Chapter 9
Transformation Functions

A transformation function uses the coordinates of corresponding control points in two images to estimate the geometric relation between the images, which is then used to transform the geometry of one image to that of the other to spatially aligned the images. Spatial alignment of images makes it possible to determine correspondence between points in overlapping areas in the images. This correspondence is needed in various image analysis applications, such as stereo depth perception, change detection, and information fusion.

Given the coordinates of n corresponding points in two images:

$$\big\{(x_i, y_i), (X_i, Y_i) : i = 1, \ldots, n\big\}, \tag{9.1}$$

we would like to find a transformation function with components f_x and f_y that satisfies

$$\begin{aligned} X_i &\approx f_x(x_i, y_i), \\ Y_i &\approx f_y(x_i, y_i), \end{aligned} \quad i = 1, \ldots, n. \tag{9.2}$$

f_x is a single-valued function that approximates 3-D points

$$\big\{(x_i, y_i, X_i) : i = 1, \ldots, n\big\}, \tag{9.3}$$

and f_y is another single-valued function that approximates 3-D points

$$\big\{(x_i, y_i, Y_i) : i = 1, \ldots, n\big\}. \tag{9.4}$$

Each component of a transformation function is, therefore, a single-valued surface fitting to a set of 3-D points, representing the coordinates of control points in the reference image and the X- or the Y-component of corresponding control points in the sensed image. Many surface-fitting methods exist in the literature that can be chosen for this purpose. In this chapter, functions most suitable for the registration of images with local geometric differences will be examined.

If the type of transformation function relating the geometries of two images is known, the parameters of the transformation can be determined from the coordinates of corresponding points in the images by a robust estimator (Chap. 8). For example, if the images to be registered represent consecutive frames in an aerial

A.A. Goshtasby, *Image Registration*,
Advances in Computer Vision and Pattern Recognition,
DOI 10.1007/978-1-4471-2458-0_9,

video captured by a platform at a high altitude, the images will have translational and small rotational differences. The transformation function to register such images has only a few parameters, and knowing a number of corresponding points in the images, the parameters of the transformation can be determined. If the geometric relation between the images is not known, a transformation function is required that uses information present among the correspondences to adapt to the local geometric differences between the images.

In the following sections, first transformation functions that have a fixed number of parameters are discussed. These are *well-known transformation functions* that describe the global geometric relations between two images. Next, *adaptive transformations* that adapt to local geometric differences between images are discussed. The number of parameters in a component of an adaptive transformation varies with the severity of the geometric difference between two images and can be as high as the number of corresponding points. At the end of this chapter, the properties of various transformation functions will be reviewed, and their performances in registration of images with varying degrees of geometric differences will be measured and compared.

9.1 Well-Known Transformation Functions

9.1.1 Translation

If the sensed image is only translated with respect to the reference image, corresponding points in the images will be related by

$$X = x + h, \tag{9.5}$$

$$Y = y + k. \tag{9.6}$$

In matrix form, this can be written as

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & h \\ 0 & 1 & k \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{9.7}$$

or simply by

$$\mathbf{P} = \mathbf{T}\mathbf{p}. \tag{9.8}$$

$\mathbf{P}$ and $\mathbf{p}$ are homogeneous coordinates of corresponding points in the sensed and reference images, respectively, and $\mathbf{T}$ is the transformation matrix showing that the sensed image is translated with respect to the reference image by (h, k).

By knowing one pair of corresponding points in the images, parameters h and k can be determined by substituting the coordinates of the points into (9.5) and (9.6) and solving the obtained system of equations for h and k. If two or more corresponding points are available, h and k are determined by one of the robust estimators discussed in the previous chapter. A robust estimator can determine the

parameters of the transformation if some of the correspondences are incorrect. If the correspondences are known to be correct, the parameters can also be determined by the ordinary least-squares method [84].

9.1.2 Rigid

When the sensed image is translated and rotated with respect to the reference image, the distance between points and the angle between lines remain unchanged from one image to another. Such a transformation is known as *rigid* or *Euclidean* transformation and can be written as

$$X = x\cos\theta - y\sin\theta + h, \tag{9.9}$$

$$Y = x\sin\theta + y\cos\theta + k. \tag{9.10}$$

In matrix form, this will be

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & h \\ 0 & 1 & k \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{9.11}$$

or simply

$$\mathbf{P} = \mathbf{TRp}. \tag{9.12}$$

θ shows the difference in orientation of the sensed image with respect to the reference image when measured in the counter-clockwise direction. The coordinates of a minimum of two corresponding points in the images are required to determine parameters θ, h, and k. From a pair of points in each image, a line is obtained. The angle between the lines in the images determines θ. Knowing θ, by substituting the coordinates of the midpoints of the lines into (9.9) and (9.10) parameters h and k are determined.

If more than two corresponding points are available, parameters θ, h, and k are determined by one of the robust methods discussed in the previous chapter. For instance, if the RM estimator is used, parameter θ is calculated for various corresponding lines and the median angle is taken as the estimated angle. Knowing θ, parameters h and k are estimated by substituting corresponding points into (9.9) and (9.10), solving the obtained equations, and taking the median of h values and the median of k values as estimations to h and k.

9.1.3 Similarity

When the sensed image is translated, rotated, and scaled with respect to the reference image, coordinates of corresponding points in the images will be related by

the *similarity transformation*, also known as the *transformation of the Cartesian coordinate system*, defined by

$$X = xs\cos\theta - ys\sin\theta + h, \tag{9.13}$$

$$Y = xs\sin\theta + ys\cos\theta + k, \tag{9.14}$$

where s shows scale, θ shows orientation, and (h, k) shows location of the coordinate system origin of the sensed image with respect to that of the reference image.

