching

Presented by: Ryandhimas Zezario

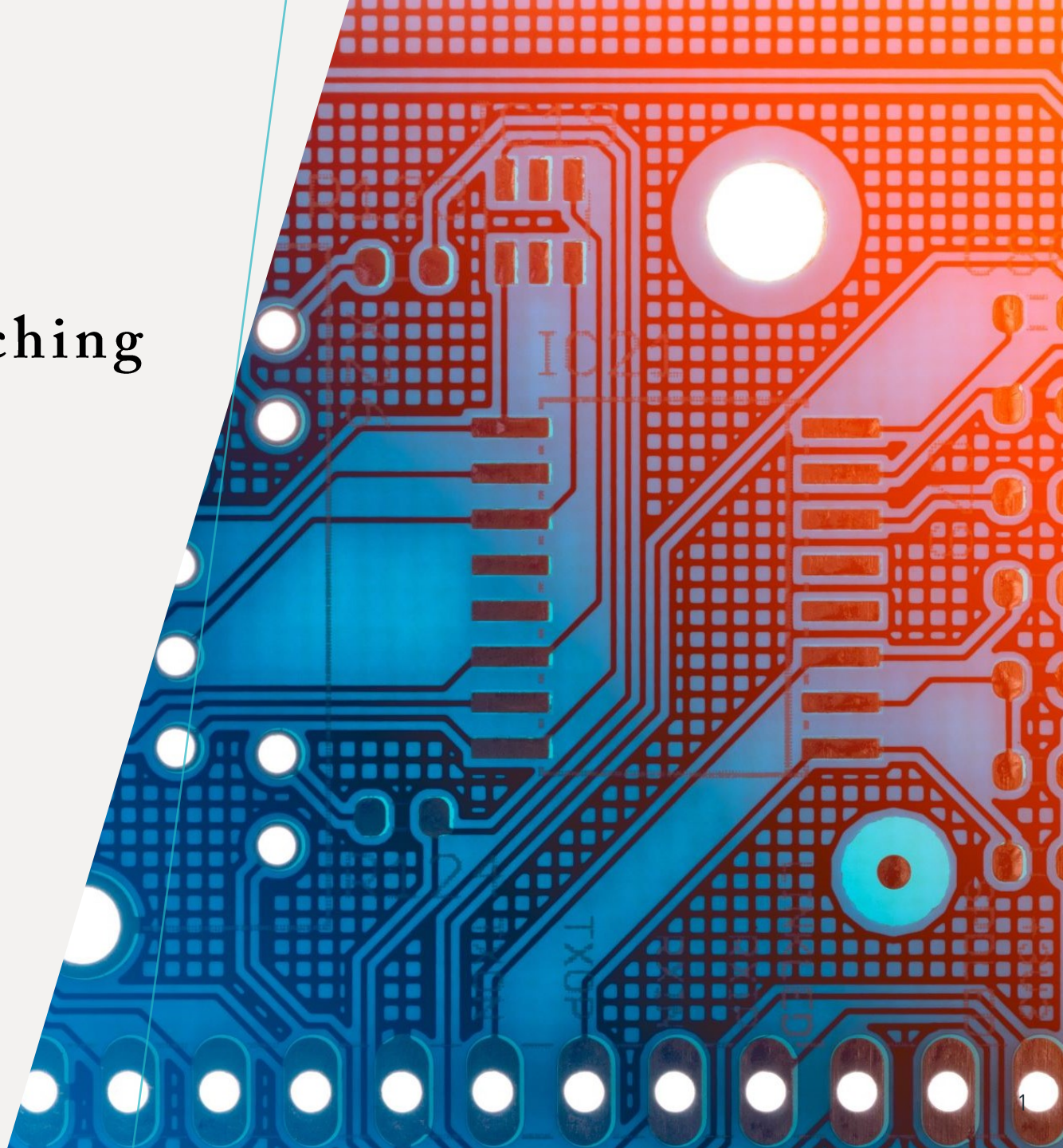
RYANDHIMAS@CITI.SINICA.EDU.TW

0978870357

授課教授：傅楸善 博士

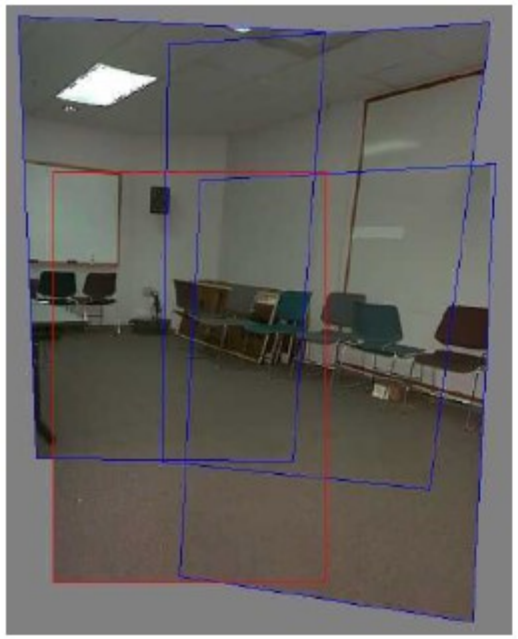
國立臺灣大學 National Taiwan University

DC&CV LAB
CSIE NTU 2020



Outlines

- 8.1 Pairwise alignment
- 8.2 Image stitching
- 8.3 Global alignment
- 8.4 Compositing



(a)

Geometric alignment of 2D images for stitching (Szeliski and Shum 1997) ©

(b)

A spherical panorama constructed from 54 photographs (Szeliski and Shum 1997) © 1997 ACM

A multi-image panorama automatically assembled from an unordered photo collection

8.1 Pairwise Alignment

- Feature-based alignment is the problem of estimating the motion between two or more sets of matched 2D or 3D points.

- In this section, we restrict ourselves to global parametric transformations, such as higher order transformation for curved surfaces

8.1 Pairwise Alignment

Transform	Matrix	Parameters p	Jacobian J
translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$	(t_x, t_y)	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
Euclidean	$\begin{bmatrix} c_\theta & -s_\theta & t_x \\ s_\theta & c_\theta & t_y \end{bmatrix}$	(t_x, t_y, θ)	$\begin{bmatrix} 1 & 0 & -s_\theta x - c_\theta y \\ 0 & 1 & c_\theta x - s_\theta y \end{bmatrix}$
similarity	$\begin{bmatrix} 1+a & -b & t_x \\ b & 1+a & t_y \end{bmatrix}$	(t_x, t_y, a, b)	$\begin{bmatrix} 1 & 0 & x & -y \\ 0 & 1 & y & x \end{bmatrix}$
affine	$\begin{bmatrix} 1+a_{00} & a_{01} & t_x \\ a_{10} & 1+a_{11} & t_y \end{bmatrix}$	$(t_x, t_y, a_{00}, a_{01}, a_{10}, a_{11})$	$\begin{bmatrix} 1 & 0 & x & y & 0 & 0 \\ 0 & 1 & 0 & 0 & x & y \end{bmatrix}$
projective	$\begin{bmatrix} 1+h_{00} & h_{01} & h_{02} \\ h_{10} & 1+h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{bmatrix}$	$(h_{00}, h_{01}, \ldots, h_{21})$	(see Section 8.1.3)

Table 8.1 *Jacobians of the 2D coordinate transformations* $\mathbf{x}' = \mathbf{f}(\mathbf{x}; \mathbf{p})$ *shown in Table 2.1, where we have re-parameterized the motions so that they are identity for* $\mathbf{p} = 0$.

8.1 Pairwise Alignment

8.1.1 2D alignment using least squares

- Given a set of matched feature points $\{(\mathbf{x}_i, \mathbf{x}'_i)\}$ and a planar parametric transformation of the form:

$$\mathbf{x}' = \mathbf{f}(\mathbf{x}; \mathbf{p}), \tag{8.1}$$

- The usual way to estimate motion parameters p is to use least squares, i.e., to minimize the sum of squared residuals

$$E_{\mathrm{LS}} = \sum_i \|\mathbf{r}_i\|^2 = \sum_i \|\mathbf{f}(\mathbf{x}_i; \mathbf{p}) - \mathbf{x}'_i\|^2, \tag{8.2}$$

- where

$$\mathbf{r}_i = \mathbf{x}'_i - \mathbf{f}(\mathbf{x}_i; \mathbf{p}) = \hat{\mathbf{x}}'_i - \tilde{\mathbf{x}}'_i \tag{8.3}$$

8.1 Pairwise Alignment

8.1.1 2D alignment using least squares

- Many of the motion models presented, i.e., translation, similarity, and affine, have a linear relationship between the amount of motion $\Delta \mathbf{x} = \mathbf{x}' - \mathbf{x}$ and the unknown parameters $\mathbf{p}$.

$$\Delta \mathbf{x} = \mathbf{x}' - \mathbf{x} = \mathbf{J}(\mathbf{x})\mathbf{p}, \tag{8.4}$$