In matrix form, this can be written as

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & h \\ 0 & 1 & k \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{9.15}$$

or simply by

$$\mathbf{P} = \mathbf{TRSp}. \tag{9.16}$$

Under the similarity transformation, the angle between corresponding lines in the images remains unchanged. Parameters s, θ, h, and k are determined by knowing a minimum of two corresponding points in the images. The scale of the sensed image with respect to the reference image is determined using the ratio of the length of the line segment obtained from the two points in the sensed image over the length of the same line segment obtained in the reference image. Knowing s, parameters θ, h, and k are determined in the same way these parameters were determined under the rigid transformation.

If more than two corresponding points in the images are available, parameters s, θ, h, and k are determined by one of the robust methods discussed in the preceding chapter. For example, if the RM estimator is used, an estimation to parameter s is made by determining s for all combinations of two corresponding points in the images, ordering the obtained s values, and taking the mid value. Knowing s, parameters θ, h, and k are determined in the same way these parameters were determined under the rigid transformation.

9.1.4 Affine

Images that have translational, rotational, scaling, and shearing differences preserve parallelism. Such a transformation is defined by

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & h \\ 0 & 1 & k \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \alpha & 0 \\ \beta & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{9.17}$$

or by

$$\mathbf{P} = \mathbf{TRSEp}. \tag{9.18}$$

An affine transformation has six parameters and can be written as a combination of a linear transformation and a translation. That is,

$$X = a_1 x + a_2 y + a_3, \tag{9.19}$$

$$Y = a_4 x + a_5 y + a_6. \tag{9.20}$$

In matrix form, this can be written as

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{9.21}$$

or

$$\mathbf{P} = \mathbf{L}\mathbf{p}. \tag{9.22}$$

The two components of the transformation defined by (9.17) depend on each other while the two components of the transformation defined by (9.21) are independent of each other. Since transformation (9.17) is constrained by $\sin^2\theta + \cos^2\theta = 1$, it cannot represent all the transformations (9.21) can define. Therefore, the affine transformation allows more differences between two images than translation, rotation, scaling, and shearing. Use of affine transformation in image registration in 2-D and higher dimensions has been studied by Nejhum et al. [70].

To find the best affine transformation when $n > 3$ correspondences are available, a robust estimator should be used. For instance, if the RM estimator is available, from various combinations of 3 correspondences, the parameters of the transformation are determined. Then the median value obtained for each parameter is taken as a robust estimation to that parameter.

9.1.5 Projective

Projective transformation, also known as *homography*, describes the true imaging geometry. Corresponding points in a flat scene and its image, or corresponding points in two images of a flat scene, are related by a projective transformation. Under the projective transformation, straight lines remain straight. A projective transformation is defined by

$$X = \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1}, \tag{9.23}$$

$$Y = \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1}. \tag{9.24}$$

In matrix form, this can be written as

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{9.25}$$

or simply by

$$\mathbf{P} = \mathbf{H}\mathbf{p}. \tag{9.26}$$

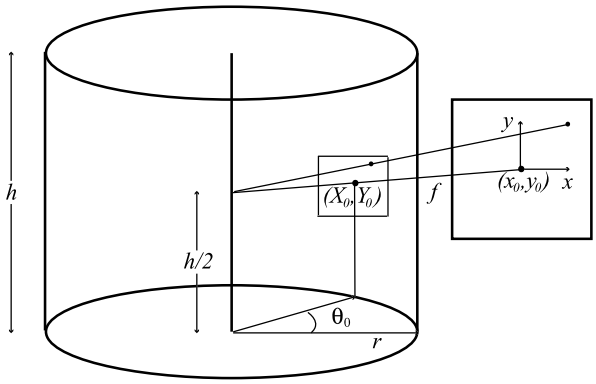


Fig. 9.1 The relation between cylindrical and planar image coordinates. h is the height and r is the radius of the cylindrical image. f is the focal length of the regular camera capturing the planar image, and (x_0, x_0) and (X_0, Y_0) are the intersections of the optical axis of the regular camera with the planar and cylindrical images, respectively

Images of a flat scene, or images of a 3-D scene taken from a distance where the heights of objects in the scene are negligible when compared with the distances of the cameras to the scene, are related by the projective transformation. A projective transformation has 8 parameters, requiring a minimum of 4 corresponding points in the images to determine them. The components of a projective transformation are interdependent due to the common denominator in (9.23) and (9.24). By substituting each corresponding point pair from the images into (9.23) and (9.24), two linear equations in terms of the unknown parameters are obtained. Having 4 corresponding points in the image, a system of 8 linear equations are obtained, from which the 8 parameters of the transformation can be determined.

Since the components of a projective transformation are interdependent, if more than 4 corresponding points are available, the residuals calculated by a robust estimator should include errors from both components as described by (8.17) in the preceding chapter.

9.1.6 Cylindrical

Suppose a cylindrical image of an environment is taken by a virtual camera with its center located in the middle of the axis of the cylinder. Also, suppose the camera has infinite optical axes that fall in a plane passing through the center of the camera and normal to the axis of the cylinder. A cylindrical image obtained in this manner can be saved as a rectangular image XY by letting $X = r\theta$ represent the image columns and $Y = i$ represent the image rows (Fig. 9.1). r is the radius of the cylinder and i varies between 0 and $h - 1$ in the discrete domain. Although such a camera does not exist in real life, images can be created that appear as if obtained by such a camera. To create a cylindrical image, images taken by a regular camera with its center fixed at the center of the cylinder and rotating about the axis of the cylinder are needed.

Suppose an image taken by a regular camera from view angle θ_0, as shown in Fig. 9.1, is available. If the optical axis of the regular camera is normal to the axis of the cylinder, the planar image will be parallel to the axis of the cylinder. The

(a) (b) (c)

Fig. 9.2 (**a**) A planar image of dimensions 256 × 256 and its corresponding cylindrical images (**b**) when $\theta_0 = 0$ and (**c**) when $\theta_0 = \pi/2$ in clockwise direction (or $-\pi/2$ in counter-clockwise direction). In these examples, $r = 128$, $h = 256$, and $f = 128$, all in pixels

coordinates of the center of the planar image (x_0, y_0) define the point where the optical axis of the regular camera intersects the planar image. Suppose this point maps to the cylindrical image at (X_0, Y_0). Then (X_0, Y_0) can be defined in terms of the radius of the cylinder r, the viewing angle θ_0, and the height of the cylinder h:

$$X_0 = r\theta_0, \tag{9.27}$$

$$Y_0 = h/2. \tag{9.28}$$

If the focal length of the regular camera is f, from the geometry in Fig. 9.1, we can write the following relations between the coordinates of a point (x, y) in the planar image and the coordinates of the corresponding point (X, Y) in the cylindrical image:

$$\frac{x - x_0}{f} = \tan\left(\frac{X}{r} - \theta_0\right), \tag{9.29}$$

$$\frac{Y - Y_0}{r} = \frac{y - y_0}{\sqrt{f^2 + (x - x_0)^2}}, \tag{9.30}$$

or

$$X = r\left\{\theta_0 + \tan^{-1}\left(\frac{x - x_0}{f}\right)\right\}, \tag{9.31}$$

$$Y = \frac{h}{2} + \frac{r(y - y_0)}{\sqrt{f^2 + (x - x_0)^2}}. \tag{9.32}$$

Therefore, given the coordinates of a point (x, y) in the planar image, we can find the coordinates of the corresponding point (X, Y) in the cylindrical image. Inversely, given the coordinates of a point (X, Y) in the cylindrical images, we can find the coordinates of the corresponding point (x, y) in the planar image from

$$x = x_0 + f\tan\left(\frac{X}{r} - \theta_0\right), \tag{9.33}$$

$$y = y_0 + \frac{Y - h/2}{r}\sqrt{f^2 + (x - x_0)^2}. \tag{9.34}$$

Using the planar image of dimensions 256 × 256 in Fig. 9.2a, the corresponding cylindrical image shown in Fig. 9.2b is obtained when letting $\theta_0 = 0$, $h = 256$, $r = 128$, and $f = 128$, all in pixel units. Changing the view angle to $\theta_0 = \pi/2$, the image shown in Fig. 9.2c is obtained. Note that in the above formulas, angle

θ increases in the clockwise direction. If θ is increased in the counter-clockwise direction, the cylindrical image will be vertically flipped with respect to the planar image.

If n planar images are taken with view angles $\theta_1, \ldots, \theta_n$, the images can be mapped to the cylindrical image and combined using formulas (9.33) and (9.34). For each regular image, mapping involves scanning the cylindrical image and for each pixel (X, Y) determining the corresponding pixel (x, y) in the planar image, reading the intensity there, and saving it at (X, Y). Since each planar image may cover only a small portion of the cylindrical image, rather than scanning the entire cylindrical image for each planar image, the midpoints of the four sides of the regular image are found in the cylindrical image using (9.31) and (9.32). Then the smallest bounding rectangle with horizontal and vertical sides is determined. This bounding rectangle will contain the entire image; therefore, the cylindrical image is scanned only within the bounding rectangle to find pixels in the planar image to map to the cylindrical image.

These formulas can be used to combine images captured from a fixed viewpoint and at different view angles to a cylindrical image. If gaps appear within the cylindrical image, and if the X-coordinate of the center of the gap is X_0, from (9.27), the view angle $\theta_0 = X_0/r$ can be determined and an image with that view angle obtained and mapped to the cylindrical image to fill the gap. The process can be repeated in this manner until all gaps are filled.

Formulas (9.31) and (9.32) can be used to map the cylindrical image to a planar image from any view angle θ_0. When planar images obtained in this manner are projected to planar screens of height h and at distance r to a viewer of height $h/2$ standing at the middle of the cylinder, the viewer will see a surround view of the environment without any geometric distortion. The cylindrical image can, therefore, be used as a means to visualize a distortion-free surround image of an environment through planar imaging and planar projection.

Note that this visualization does not require that the number of planar images captured and the number of planar projections used in viewing be the same. Therefore, if a number of video cameras are hinged together in such a way that they share the same center and their optical axes lie in the same plane, video frames of a dynamic scene simultaneously captured by the cameras can be combined into a cylindrical video and mapped to a desired number of planar images and projected to planar screens surrounding a viewer. The viewer will then see the dynamic scene from all directions.

9.1.7 Spherical

Consider a spherical image obtained by a virtual camera where the image center coincides with the camera center. Suppose the camera has infinite optical axes, each axis connecting the camera center to a point on the sphere. Points on the spherical image as well as directions of the optical axes can be represented by the angular

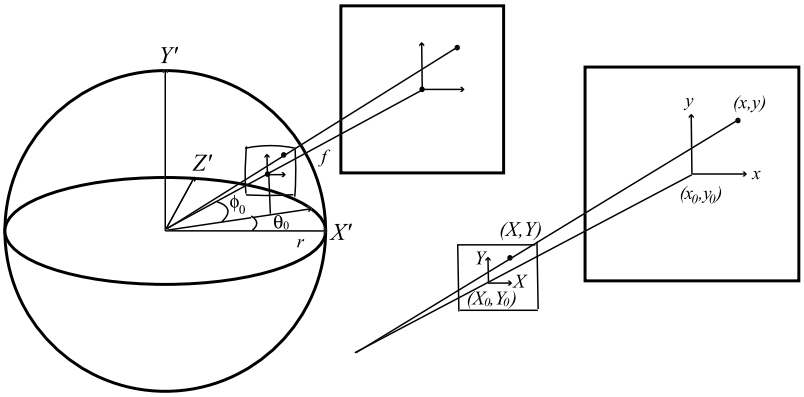


Fig. 9.3 The relation between the spherical image coordinates (X, Y) and the planar image coordinates (x, y). (x_0, y_0) is the center of the planar image and (X_0, Y_0) is the corresponding point in the spherical image. The ray connecting (x_0, y_0) to (X_0, Y_0) passes through the center of the spherical image and is normal to the planar image. θ_0 shows the angle the projection of this ray to the $X'Z'$-plane makes with the X'-axis, and ϕ_0 is the angle this ray makes with the $X'Z'$-plane. $X'Y'Z'$ is the coordinate system of the sphere