- where $\mathbf{J} = \partial \mathbf{f} / \partial \mathbf{p}$ is the Jacobian of the transformation f with respect to the motion parameters $\mathbf{p}$. In this case, a simple linear regression (linear least squares problem) can be formulated as:

$$E_{\mathrm{LLS}} = \sum_i \|\mathbf{J}(\mathbf{x}_i)\mathbf{p} - \Delta \mathbf{x}_i\|^2 \tag{8.5}$$

$$= \mathbf{p}^T \left[\sum_i \mathbf{J}^T(\mathbf{x}_i)\mathbf{J}(\mathbf{x}_i)\right] \mathbf{p} - 2\mathbf{p}^T \left[\sum_i \mathbf{J}^T(\mathbf{x}_i)\Delta \mathbf{x}_i\right] + \sum_i \|\Delta \mathbf{x}_i\|^2 \tag{8.6}$$

$$= \mathbf{p}^T \mathbf{A}\mathbf{p} - 2\mathbf{p}^T \mathbf{b} + c. \tag{8.7}$$

8.1 Pairwise Alignment

8.1.1 2D alignment using least squares

- The minimum can be found by solving the Symmetric Positive Definite (SPD) system of normal equations

$$\mathbf{A}\mathbf{p} = \mathbf{b}, \tag{8.8}$$

where

$$\mathbf{A} = \sum_i \mathbf{J}^T(\mathbf{x}_i)\mathbf{J}(\mathbf{x}_i) \tag{8.9}$$

is called the Hessian and $\mathbf{b} = \sum_i \mathbf{J}^T(\mathbf{x}_i)\Delta\mathbf{x}_i$.

8.1 Pairwise Alignment

8.1.1 2D alignment using least squares

Uncertainty weighting

- The above least squares formulation assumes that all feature points are matched with the same accuracy.

- This is often not the case, since certain points may fall into more textured regions than others. If we associate a scalar variance estimate σ_i^2 with each correspondence, we can minimize the weighted least squares problem instead.

$$E_{\mathrm{WLS}} = \sum_i \sigma_i^{-2} \|\mathbf{r}_i\|^2. \tag{8.10}$$

8.1 Pairwise Alignment

8.1.1 2D alignment using least squares

Uncertainty weighting

- Weighting each squared residual by its inverse covariance $\Sigma_i^{-1} = \sigma_n^{-2}\mathbf{A}_i$ (which is called the information matrix), we obtain

$$E_{\mathrm{CWLS}} = \sum_i \|\mathbf{r}_i\|^2_{\Sigma_i^{-1}} = \sum_i \mathbf{r}_i^T \Sigma_i^{-1} \mathbf{r}_i = \sum_i \sigma_n^{-2} \mathbf{r}_i^T \mathbf{A}_i \mathbf{r}_i. \tag{8.11}$$

8.1 Pairwise Alignment

8.1.2 Application: Panography

- In a panograph, images are translated and optionally rotated and scaled before being blended with simple averaging

Figure 8.3 *A simple panograph consisting of three images automatically aligned with a translational model and then averaged together.*

8.1 Pairwise Alignment

8.1.2 Application: Panography

- Consider a simple translational model. We want all the corresponding features in different images to line up as best as possible.
- Let be $\mathbf{t}_j$ the location of the jth image coordinate frame in the global composite frame and $\mathbf{x}_{ij}$ be the location of the ith matched feature in the jth image.
- In order to align the images, we wish to minimize the least squares error

$$E_{\text{PLS}} = \sum_{ij} \|(\mathbf{t}_j + \mathbf{x}_{ij}) - \mathbf{x}_i\|^2, \tag{8.12}$$

8.1 Pairwise Alignment

8.1.3 Iterative algorithms

- While linear least squares is the simplest method for estimating parameters, most problems in computer vision do not have a simple linear relationship between the measurements and the unknowns.

- In this case, the resulting problem is called non-linear least squares or non-linear regression.

8.1 Pairwise Alignment

8.1.3 Iterative algorithms

- Consider, for example, the problem of estimating a rigid Euclidean 2D transformation (translation plus rotation) between two sets of points.
- If we parameterize this transformation by the translation amount and the rotation angle θ, as in Table 2.1, the Jacobian of this transformation depends on the current value of θ.
- By re-parameterized the motion matrices so that they are always the identity at the origin p = 0, which makes it easier to initialize the motion parameters

8.1 Pairwise Alignment

8.1.3 Iterative algorithms

- To minimize the non-linear least squares problem, we iteratively find an update $\Delta\mathbf{p}$ to the current parameter estimate p by minimizing

$$E_{\text{NLS}}(\Delta\mathbf{p}) = \sum_i \|\mathbf{f}(\mathbf{x}_i; \mathbf{p} + \Delta\mathbf{p}) - \mathbf{x}'_i\|^2 \tag{8.13}$$

$$\approx \sum_i \|\mathbf{J}(\mathbf{x}_i; \mathbf{p})\Delta\mathbf{p} - \mathbf{r}_i\|^2 \tag{8.14}$$

$$= \Delta\mathbf{p}^T \left[\sum_i \mathbf{J}^T\mathbf{J}\right] \Delta\mathbf{p} - 2\Delta\mathbf{p}^T \left[\sum_i \mathbf{J}^T\mathbf{r}_i\right] + \sum_i \|\mathbf{r}_i\|^2 \tag{8.15}$$

$$= \Delta\mathbf{p}^T\mathbf{A}\Delta\mathbf{p} - 2\Delta\mathbf{p}^T\mathbf{b} + c, \tag{8.16}$$

where the "Hessian"[5] $\mathbf{A}$ is the same as Equation (8.9) and the right hand side vector

$$\mathbf{b} = \sum_i \mathbf{J}^T(\mathbf{x}_i)\mathbf{r}_i \tag{8.17}$$

8.1 Pairwise Alignment

8.1.3 Iterative algorithms

- Once A and b have been computed, we solve for $\Delta \mathbf{p}$ using

$$(\mathbf{A} + \lambda \text{diag}(\mathbf{A}))\Delta \mathbf{p} = \mathbf{b}, \tag{8.18}$$

- For the other 2D motion models, the derivatives in Table 8.1 are all fairly straightforward, except for the projective 2D motion (homography), which arises in image-stitching applications.

8.1 Pairwise Alignment

8.1.3 Iterative algorithms

- These equations can be re-written from (2.21) in their new parametric form as

$$x' = \frac{(1+h_{00})x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + 1} \quad \text{and} \quad y' = \frac{h_{10}x + (1+h_{11})y + h_{12}}{h_{20}x + h_{21}y + 1}. \tag{8.19}$$

The Jacobian is therefore

$$\mathbf{J} = \frac{\partial \mathbf{f}}{\partial \mathbf{p}} = \frac{1}{D}\begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -x'x & -x'y \\ 0 & 0 & 0 & x & y & 1 & -y'x & -y'y \end{bmatrix}, \tag{8.20}$$

where $D = h_{20}x + h_{21}y + 1$ is the denominator in (8.19), which depends on the current parameter settings (as do x' and y').

8.1 Pairwise Alignment

8.1.3 Iterative algorithms

- An initial guess for the eight unknowns $\{h_{00}, h_{01}, \ldots, h_{21}\}$ can be obtained by multiplying both sides of the equations in (8.19) through by the denominator, which yields the linear set of equations

$$\begin{bmatrix} \hat{x}' - x \\ \hat{y}' - y \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -\hat{x}'x & -\hat{x}'y \\ 0 & 0 & 0 & x & y & 1 & -\hat{y}'x & -\hat{y}'y \end{bmatrix} \begin{bmatrix} h_{00} \\ \vdots \\ h_{21} \end{bmatrix}. \tag{8.21}$$

- One way to compensate for this is to reweight each equation by the inverse of the current estimate of the denominator, D,

$$\frac{1}{D}\begin{bmatrix} \hat{x}' - x \\ \hat{y}' - y \end{bmatrix} = \frac{1}{D}\begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -\hat{x}'x & -\hat{x}'y \\ 0 & 0 & 0 & x & y & 1 & -\hat{y}'x & -\hat{y}'y \end{bmatrix} \begin{bmatrix} h_{00} \\ \vdots \\ h_{21} \end{bmatrix}. \tag{8.22}$$

8.1 Pairwise Alignment

8.1.3 Iterative algorithms

- The most principled way to do the estimation, however, is to directly minimize the squared residual equations (8.13) using the Gauss–Newton approximation, i.e., performing a first order Taylor series expansion in p, as shown in (8.14), which yields the set of equations

$$\begin{bmatrix} \hat{x}' - \tilde{x}' \\ \hat{y}' - \tilde{y}' \end{bmatrix} = \frac{1}{D} \begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -\tilde{x}'x & -\tilde{x}'y \\ 0 & 0 & 0 & x & y & 1 & -\tilde{y}'x & -\tilde{y}'y \end{bmatrix} \begin{bmatrix} \Delta h_{00} \\ \vdots \\ \Delta h_{21} \end{bmatrix}. \tag{8.23}$$

8.1 Pairwise Alignment

8.1.4 Robust least squares and RANSAC

- It is preferable to use an M-estimator when there are outliers among the correspondences .
- It involves applying a robust penalty function p(r) to the residuals instead of squaring them

$$E_{\mathrm{RLS}}(\Delta\mathbf{p}) = \sum_i \rho(\|\mathbf{r}_i\|) \tag{8.25}$$

- We can take the derivative of this function with respect to p and set it to 0,

$$\sum_i \psi(\|\mathbf{r}_i\|)\frac{\partial\|\mathbf{r}_i\|}{\partial\mathbf{p}} = \sum_i \frac{\psi(\|\mathbf{r}_i\|)}{\|\mathbf{r}_i\|}\mathbf{r}_i^T\frac{\partial\mathbf{r}_i}{\partial\mathbf{p}} = 0, \tag{8.26}$$

where $\psi(r) = \rho'(r)$ is the derivative of ρ and is called the *influence function*.