coordinates (θ, ϕ). If an image is obtained by a regular camera with an optical axis in direction $(0, 0)$, the relation between this planar image and the spherical image (Fig. 9.3) will be:

$$\theta = \tan^{-1}\left(\frac{x - x_0}{f}\right), \tag{9.35}$$

$$\phi = \tan^{-1}\left(\frac{y - y_0}{f}\right). \tag{9.36}$$

Values at (θ, ϕ) can be saved in an XY array for storage purposes, where

$$X = r\theta = r\tan^{-1}\left(\frac{x - x_0}{f}\right), \tag{9.37}$$

$$Y = r(\phi + \pi/2) = r\left[\tan^{-1}\left(\frac{y - y_0}{f}\right) + \frac{\pi}{2}\right]. \tag{9.38}$$

By varying θ from 0 to 2π and ϕ from $-\pi/2$ to $\pi/2$, and letting r represent the radius of the spherical image in pixel units, the obtained rectangular image (X, Y) will show the spherical image in its entirety.

If the planar image is obtained when the regular camera optical axis was in direction (θ_0, ϕ_0), as shown in Fig. 9.3, we first assume the image is obtained at direction $(0, 0)$, project it to the spherical image, and then shift the spherical image in such a way that its center moves to (θ_0, ϕ_0). This simply implies replacing θ in (9.37) with $\theta + \theta_0$ and ϕ in (9.38) with $\phi + \phi_0$. Therefore,

$$X = r\left[\theta_0 + \tan^{-1}\left(\frac{x_0 - x}{f}\right)\right], \tag{9.39}$$

$$Y = r\left[\phi_0 + \frac{\pi}{2} + \tan^{-1}\left(\frac{y - y_0}{f}\right)\right]. \tag{9.40}$$

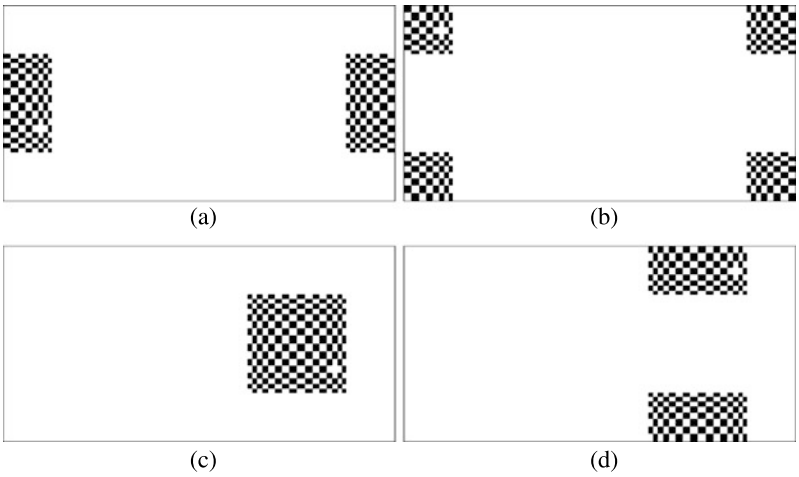


Fig. 9.4 (**a**)–(**d**) Spherical images corresponding to the planar image in Fig. 9.2a when viewing the planar image from directions $(\theta_0, \phi_0) = (0, 0), (0, \pi/2), (\pi/2, 0)$, and $(\pi/2, \pi/2)$, respectively

Conversely, knowing the coordinates (X, Y) of a point in the spherical image, the coordinates of the corresponding point in the planar image when viewed in direction (θ_0, ϕ_0) will be

$$x = x_0 + f \tan\left(\frac{X}{r} - \theta_0\right), \tag{9.41}$$

$$y = y_0 + f \tan\left(\frac{Y}{r} - \phi_0 - \frac{\pi}{2}\right). \tag{9.42}$$

The planar image of Fig. 9.2a when mapped to a spherical image of radius $r = 128$ pixels according to (9.41) and (9.42) with various values of (θ_0, ϕ_0) are shown in Fig. 9.4. Parameter f is set equal to r in these examples.

A rectangular image with XY coordinates and dimensions $2\pi r \times \pi r$ can be created by combining planar images taken at different orientations (θ_0, ϕ_0) of an environment. Having a spherical image created with coordinates (θ, ϕ), or equivalently (X, Y), we can project the spherical image to any plane and create a planar image. Such images, when projected to planes surrounding a viewer, will enable the viewer to see the environment from all directions.

Given a planar image that represents a particular view of a scene, its mapping to the spherical image is obtained by scanning the XY image and for each pixel (X, Y), locating the corresponding pixel (x, y) in the planar image from (9.41) and (9.42). If (x, y) falls inside the planar image, its intensity is read and saved at (X, Y). To avoid scanning XY areas where the planar image is not likely to produce a result, first, a bounding rectangle is found in the XY image where the planar image is mapped. This involves substituting the coordinates of the four corners of the image into (9.39) and (9.40) as (x, y) and finding the corresponding coordinates (X, Y) in the spherical image. This will create a rectangle inside which the planar image will

be mapped. Then the bounding rectangle is scanned to determine the corresponding pixels in the planar image and mapped to the spherical image.

To find the projection of the spherical image to a planar image of a particular size and direction (θ_0, ϕ_0), the planar image is scanned and for each pixel (x, y) the corresponding pixel in the spherical image is located using (9.39) and (9.40). Then intensity at (X, Y) is read and saved at (x, y). Note that when $X > 2\pi r$, because $\theta \pm 2\pi = \theta$, we should let $X = X - 2\pi r$, and when $X < 0$, we should let $X = X + 2\pi r$. Similarly, we should let $\phi = -\pi - \phi$ when $\phi < -\pi/2$, $\phi = \pi - \phi$ when $\phi > \pi/2$, $Y = -Y$ when $Y < 0$, and $Y = 2Y - \pi r$ when $Y > \pi r$.

9.2 Adaptive Transformation Functions

9.2.1 Explicit

An explicit transformation function of variables x and y is defined by

$$F = f(x, y). \tag{9.43}$$

An explicit function produces a single value for each point in the xy domain. An explicit function of variables x and y can be considered a single-valued surface that spans over the xy domain. Therefore, given a set of 3-D points

$$\{(x_i, y_i, F_i) : i = 1, \ldots, n\}, \tag{9.44}$$

an explicit function interpolates the points by satisfying

$$F_i = f(x_i, y_i), \quad i = 1, \ldots, n, \tag{9.45}$$

and approximates the points by satisfying

$$F_i \approx f(x_i, y_i), \quad i = 1, \ldots, n. \tag{9.46}$$

If (x_i, y_i) are the coordinates of the ith control point in the reference image and F_i is the X or the Y coordinate of the corresponding control point in the sensed image, the surface interpolating/approximating points (9.44) will represent the X- or the Y-component of the transformation.

If corresponding points in the images are accurately located, an interpolating function should be used to ensure that the obtained transformation function maps corresponding points to each other. However, if the coordinates of corresponding points contain inaccuracies, approximating functions should be used to smooth the inaccuracies.

A chronological review of approximation and interpolation methods is provided by Meijering [68] and comparison of various approximation and interpolation methods is provided by Franke [25] and Renka and Brown [82]. Bibliography and categorization of explicit approximation and interpolation methods are provided by Franke and Schumaker [28, 93] and Grosse [40].

In the remainder of this chapter, transformation functions that are widely used or could potentially be used to register images with local geometric differences are reviewed.

9.2.1.1 Multiquadrics

Interpolation by radial basis functions is in general defined by

$$f(x, y) = \sum_{i=1}^{n} A_i R_i(x, y). \tag{9.47}$$

Parameters $\{A_i : i = 1, \ldots, n\}$ are determined by letting $f(x_i, y_i) = F_i$ for $i = 1, \ldots, n$ and solving the obtained system of linear equations. $R_i(x, y)$ is a radial function whose value is proportional to the distance between (x, y) and (x_i, y_i). A surface point is obtained from a weighted sum of these radial functions. Powell [75] has provided an excellent review of radial basis functions.

When

$$R_i(x, y) = \left[(x - x_i)^2 + (y - y_i)^2 + d^2\right]^{\frac{1}{2}}, \tag{9.48}$$

$f(x, y)$ represents a multiquadric interpolation [42, 43]. As d^2 is increased, a smoother surface is obtained. In a comparative study carried out by Franke [25], multiquadrics were found to produce the best accuracy in the interpolation of randomly spaced data in the plane when compared with many other interpolation methods.

Multiquadric interpolation depends on parameter d^2. This parameter works like a stiffness parameter and as it is increased, a smoother surface is obtained. The best stiffness parameter for a data set depends on the spacing and organization of the data as well as on the data gradient. Carlson and Foley [10], Kansa and Carlson [47], and Franke and Nielson [29] have studied the role parameter d^2 plays on multiquadric interpolation accuracy.

An example of the use of multiquadric interpolation in image registration is given in Fig. 9.5. Images (a) and (b) represent multiview images of a partially snow covered rocky mountain. 165 corresponding points are identified in the images using the coarse-to-fine matching Algorithm F5 in Sect. 7.10. Corresponding points in corresponding regions that fall within 1.5 pixels of each other after transformation of a sensed region to align with its corresponding reference region by an affine transformation are chosen and used in the following experiments. About half (83 correspondences) are used to determine the transformation parameters and the remaining half (82 correspondences) are used to evaluate the registration accuracy. Images (a) and (b) will be referred to as the *Mountain image set*.

Resampling image (b) to align with image (a) by multiquadrics using the 83 correspondences (shown in red) produced the image shown in (c) when letting $d = 12$ pixels. Assigning values larger or smaller than 12 to d increases root-mean-squared error (RMSE) at the 82 remaining correspondences. Overlaying of images (a) and (c) is shown in (d). The reference image is shown in the red and blue bands and the resampled sensed image is shown in the green band of a color image. Pixels in the overlaid image where the images register well appear gray, while pixels in the overlaid image where the images are locally shifted with respect to each other appear purple or green. Although registration within the convex hull of the control points may be acceptable, registration outside the convex hull of the control points contain large errors and is not acceptable.