8.1 Pairwise Alignment

8.1.4 Robust least squares and RANSAC

- If we introduce a weight function, $w(r) = \Psi(r)/r$, , we observe that finding the stationary point of (8.25) using (8.26) is equivalent to minimizing the Iteratively Reweighted Least Squares (IRLS) problem

$$E_{\text{IRLS}} = \sum_i w(\|\mathbf{r}_i\|)\|\mathbf{r}_i\|^2, \tag{8.27}$$

where the $w(\|\mathbf{r}_i\|)$ play the same local weighting role as σ_i^{-2} in (8.10)

8.1 Pairwise Alignment

8.1.4 Robust least squares and RANSAC

- The IRLS algorithm alternates between computing the influence functions $w(\|\mathbf{r}_i\|)$ and solving the resulting weighted least squares problem (with fixed w values)
- While M-estimators can definitely help reduce the influence of outliers, in some cases, starting with too many outliers will prevent IRLS (or other gradient descent algorithms) from converging to the global optimum.
- A better approach is often to find a starting set of inlier correspondences, i.e., points that are consistent with a dominant motion estimate

8.1 Pairwise Alignment

8.1.4 Robust least squares and RANSAC

- Two widely used approaches to this problem are called RANdom SAmple Consensus, or RANSAC for short (Fischler and Bolles 1981), and Least Median of Squares (LMS) (Rousseeuw 1984).

- Both techniques start by selecting (at random) a subset of k correspondences, which is then used to compute an initial estimate for p. The residuals of the full set of correspondences are then computed as

$$\mathbf{r}_i = \tilde{\mathbf{x}}'_i(\mathbf{x}_i; \mathbf{p}) - \hat{\mathbf{x}}'_i, \tag{8.28}$$

where $\tilde{\mathbf{x}}'_i$ are the *estimated* (mapped) locations and $\hat{\mathbf{x}}'_i$ are the sensed (detected) feature point locations.

8.1 Pairwise Alignment

8.1.4 Robust least squares and RANSAC

- The RANSAC technique then counts the number of inliers that are within of their predicted location, i.e., whose $\|\mathbf{r}_i\| \leq \epsilon$. (The epsilon value is application dependent but is often around 1–3 pixels.)

- Least median of squares finds the median value of the $\|\mathbf{r}_i\|^2$ values.

- The random selection process is repeated S times and the sample set with the largest number of inliers (or with the smallest median residual) is kept as the final solution.

8.1 Pairwise Alignment

8.1.4 Robust least squares and RANSAC

- When the number of measurements is quite large, it may be preferable to only score a subset of the measurements in an initial round that selects the most plausible hypotheses for additional scoring and selection.
- This modification of RANSAC, which can significantly speed up its performance, is called Preemptive RANSAC (Nist´er 2003).
- In another variant on RANSAC called PROSAC (PROgressive SAmple Consensus), random samples are initially added from the most "confident" matches, thereby speeding up the process of finding a (statistically) likely good set of inliers (Chum and Matas 2005).

8.1 Pairwise Alignment

8.1.5 3D Alignment

- Instead of aligning 2D sets of image features, many computer vision applications require the alignment of 3D points.

- In the case where the 3D transformations are linear in the motion parameters, e.g., for translation, similarity, and affine, regular least squares (8.5) can be used

The case of rigid (Euclidean) motion,

$$E_{\mathrm{R3D}} = \sum_i \|\mathbf{x}'_i - \mathbf{R}\mathbf{x}_i - \mathbf{t}\|^2, \tag{8.31}$$

which arises more frequently and is often called the absolute orientation problem (Horn 1987), requires slightly different techniques.

8.1 Pairwise Alignment

8.1.5 3D Alignment

- If only scalar weightings are being used (as opposed to full 3D per-point anisotropic covariance estimates), the weighted centroids of the two point clouds c and c0 can be used to estimate the translation $\mathbf{t} = \mathbf{c}' - \mathbf{R}\mathbf{c}$.

- We are then left with the problem of estimating the rotation between two sets of points $\{\hat{\mathbf{x}}_i = \mathbf{x}_i - \mathbf{c}\}$ and $\{\hat{\mathbf{x}}'_i = \mathbf{x}'_i - \mathbf{c}'\}$ that are both centered at the origin

- One commonly used technique is called the orthogonal Procrustes algorithm (Golub and Van Loan 1996, p. 601) and involves computing the singular value decomposition (SVD) of the 3 x 3 correlation matrix

$$\mathbf{C} = \sum_i \hat{\mathbf{x}}' \hat{\mathbf{x}}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T. \tag{8.32}$$

The rotation matrix is then obtained as $\mathbf{R} = \mathbf{U}\mathbf{V}^T$.

8.2 Image Stitching

- Image stitching algorithms create the high-resolution photo-mosaics used to produce today's digital maps and satellite photos. They are also now a standard mode in smartphone cameras and can be used to create beautiful ultra wide-angle panoramas.
- Image stitching originated in the photogrammetry community, where more manually intensive methods based on surveyed ground control points or manually registered tie points have long been used to register aerial photos into large-scale photo-mosaics (Slama 1980).
- While early techniques worked by directly minimizing pixel-to-pixel dissimilarities, today's algorithms extract a sparse set of features and match them to each other, as described in Chapter 7.

8.2 Image Stitching

- What, then, are the essential problems in image stitching?
- As with image alignment, we must first determine the appropriate mathematical model relating pixel coordinates in one image to pixel coordinates in another

8.2 Image Stitching

8.2.1 Parametric motion models

- Before we can register and align images, we need to establish the mathematical relationships that map pixel coordinates from one image to another.

- A variety of such parametric motion models are possible, from simple 2D transforms, to planar perspective models, 3D camera rotations, lens distortions, and mapping to non-planar (e.g., cylindrical) surfaces

8.2 Image Stitching

8.2.1 Parametric motion models

(a) translation [2 dof] (b) affine [6 dof] (c) perspective [8 dof] (d) 3D rotation [3+ dof]

Figure 8.4 *Two-dimensional motion models and how they can be used for image stitching.*

8.2 Image Stitching

8.2.1 Parametric motion models

- More recent stitching algorithms first extract features and then match them up, often using robust techniques such as RANSAC (Section 8.1.4) to compute a good set of inliers.

- The final computation of the homography i.e., the solution of the least squares fitting problem given pairs of corresponding features

$$x_1 = \frac{(1+h_{00})x_0 + h_{01}y_0 + h_{02}}{h_{20}x_0 + h_{21}y_0 + 1} \quad \text{and} \tag{8.35}$$

$$y_1 = \frac{h_{10}x_0 + (1+h_{11})y_0 + h_{12}}{h_{20}x_0 + h_{21}y_0 + 1}, \tag{8.36}$$

uses iterative least squares, as described in Section 8.1.3 and Equations (8.21–8.23).

8.2 Image Stitching

8.2.2 Application: Whiteboard and document scanning

- The simplest image-stitching application is to stitch together a number of image scans taken on a flatbed scanner.
- Say you have a large map, or a piece of child's artwork, that is too large to fit on your scanner.
- Simply take multiple scans of the document, making sure to overlap the scans by a large enough amount to ensure that there are enough common features.
- Next, take successive pairs of images that you know overlap, extract features, match them up, and estimate the 2D rigid transform

8.2 Image Stitching

8.2.2 Application: Whiteboard and document scanning

- Two-point RANSAC, if necessary, to find a good set of inliers.
- Then, on a final compositing surface (aligned with the first scan, for example), resample your images (Section 3.6.1) and average them together.
- Can you see any potential problems with this scheme?