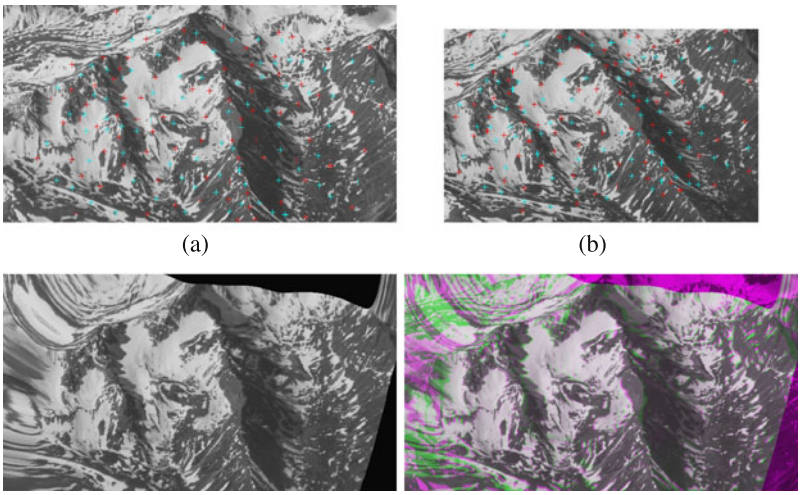


Fig. 9.5 (**a**) Reference and (**b**) sensed images used in image registration. The control points marked in *red* '+' are used to determine the registration parameters. The control points marked in *light blue* '+' are used to determine the registration accuracy. (**c**) Resampling of image (**b**) to align with image (**a**) using multiquadrics with $d = 12$ pixels. (**d**) Overlaying of the reference image (*purple*) and the resampled sensed image (*green*). Areas that are correctly registered appear in *gray*, while misregistered areas appear in *purple* or *green*. The reference image areas where there is no correspondence in the sensed image also appear in *purple* (the color of the reference image)

Multiquadrics use monotonically increasing basis functions. This implies that farther control points affect registration of a local neighborhood more than control points closer to the neighborhood. This is not a desirable property in image registration because we do not want a local error affect registration of distant points and would like to keep the influence of a control point local to its neighborhood. To obtain a locally sensitive transformation function, monotonically decreasing radial basis functions are needed.

If a transformation function is defined by monotonically decreasing radial basis functions, the farther a control point is from a neighborhood, the smaller will be its influence on that neighborhood. Radial basis functions that are monotonically decreasing are, therefore, more suitable for registration of images with local geometric differences. Moreover, monotonically decreasing basis functions keep the inaccuracy in a correspondence to a small neighborhood of the inaccuracy and will not spread the inaccuracy over the entire image domain.

Examples of radial basis functions with monotonically decreasing basis functions are Gaussians [37, 87],

$$R_i(x, y) = \exp\left\{-\frac{(x - x_i)^2 + (y - y_i)^2}{2\sigma_i^2}\right\} \tag{9.49}$$

and inverse multiquadrics [25, 43],

$$R_i(x, y) = \left[(x - x_i)^2 + (y - y_i)^2 + d^2\right]^{-\frac{1}{2}}. \tag{9.50}$$

Franke [25] has found through extensive experimentation that monotonically decreasing radial basis functions do not perform as well as monotonically increasing radial basis functions when data are accurate and are randomly spaced. Therefore, if the coordinates of corresponding points in the images are known to be accurate, multiquadric is preferred over inverse multiquadric in image registration. However, if some point coordinates are not accurate or the local geometric difference between some areas in the images is sharp, monotonically decreasing radial functions are preferred over monotonically increasing radial functions in image registration.

9.2.1.2 Surface Spline

Surface spline, also known as *thin-plate spline* (TPS), is perhaps the most widely used transformation function in nonrigid image registration. Harder and Desmarais [41] introduced it as an engineering mathematical tool and Duchon [20] and Meinguet [69] investigated its properties. It was used as a transformation function in the registration of remote sensing images by Goshtasby [33] and in the registration of medical images by Bookstein [6].

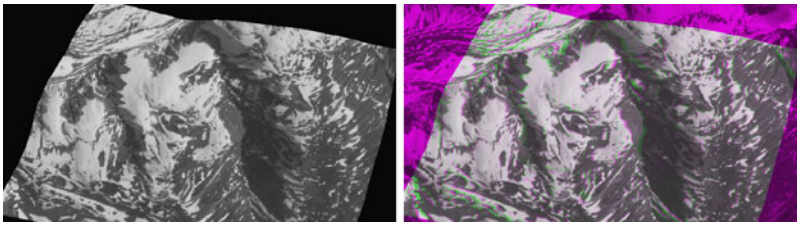
Given a set of points in the plane with associating values as described by (9.44), the surface spline interpolating the points is defined by

$$f(x, y) = A_1 + A_2 x + A_3 y + \sum_{i=1}^{n} B_i r_i^2 \ln r_i^2, \tag{9.51}$$

where $r_i^2 = (x - x_i)^2 + (y - y_i)^2 + d^2$. Surface spline is formulated in terms of an affine transformation and a weighted sum of radially symmetric (logarithmic) basis functions. In some literature, basis functions of form $r_i^2 \log r_i$ are used. Since $r_i^2 \log r_i^2 = 2r_i^2 \log r_i$, by renaming $2B_i$ by B_i we obtain the same equation. $r_i^2 \log r_i^2$ is preferred over $r_i^2 \log r_i$ as it avoids calculation of the square root of r_i^2.

Surface spline represents the equation of a plate of infinite extent deforming under point loads at $\{(x_i, y_i) : i = 1, \ldots, n\}$. The plate deflects under the imposition of the loads to take values $\{F_i : i = 1, \ldots, n\}$. Parameter d^2 acts like a stiffness parameter. As d^2 is increased, a smoother surface is obtained. When spacing between the points varies greatly in the image domain, a stiffer surface increases fluctuations in the interpolating surface. Franke [26] used a tension parameter as a means to keep fluctuations in interpolation under control.

Equation (9.51) contains $n + 3$ parameters. By substituting the coordinates of n points as described by (9.44) into (9.51), n equations are obtained. Three more



(a) (b)

Fig. 9.6 Registration of the Mountain image set using surface spline as the transformation function. (**a**) Resampled sensed image. (**b**) Overlaying of the reference and resampled sensed images

equations are obtained from the following constraints:

$$\sum_{i=1}^{n} B_i = 0, \tag{9.52}$$

$$\sum_{i=1}^{n} x_i B_i = 0, \tag{9.53}$$

$$\sum_{i=1}^{n} y_i B_i = 0. \tag{9.54}$$

Constraint (9.52) ensures that the sum of the loads applied to the plate is 0 so that the plate will not move up or down. Constraints (9.53) and (9.54) ensure that moments with respect to the x- and y-axes are zero, so the surface will not rotate under the imposition of the loads.

Using surface spline transformation to register the Mountain image set in Fig. 9.5, the results shown in Fig. 9.6 are obtained when letting the stiffness parameter $d = 0$. Comparing these results with those obtained by multiquadric interpolation, we see that while within the convex hull of the control points similar results are obtained, outside the convex of the control points surface spline produces significantly better results than multiquadric. By increasing the stiffness parameter d^2, registration error increases.

When the control point correspondences contain errors and the density of control points in the reference image is not uniform, improved registration accuracy can be achieved by allowing each component of the transformation to approximate rather than interpolate the points. Rohr et al. [85] added a smoothing term to the interpolating spline while letting $d^2 = 0$ to obtain a surface that contained smaller fluctuations. As the smoothness term is increased, the obtained surface becomes smoother and fluctuations become smaller, but the surface moves away from some of the control points. The process, therefore, requires interaction by the user to specify a smoothness parameter that is large enough to reduce noise among control-point correspondences but not so smooth as to increase distances between the surface and the points it is approximating.

Monotonically increasing radial basis functions such as multiquadrics and surface splines that interpolate points produce a smooth mapping from one image to

another. If the correspondences are accurate, surfaces representing the components of the transformation represent smoothly varying geometric differences between the images. However, when a function is defined in terms of monotonically increasing basis functions, a positional error in a pair of corresponding points in the images will influence the registration accuracy everywhere in the image domain.

Since radial basis functions are symmetric, when spacing between the control points varies greatly across the image domain, the transformation may produce large errors away from the control points. To increase registration accuracy, the density of the control points may be increased, but that will not only slow down the process, it will make the process unstable as it will require the solution of large systems of equations to find the parameters of the transformation.

Compactly supported radial basis functions, examined next, use local basis functions to keep errors and deformations local.

9.2.1.3 Compactly Supported Radial Basis Functions

Monotonically decreasing radial basis functions can be defined with local support in such a way that the data value at (x, y) is determined from data at a small number of points near (x, y). Interpolation by compactly supported radial basis functions is defined by

$$f(x, y) = \sum_{i=1}^{n} A_i R_i(x, y) = \sum_{i=1}^{n} A_i W(r_i), \tag{9.55}$$

where $r_i = \sqrt{(x - x_i)^2 + (y - y_i)^2}$. By replacing r_i with $\sqrt{(x - x_i)^2 + (y - y_i)^2}$, a function in (x, y) is obtained, which has been denoted by $R_i(x, y)$ in the above formula. $W(r_i)$ can take different forms. Wendland [102] defined it by

$$W(r_i) = \begin{cases} (a - r_i)^2, & 0 \le r_i \le a, \\ 0, & r_i > a, \end{cases} \tag{9.56}$$

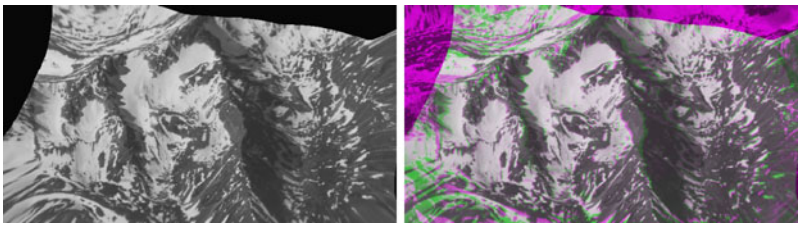
while Buhmann [9] defined it by

$$W(r_i) = \begin{cases} \frac{112}{45}(a - r_i)^{\frac{9}{2}} + \frac{16}{3}(a - r_i)^{\frac{7}{2}} - 7(a - r_i)^4 - \frac{14}{15}(a - r_i)^2 + \frac{1}{9}, \\ \quad 0 \le r_i \le a, \\ 0, \quad r_i > a. \end{cases} \tag{9.57}$$

In both cases, $W(r_i)$ not only vanishes at distance a from (x_i, y_i), but its gradient vanishes also. Therefore, the basis functions smoothly vanish at distance a from their centers and a weighted sum of them will create a surface that will be smooth everywhere in the image domain.

Parameter a should be large enough so that within each region of radius a, at least a few control points appear in the reference image. The unknown parameters $\{A_i : i = 1, \ldots, n\}$ are determined by solving the following system of linear equations:

$$F_j = \sum_{i=1}^{n} A_i R_i(x_j, y_j), \quad j = 1, \ldots, n. \tag{9.58}$$



(a) (b)

Fig. 9.7 Registration of the Mountain image set using Wendland's compactly supported radial basis functions with parameter $a = 5000$ pixels. (**a**) Resampled sensed image. (**b**) Overlaying of the reference and resampled sensed images

Note that although the basis functions have local support, a global system of equations has to be solved to find parameters $\{A_i : i = 1, \ldots, n\}$.

Using Wendland's compactly supported radial functions as the transformation to register the Mountain image set in Fig. 9.5, acceptable results are not obtained when a is small enough to consider the transformation local. As parameter a is increased, registration accuracy improves. Registration of the images when $a = 5000$ pixels is shown in Fig. 9.7. Results are acceptable within the convex hull of the control points, but they are inferior to those obtained by surface spline.