8.2 Image Stitching

8.2.2 Application: Whiteboard and document scanning

- One complication is that a 2D rigid transformation is non-linear in the rotation angle θ so you will have to either use non-linear least squares or constrain R to be orthonormal, as described in Section 8.1.3.
- A bigger problem lies in the pairwise alignment process. As you align more and more pairs, the solution may drift so that it is no longer globally consistent.
- In this case, a global optimization procedure, as described in Section 8.3, may be required.

8.2 Image Stitching

8.2.3 Rotational panoramas

- The most typical case for panoramic image stitching is when the camera undergoes a pure rotation.
- Think of standing at the rim of the Grand Canyon. Relative to the distant geometry in the scene, as you snap away, the camera is undergoing a pure rotation, which is equivalent to assuming that all points are very far from the camera, i.e., on the plane at infinity

8.2 Image Stitching

8.2.3 Rotational panoramas

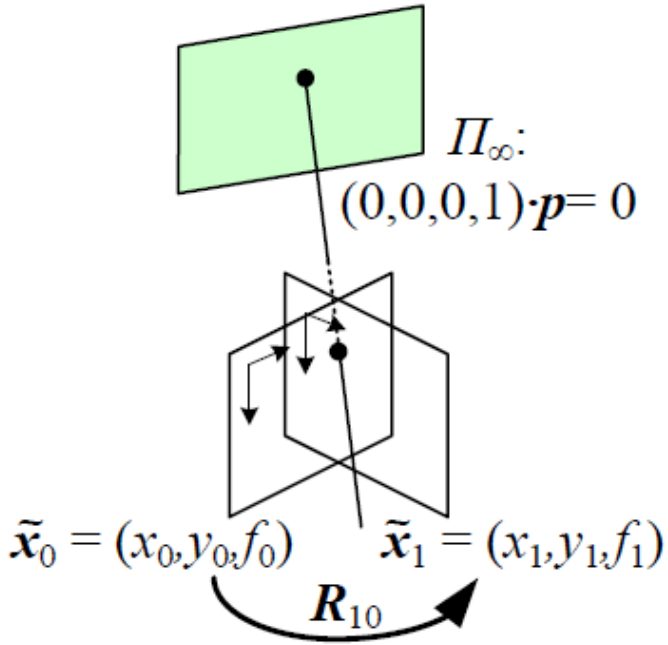


Figure 8.5 *Pure 3D camera rotation. The form of the homography (mapping) is particularly simple and depends only on the 3D rotation matrix and focal lengths.*

8.2 Image Stitching

8.2.3 Rotational panoramas

- Setting $t_0 = t_1 = 0$, we get the simplified 3X3 homography

$$\tilde{\mathbf{H}}_{10} = \mathbf{K}_1 \mathbf{R}_1 \mathbf{R}_0^{-1} \mathbf{K}_0^{-1} = \mathbf{K}_1 \mathbf{R}_{10} \mathbf{K}_0^{-1}, \tag{8.38}$$

where $\mathbf{K}_k = \text{diag}(f_k, f_k, 1)$ is the simplified camera intrinsic matrix (2.59), assuming that $c_x = c_y = 0$, i.e., we are indexing the pixels starting from the image center (Szeliski 1996).

8.2 Image Stitching

8.2.3 Rotational panoramas

- Setting $t_0 = t_1 = 0$, we get the simplified 3X3 homography

This can also be re-written as

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \sim \begin{bmatrix} f_1 & & \\ & f_1 & \\ & & 1 \end{bmatrix} \mathbf{R}_{10} \begin{bmatrix} f_0^{-1} & & \\ & f_0^{-1} & \\ & & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} \tag{8.39}$$

or

$$\begin{bmatrix} x_1 \\ y_1 \\ f_1 \end{bmatrix} \sim \mathbf{R}_{10} \begin{bmatrix} x_0 \\ y_0 \\ f_0 \end{bmatrix}, \tag{8.40}$$

8.2 Image Stitching

8.2.3 Rotational panoramas

- Given this representation, how do we update the rotation matrices to best align two overlapping images?

- Given a current estimate for the homography $\tilde{\mathbf{H}}_{10}$ in (8.38), the best way to update $\mathbf{R}_{10}$ is to prepend an incremental rotation matrix $\mathbf{R}(\boldsymbol{\omega})$ to the current estimate $\mathbf{R}_{10}$ (Szeliski and Shum 1997; Shum and Szeliski 2000),

$$\tilde{\mathbf{H}}(\boldsymbol{\omega}) = \mathbf{K}_1\mathbf{R}(\boldsymbol{\omega})\mathbf{R}_{10}\mathbf{K}_0^{-1} = [\mathbf{K}_1\mathbf{R}(\boldsymbol{\omega})\mathbf{K}_1^{-1}][\mathbf{K}_1\mathbf{R}_{10}\mathbf{K}_0^{-1}] = \mathbf{D}\tilde{\mathbf{H}}_{10}. \tag{8.41}$$

$$\mathbf{D} = \mathbf{K}_1\mathbf{R}(\boldsymbol{\omega})\mathbf{K}_1^{-1} \approx \mathbf{K}_1(\mathbf{I} + [\boldsymbol{\omega}]_\times)\mathbf{K}_1^{-1} = \begin{bmatrix} 1 & -\omega_z & f_1\omega_y \\ \omega_z & 1 & -f_1\omega_x \\ -\omega_y/f_1 & \omega_x/f_1 & 1 \end{bmatrix}. \tag{8.42}$$

8.2 Image Stitching

8.2.3 Rotational panoramas

- Notice how there is now a nice one-to-one correspondence between the entries in the D matrix and the $h_{00}, \ldots, h_{21}$ parameters used in Table 8.1 and Equation (8.19), i.e.,

$$(h_{00}, h_{01}, h_{02}, h_{00}, h_{11}, h_{12}, h_{20}, h_{21}) = (0, -\omega_z, f_1\omega_y, \omega_z, 0, -f_1\omega_x, -\omega_y/f_1, \omega_x/f_1). \tag{8.43}$$

We can therefore apply the chain rule to Equations (8.24 and 8.43) to obtain

$$\begin{bmatrix} \hat{x}' - x \\ \hat{y}' - y \end{bmatrix} = \begin{bmatrix} -xy/f_1 & f_1 + x^2/f_1 & -y \\ -(f_1 + y^2/f_1) & xy/f_1 & x \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}, \tag{8.44}$$

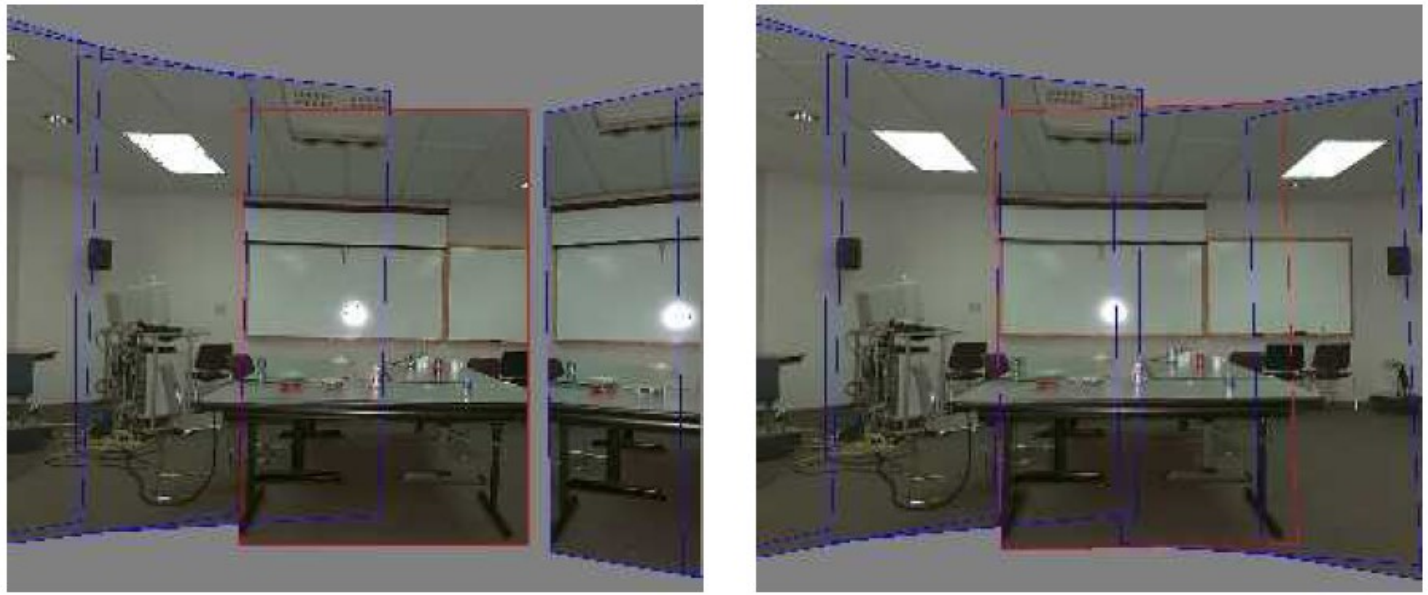
8.2 Image Stitching

8.2.4 Gap Closing

- The techniques presented in this section can be used to estimate a series of rotation matrices and focal lengths, which can be chained together to create large panoramas.
- Unfortunately, because of accumulated errors, this approach will rarely produce a closed 360 panorama. Instead, there will invariably be either a gap or an overlap
- We can solve this problem by matching the first image in the sequence with the last one.
- The difference between the two rotation matrix estimates associated with the repeated first indicates the amount of misregistration

8.2 Image Stitching

8.2.4 Gap Closing



(a) (b)

Figure 8.6 *Gap closing (Szeliski and Shum 1997) © 1997 ACM: (a) A gap is visible when the focal length is wrong ($f = 510$). (b) No gap is visible for the correct focal length ($f = 468$).*

8.2 Image Stitching

8.2.5 Application: Video summarization and compression

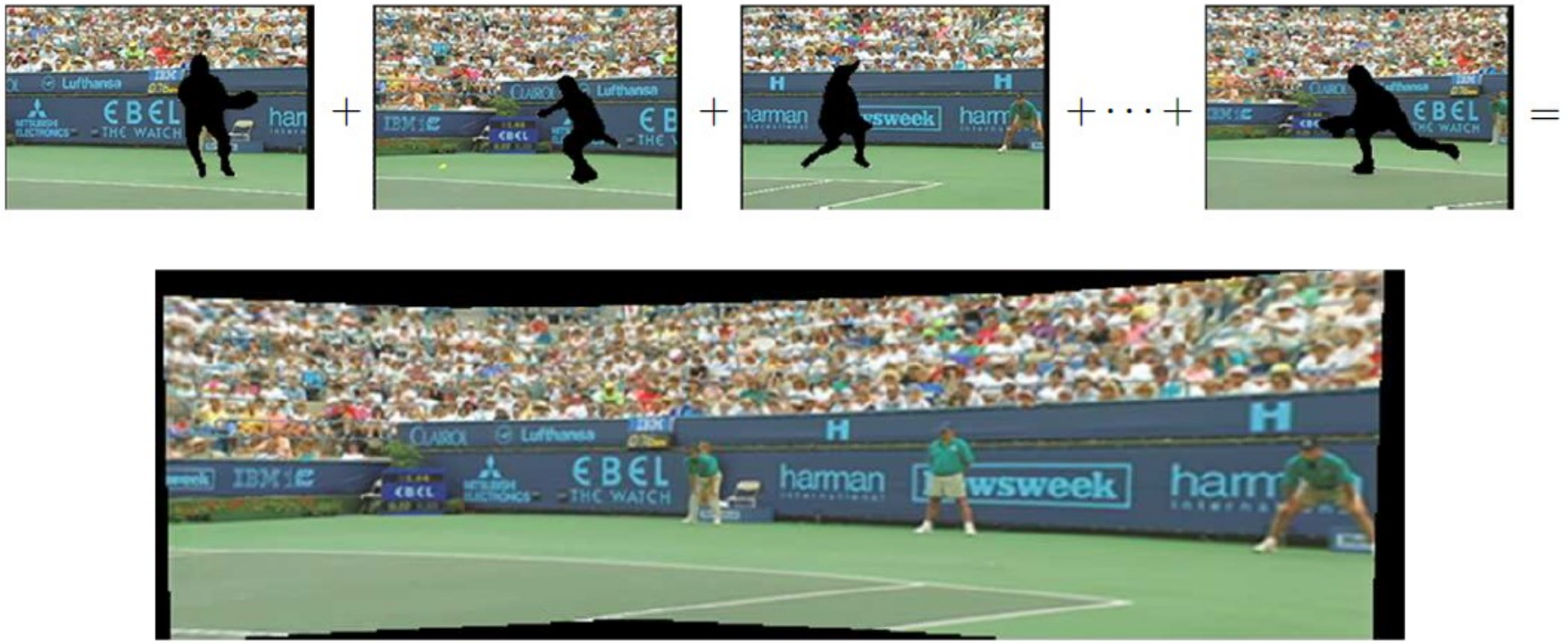


Figure 8.7 *Video stitching the background scene to create a single* sprite *image that can be transmitted and used to re-create the background in each frame (Lee, Chen* et al. *1997)* © *1997 IEEE.*

8.2 Image Stitching

8.2.6 Cylindrical and spherical coordinates

- An alternative to using homographies or 3D motions to align images is to first warp the images into cylindrical coordinates and then use a pure translational model to align them (Chen 1995; Szeliski 1996).

- Unfortunately, this only works if the images are all taken with a level camera or with a known tilt angle.

- Assume for now that the camera is in its canonical position, i.e., its rotation matrix is the identity, $R = I$, so that the optical axis is aligned with the z axis and the y axis is aligned vertically. The 3D ray corresponding to an $(x; y)$ pixel is therefore $(x; y; f)$.

8.2 Image Stitching

8.2.6 Cylindrical and spherical coordinates

We wish to project this image onto a *cylindrical surface* of unit radius (Szeliski 1996). Points on this surface are parameterized by an angle θ and a height h, with the 3D cylindrical coordinates corresponding to (θ, h) given by

$$(\sin\theta, h, \cos\theta) \propto (x, y, f), \tag{8.45}$$

as shown in Figure 8.8a. From this correspondence, we can compute the formula for the warped or mapped coordinates (Szeliski and Shum 1997),

8.2 Image Stitching

8.2.6 Cylindrical and spherical coordinates

$$x' = s\theta = s\tan^{-1}\frac{x}{f}, \tag{8.46}$$

$$y' = sh = s\frac{y}{\sqrt{x^2 + f^2}}, \tag{8.47}$$

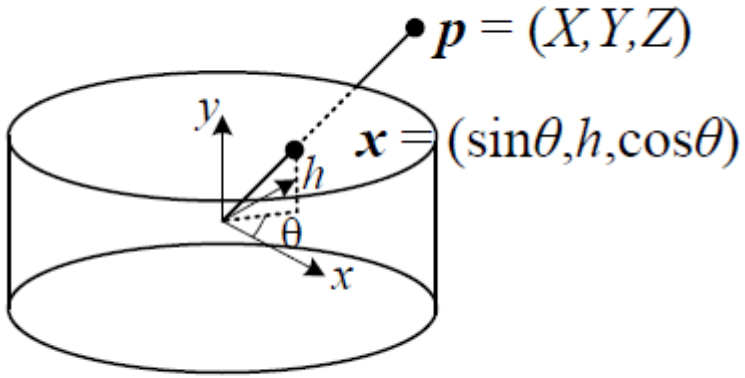
where s is an arbitrary scaling factor (sometimes called the *radius* of the cylinder) that can be set to $s = f$ to minimize the distortion (scaling) near the center of the image.[16] The inverse of this mapping equation is given by

$$x = f\tan\theta = f\tan\frac{x'}{s}, \tag{8.48}$$

$$y = h\sqrt{x^2 + f^2} = \frac{y'}{s}f\sqrt{1 + \tan^2 x'/s} = f\frac{y'}{s}\sec\frac{x'}{s}. \tag{8.49}$$

8.2 Image Stitching

8.2.6 Cylindrical and spherical coordinates



(a)

8.2 Image Stitching

8.2.6 Cylindrical and spherical coordinates

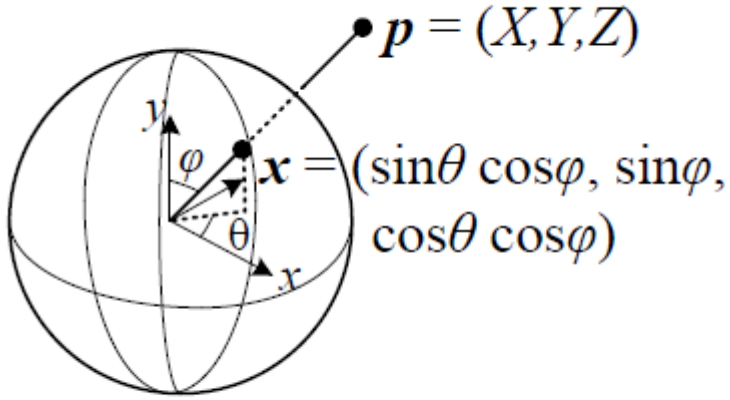
- Images can also be projected onto a spherical surface (Szeliski and Shum 1997), which is useful if the final panorama includes a full sphere or hemisphere of views, instead of just a cylindrical strip.

- In this case, the sphere is parameterized by two angles (θ, ϕ) spherical coordinates given by

$$(\sin\theta\cos\phi, \sin\phi, \cos\theta\cos\phi) \propto (x, y, f), \tag{8.50}$$

8.2 Image Stitching

8.2.6 Cylindrical and spherical coordinates



(b)

8.2 Image Stitching

8.2.6 Cylindrical and spherical coordinates

as shown in Figure 8.8b.[17] The correspondence between coordinates is now given by (Szeliski and Shum 1997):

$$x' = s\theta = s \tan^{-1} \frac{x}{f}, \tag{8.51}$$

$$y' = s\phi = s \tan^{-1} \frac{y}{\sqrt{x^2 + f^2}}, \tag{8.52}$$

while the inverse is given by

$$x = f \tan\theta = f \tan \frac{x'}{s}, \tag{8.53}$$

$$y = \sqrt{x^2 + f^2} \tan\phi = \tan \frac{y'}{s} f \sqrt{1 + \tan^2 x'/s} = f \tan \frac{y'}{s} \sec \frac{x'}{s}. \tag{8.54}$$

8.2 Image Stitching

8.2.6 Cylindrical and spherical coordinates

(a) (b)

Figure 8.9 *A cylindrical panorama (Szeliski and Shum 1997) © 1997 ACM: (a) two cylindrically warped images related by a horizontal translation; (b) part of a cylindrical panorama composited from a sequence of images.*

8.3 Global Alignment

- In this section, we extend the pairwise matching criteria (8.2, 9.1, and 9.43) to a global energy function that involves all of the per-image pose parameters (Section 8.3.1).
- Once we have computed the global alignment, we often need to perform local adjustments, such as parallax removal, to reduce double images and blurring due to local mis-registrations (Section 8.3.2).
- Finally, if we are given an unordered set of images to register, we need to discover which images go together to form one or more panoramas.
- This process of panorama recognition is described in Section 8.3.3.

8.3 Global Alignment

8.3.1 Bundle adjustment

Consider the feature-based alignment problem given in Equation (8.2), i.e.,

$$E_{\text{pairwise-LS}} = \sum_i \|\mathbf{r}_i\|^2 = \|\tilde{\mathbf{x}}'_i(\mathbf{x}_i; \mathbf{p}) - \hat{\mathbf{x}}'_i\|^2. \tag{8.58}$$

For multi-image alignment, instead of having a single collection of pairwise feature correspondences, $\{(\mathbf{x}_i, \hat{\mathbf{x}}'_i)\}$, we have a collection of n features, with the location of the ith feature point in the jth image denoted by $\mathbf{x}_{ij}$ and its scalar confidence (i.e., inverse variance) denoted by c_{ij}.[20] Each image also has some associated pose parameters.

8.3 Global Alignment

8.3.1 Bundle adjustment

- In this section, we assume that the pose consists of a rotation matrix $\mathbf{R}_j$ and a focal length f_j, although formulations in terms of homographies are also possible
- The equation mapping a 3D point $\mathbf{x}_i$ into a point $\mathbf{x}_{ij}$ in frame j can be re-written from Equations (2.68) and (8.38) as

$$\tilde{\mathbf{x}}_{ij} \sim \mathbf{K}_j \mathbf{R}_j \mathbf{x}_i \quad \text{and} \quad \mathbf{x}_i \sim \mathbf{R}_j^{-1} \mathbf{K}_j^{-1} \tilde{\mathbf{x}}_{ij}, \tag{8.59}$$

where $\mathbf{K}_j = \text{diag}(f_j, f_j, 1)$ is the simplified form of the calibration matrix. The motion mapping a point $\mathbf{x}_{ij}$ from frame j into a point $\mathbf{x}_{ik}$ in frame k is similarly given by

$$\tilde{\mathbf{x}}_{ik} \sim \tilde{\mathbf{H}}_{kj} \tilde{\mathbf{x}}_{ij} = \mathbf{K}_k \mathbf{R}_k \mathbf{R}_j^{-1} \mathbf{K}_j^{-1} \tilde{\mathbf{x}}_{ij}. \tag{8.60}$$

8.3 Global Alignment

8.3.1 Bundle adjustment

- An alternative way to formulate the optimization is to use true bundle adjustment, i.e., to solve not only for the pose parameters $\{(\mathbf{R}_j, f_j)\}$ but also for the 3D point positions $\{\mathbf{x}_i\}$.

$$E_{\mathrm{BA-2D}} = \sum_i \sum_j c_{ij} \|\tilde{\mathbf{x}}_{ij}(\mathbf{x}_i; \mathbf{R}_j, f_j) - \hat{\mathbf{x}}_{ij}\|^2, \tag{8.62}$$

- An alternative formulation is to minimize the error in 3D projected ray directions (Shum and Szeliski 2000), i.e.,

$$E_{\mathrm{BA-3D}} = \sum_i \sum_j c_{ij} \|\tilde{\mathbf{x}}_i(\hat{\mathbf{x}}_{ij}; \mathbf{R}_j, f_j) - \mathbf{x}_i\|^2, \tag{8.63}$$

where $\tilde{\mathbf{x}}_i(\mathbf{x}_{ij}; \mathbf{R}_j, f_j)$ is given by the second half of (8.59).

8.3 Global Alignment

8.3.1 Bundle adjustment

- However, if we eliminate the 3D rays xi, we can derive a pairwise energy formulated in 3D ray space (Shum and Szeliski 2000)

$$E_{\text{all-pairs-3D}} = \sum_i \sum_{jk} c_{ij} c_{ik} \|\tilde{\mathbf{x}}_i(\hat{\mathbf{x}}_{ij}; \mathbf{R}_j, f_j) - \tilde{\mathbf{x}}_i(\hat{\mathbf{x}}_{ik}; \mathbf{R}_k, f_k)\|^2. \quad (8.64)$$

8.3 Global Alignment

8.3.2 Parallax Removal

- Once we have optimized the global orientations and focal lengths of our cameras, we may find that the images are still not perfectly aligned, i.e., the resulting stitched image looks blurry or ghosted in some places.

- This can be caused by a variety of factors, including unmodeled radial distortion, 3D parallax (failure to rotate the camera around its front nodal point), small scene motions such as waving tree branches, and large-scale scene motions such as people moving in and out of pictures

8.3 Global Alignment

8.3.2 Parallax Removal

- 3D parallax can be handled by doing a full 3D bundle adjustment, i.e., by replacing the projection, which models camera translations.
- The 3D positions of the matched feature points and cameras can then be simultaneously recovered, although this can be significantly more expensive than parallax-free image registration.
- Once the 3D structure has been recovered, the scene could (in theory) be projected to a single (central) viewpoint that contains no parallax.

8.3 Global Alignment

8.3.2 Parallax Removal

(a) (b) (c)

Figure 8.10 *Deghosting a mosaic with motion parallax (Shum and Szeliski 2000) © 2000 IEEE: (a) composite with parallax; (b) after a single deghosting step (patch size 32); (c) after multiple steps (sizes 32, 16 and 8).*

8.3 Global Alignment

8.3.3 Parallax Removal

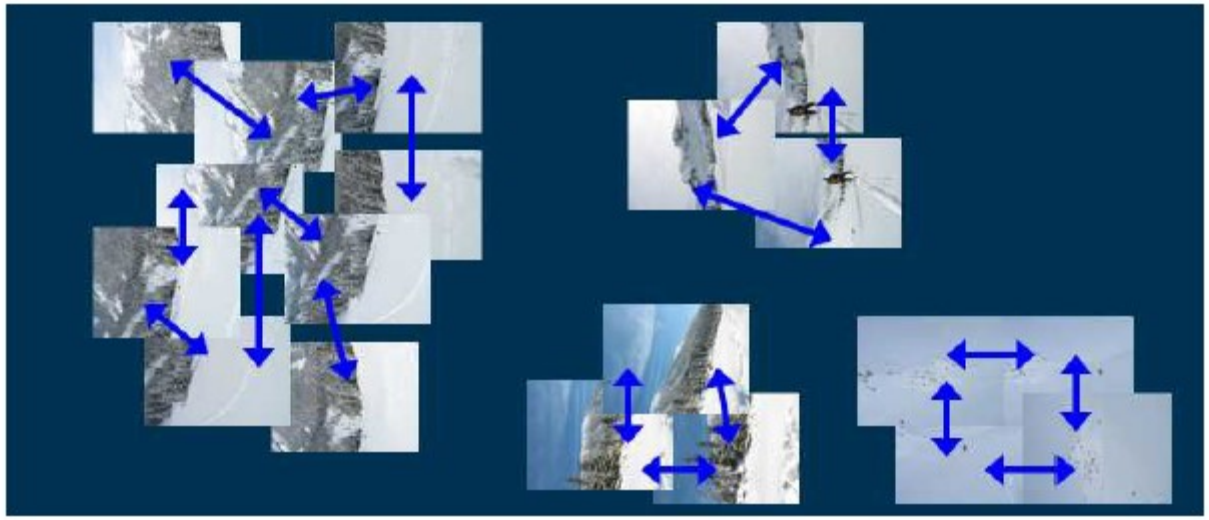
- The final piece needed to perform fully automated image stitching is a technique to recognize which images actually go together, which Brown and Lowe (2007) call recognizing panoramas.

- To recognize panoramas, Brown and Lowe (2007) first find all pairwise image overlaps using a feature-based method and then find connected components in the overlap graph to "recognize" individual panoramas (Figure 8.11)

8.3 Global Alignment

8.3.3 Parallax Removal

(a)



(b)

8.3 Global Alignment

8.3.3 Parallax Removal

(c)

Figure 8.11 *Recognizing panoramas (Brown, Szeliski, and Winder 2005), figures courtesy of Matthew Brown: (a) input images with pairwise matches; (b) images grouped into connected components (panoramas); (c) individual panoramas registered and blended into stitched composites.*

8.3 Global Alignment

8.3.3 Parallax Removal

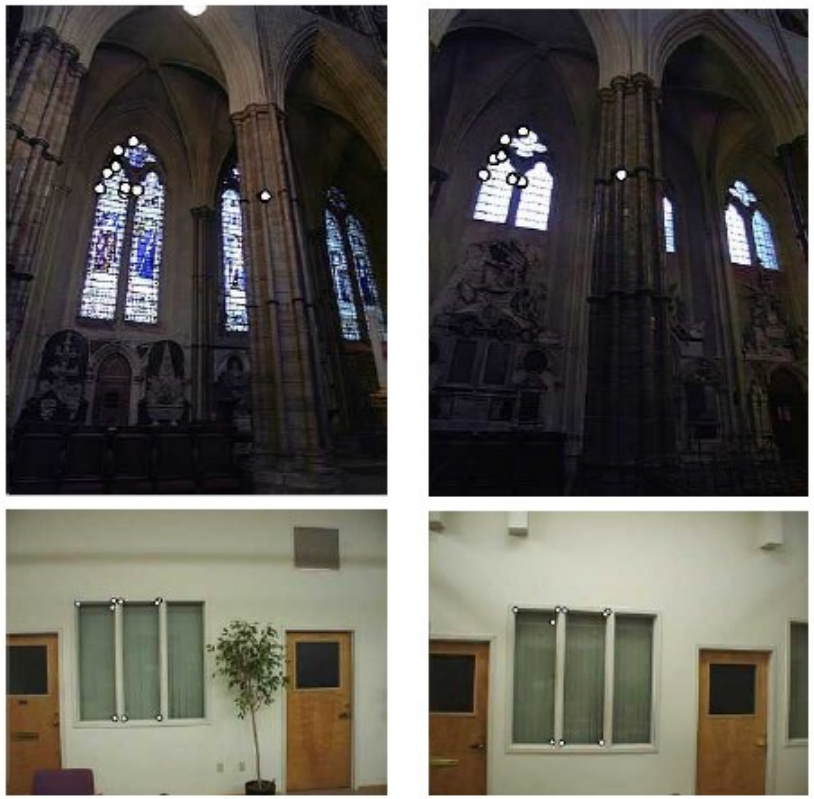


Figure 8.12 *Matching errors (Brown, Szeliski, and Winder 2004): accidental matching of several features can lead to matches between pairs of images that do not actually overlap.*

8.3 Global Alignment

8.3.3 Parallax Removal

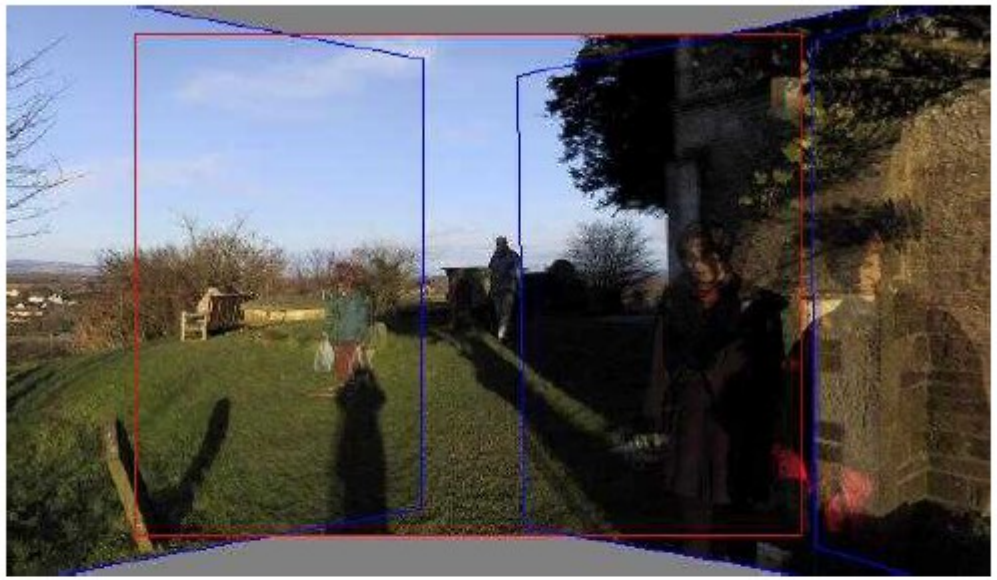


Figure 8.13 *Validation of image matches by direct pixel error comparison can fail when the scene contains moving objects (Uyttendaele, Eden, and Szeliski 2001) © 2001 IEEE.*

8.4 Compositing

- Once we have registered all of the input images with respect to each other, we need to decide how to produce the final stitched mosaic image.
- This involves selecting a final compositing surface (flat, cylindrical, spherical, etc.) and view (reference image).
- It also involves selecting which pixels contribute to the final composite and how to optimally blend these pixels to minimize visible seams, blur, and ghosting.

8.4 Compositing

8.4.1 Choosing a compositing surface

- The first choice to be made is how to represent the final image.
- If only a few images are stitched together, a natural approach is to select one of the images as the reference and to then warp all of the other images into its reference coordinate system.
- The resulting composite is sometimes called a flat panorama, since the projection onto the final surface is still a perspective projection, and hence straight lines remain straight (which is often a desirable attribute)
- For larger fields of view, however, we cannot maintain a flat representation without excessively stretching pixels near the border of the image.

8.4 Compositing

8.4.1 Choosing a compositing surface

- The first choice to be made is how to represent the final image.
- If only a few images are stitched together, a natural approach is to select one of the images as the reference and to then warp all of the other images into its reference coordinate system.
- The resulting composite is sometimes called a flat panorama, since the projection onto the final surface is still a perspective projection, and hence straight lines remain straight (which is often a desirable attribute)

8.4 Compositing

8.4.1 Choosing a compositing surface

- For larger fields of view, however, we cannot maintain a flat representation without excessively stretching pixels near the border of the image.
- The usual choice for compositing larger panoramas is to use a cylindrical or spherical projection.
- In fact, any surface used for environment mapping in computer graphics can be used, including a cube map, which represents the full viewing sphere with the six square faces of a cube.

8.4 Compositing

8.4.1 Choosing a compositing surface

- View selection
- Coordinate transformations
- Sampling issues

8.4 Compositing

8.4.1 Choosing a compositing surface

View selection

- Once we have chosen the output parameterization, we still need to determine which part of the scene will be centered in the final view. As mentioned above, for a flat composite, we can choose one of the images as a reference.

- For example, for rotational panoramas represented as a collection of 3D rotation matrices, we can choose the image whose z-axis is closest to the average z-axis (assuming a reasonable field of view).

8.4 Compositing

8.4.1 Choosing a compositing surface

Coordinate transformations

- After selecting the parameterization and reference view, we still need to compute the mappings between the input and output pixels coordinates.
- If the final compositing surface is flat (e.g., a single plane or the face of a cube map) and the input images have no radial distortion, the coordinate transformation is the simple homography described by (8.38).
- If the final composite surface has some other analytic form (e.g., cylindrical or spherical), we need to convert every pixel in the final panorama into a viewing ray (3D point) and then map it back into each image according to the projection (and optionally radial distortion) equations.

8.4 Compositing

8.4.1 Choosing a compositing surface

Sampling issues

- While the above computations can yield the correct (fractional) pixel addresses in each input image, we still need to pay attention to sampling issues.
- For example, if the final panorama has a lower resolution than the input images, pre-filtering the input images is necessary to avoid aliasing
- The basic problem is to compute the appropriate pre-filter, which depends on the distance (and arrangement) between neighboring samples in a source image.
- As discussed in Sections 3.5.2 and 3.6.1, various approximate solutions, such as MIP mapping (Williams 1983) or elliptically weighted Gaussian averaging (Greene and Heckbert 1986) have been developed in the graphics community

8.4 Compositing

8.4.2 Pixel selection and weighting (de-ghosting)

- Once the source pixels have been mapped onto the final composite surface, we must still decide how to blend them in order to create an attractive-looking panorama.
- If all of the images are in perfect registration and identically exposed, this is an easy problem, i.e., any pixel or combination will do.
- However, for real images, visible seams (due to exposure differences), blurring (due to mis-registration), or ghosting (due to moving objects) can occur.

8.4 Compositing

8.4.2 Pixel selection and weighting (de-ghosting)

Feathering and center-weighting

- When an image has some cutout regions, down-weighting pixels near the edges of both cutouts and the image is preferable.
- This can be done by computing a distance map or grassfire transform.

$$w_k(\mathbf{x}) = \arg\min_{\mathbf{y}} \{\|\mathbf{y}\| \mid \tilde{I}_k(\mathbf{x}+\mathbf{y}) \text{ is invalid } \}, \tag{8.71}$$

where each valid pixel is tagged with its Euclidean distance to the nearest invalid pixel (Section 3.3.3). The Euclidean distance map can be efficiently computed using a two-pass raster algorithm (Danielsson 1980; Borgefors 1986).

8.4 Compositing

8.4.2 Pixel selection and weighting (de-ghosting)

Feathering and center-weighting

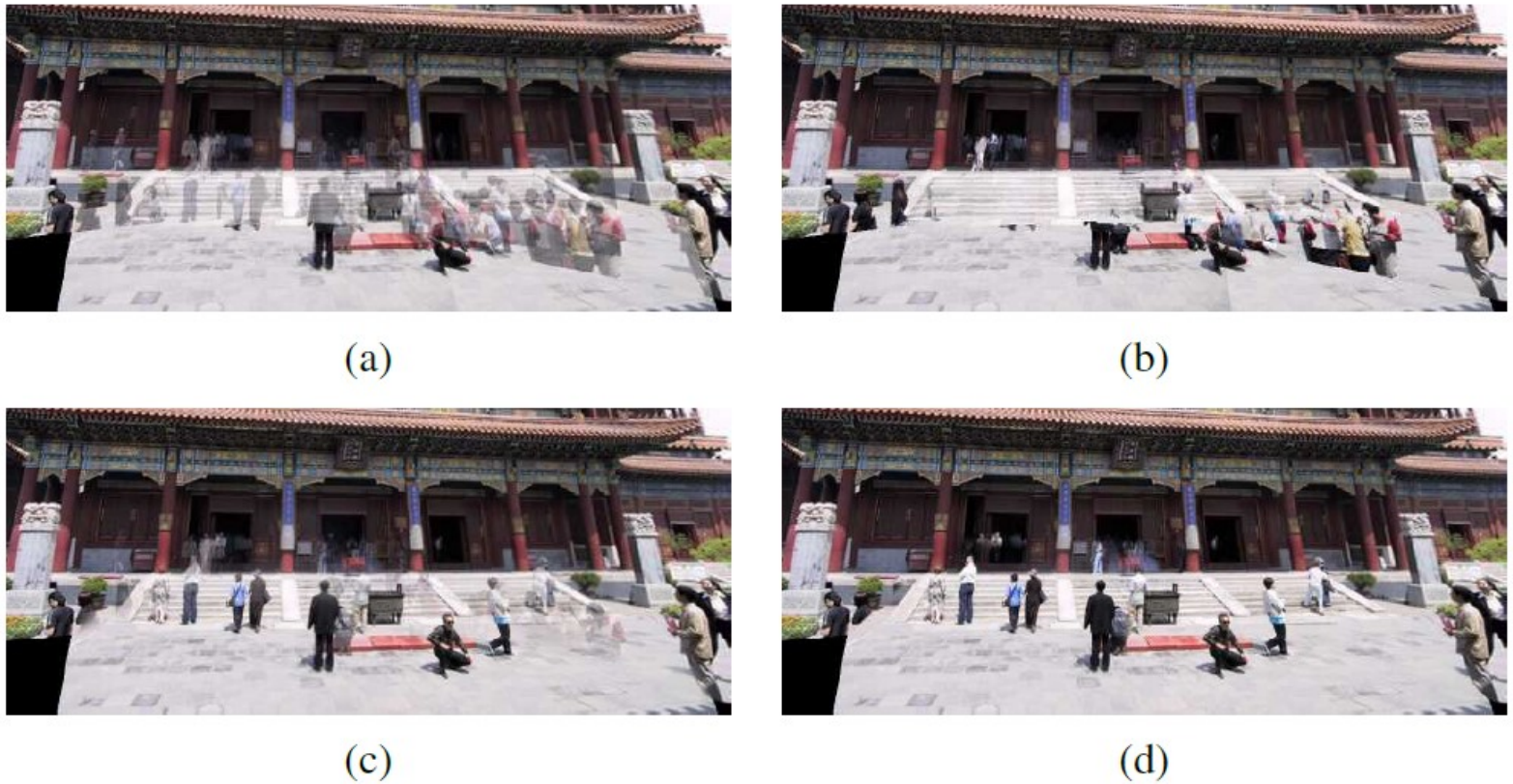
- The simplest way to create a final composite is to simply take an average value at each pixel,

$$C(\mathbf{x}) = \sum_k w_k(\mathbf{x}) \tilde{I}_k(\mathbf{x}) \Big/ \sum_k w_k(\mathbf{x}) \ , \tag{8.70}$$

where $\tilde{I}_k(\mathbf{x})$ are the *warped* (re-sampled) images and $w_k(\mathbf{x})$ is 1 at valid pixels and 0 elsewhere. On computer graphics hardware, this kind of summation can be performed in an *accumulation buffer* (using the A channel as the weight).

8.4 Compositing

8.4.2 Pixel selection and weighting (de-ghosting)



(a) (b) (c) (d)

8.4 Compositing

8.4.2 Pixel selection and weighting (de-ghosting)

(e) (f)

(g) (h)

Figure 8.14 *Final composites computed by a variety of algorithms (Szeliski 2006a): (a) average, (b) median, (c) feathered average, (d)* p-norm $p = 10$, *(e) Voronoi, (f) weighted ROD vertex cover with feathering, (g) graph cut seams with Poisson blending and (h) with pyramid blending.*

8.4 Compositing

8.4.2 Pixel selection and weighting (de-ghosting)

Optimal seam selection

- Computing the Voronoi diagram is one way to select the seams between regions where different images contribute to the final composite.
- However, Voronoi images totally ignore the local image structure underlying the seam.
- A better approach is to place the seams in regions where the images agree, so that transitions from one source to another are not visible. In this way, the algorithm avoids "cutting through" moving objects where a seam would look unnatural (Davis 1998).
- For a pair of images, this process can be formulated as a simple dynamic program starting from one edge of the overlap region and ending at the other

8.4 Compositing

8.4.3 Photomontage

- While image stitching is normally used to composite partially overlapping photographs, it can also be used to composite repeated shots of a scene taken with the aim of obtaining the best possible composition and appearance of each element.

8.4 Compositing

8.4.3 Photomontage

Figure 8.16 *Photomontage (Agarwala, Dontcheva* et al. *2004) © 2004 ACM. From a set of five source images (of which four are shown on the left), Photomontage quickly creates a composite family portrait in which everyone is smiling and looking at the camera (right). Users simply flip through the stack and coarsely draw strokes using the designated source image objective over the people they wish to add to the composite. The user-applied strokes and computed regions (middle) are color-coded by the borders of the source images on the left.*

8.4 Compositing

8.4.4 Blending

- Once the seams between images have been determined and unwanted objects removed, we still need to blend the images to compensate for exposure differences and other misalignments.

- The spatially varying weighting (feathering) previously discussed can often be used to accomplish this.

8.4 Compositing

8.4.4 Blending

Laplacian pyramid blending

- Instead of using a single transition width, a frequency-adaptive width is used by creating a band-pass (Laplacian) pyramid and making the transition widths within each level a function of the level, i.e., the same width in pixels

8.4 Compositing

8.4.4 Blending

Gradient domain blending

- An alternative approach to multi-band image blending is to perform the operations in the gradient domain.
- Perez, Gangnet, and Blake (2003) show how gradient domain reconstruction can be used to do seamless object insertion in image editing applications

8.4 Compositing

8.4.4 Blending

Gradient domain blending

(a) (b) (c)

Figure 8.18 *Poisson image editing (Pérez, Gangnet, and Blake 2003) © 2003 ACM: (a) The dog and the two children are chosen as source images to be pasted into the destination swimming pool. (b) Simple pasting fails to match the colors at the boundaries, whereas (c) Poisson image blending masks these differences.*

8.4 Compositing

8.4.4 Blending

Exposure compensation

- Pyramid and gradient domain blending can do a good job of compensating for moderate amounts of exposure differences between images.
- However, when the exposure differences become large, alternative approaches may be necessary

8.4 Compositing

8.4.4 Blending

Exposure compensation

- Uyttendaele, Eden, and Szeliski (2001) iteratively estimate a local correction between
- each source image and a blended composite.

- First, a block-based quadratic transfer function is fit between each source image and an initial feathered composite.

- Next, transfer functions are averaged with their neighbors to get a smoother mapping and per-pixel transfer functions are computed by splining (interpolating) between neighboring block values.

- Once each source image has been smoothly adjusted, a new feathered composite is computed and the process is repeated (typically three times).

Thank You