To overcome some of the weaknesses of compactly supported radial basis functions of a fixed support radius a, use of a hierarchy of compactly supported radial basis functions of varying support radii has been proposed [72]. Starting from basis functions of a large radius, basis functions of smaller radii are added to the approximation until residual errors in approximation fall within a desired range. A method proposed by Floater and Iske [23] uses a hierarchy of basis functions. The radii of basis functions at different levels are estimated by successive triangulation of the points and determination of the triangle sizes at each hierarchy. Wider basis functions are used to capture global structure in data while narrower basis functions are used to capture local details in data.

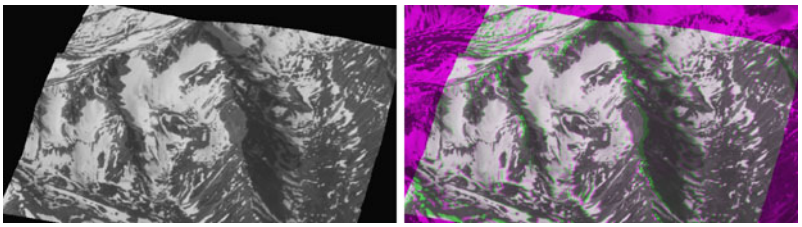
To avoid solving a system of equations, Maude [66] used weight functions with local support to formulate an approximation method to irregularly spaced data. Maude's weight functions are defined by:

$$W_i(x, y) = W(R_i) = \begin{cases} 1 - 3R_i^2 + 2R_i^3, & 0 \le R_i \le 1, \\ 0, & R_i > 1, \end{cases} \tag{9.59}$$

where $R_i = \sqrt{(x - x_i)^2 + (y - y_i)^2}/R_k$ and R_k is the distance of (x, y) to the kth point closest to it. Note that not only $W(R_i)$ vanishes at distance R_k to point (x, y), its first derivative vanishes there also. Then

$$f(x, y) = \frac{\sum_{i=1}^{k} F_i W_i(x, y)}{\sum_{i=1}^{k} W_i(x, y)} \tag{9.60}$$

is used as the approximating functional value at (x, y).



(a) (b)

Fig. 9.8 Registration results using Maude's local interpolation formula with neighborhood size $k = 10$ points. (**a**) Resampling of sensed image to overlay the reference image. (**b**) Overlaying of the reference and resampled sensed images

Therefore, to estimate functional value at (x, y), the k points closest to (x, y) are identified. Let's suppose data values at the points are: $\{F_i : i = 1, \ldots, k\}$. Then a weighted sum of the values is calculated and used as the value at (x, y). The weights vanish at the kth point and the sum of the weights everywhere in a region of radius R_k centered at (x, y) is 1.

Note that the neighborhood size automatically adjusts to the local density of points. In areas where a high density of points is available, parameter R_k will be small, while in sparse areas, R_k will be large. The method does not require the solution of a system of equations, but it does require determination of the k control points that are closest to pixel (x, y) in the reference image.

Maude's weighted mean approximation uses rational weights, which is known to produce flat spots in the obtained surface at the control points. We will see later in this chapter how such errors can be reduced through parametric reformulation of the problem. Another way to remedy the flat-spot effect is to use data values as well as data gradients at the points. This can be achieved by replacing F_i in (9.60) with a linear function that evaluates to F_i at (x_i, y_i) and fits the k points closest to (x, y) by the least-squares method. Denoting such a linear function by $L_i(x, y)$, (9.60) becomes

$$f(x, y) = \frac{\sum_{i=1}^{k} L_i(x, y) W_i(x, y)}{\sum_{i=1}^{k} W_i(x, y)}. \tag{9.61}$$

This represents a local weighted linear approximation. Registering the mountain images using (9.61) as the components of the transformation with $k = 10$, the result shown in Fig. 9.8 is obtained. Except for areas with sharp geometric differences, the images are registered relatively well.

A local weighted mean method that interpolates irregularly spaced data is described by McLain [67]. In this method, first, the given points are triangulated. Then, the patch over each triangle is computed from the weighted sum of data at the vertices of the triangle. If data at the three vertices of a triangle are F_1, F_2, and F_3, the functional value at (x, y) inside the triangle is obtained from

$$f(x, y) = W_1(x, y) F_1 + W_2(x, y) F_2 + W_3(x, y) F_3, \tag{9.62}$$

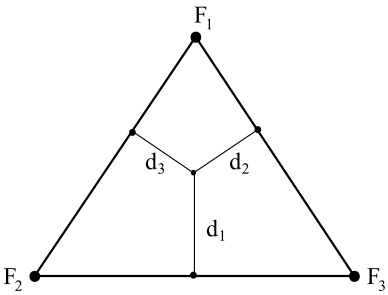


Fig. 9.9 McLain [67] interpolation over a triangle

where W_1, W_2, and W_3 are weights associated with data at the vertices of the triangle and are determined by first calculating the distance of point (x, y) to the three sides of the triangle (Fig. 9.9):

$$d_i(x, y) = l_i x + m_i y + n_i, \quad \text{for } i = 1, 2, 3. \tag{9.63}$$

Coefficients l_i, m_i, and n_i are determined only once for each triangle side and are normalized so that $d_i = 1$ when $(x, y) = (x_i, y_i)$. Then the weight associated with a vertex is set proportional to the distance of (x, y) to the triangle side opposing it. That is

$$W_i(x, y) = \frac{d_i^2}{d_1(x, y)^2 + d_2(x, y)^2 + d_3(x, y)^2}, \quad \text{for } i = 1, 2, 3. \tag{9.64}$$

Square weights are used to ensure continuous and smooth transition from one triangle to the next. If second derivative continuity is required across triangle edges, the cubic power of distances is needed to define the weights [67].

Radial basis functions with local support are preferred over radial basis functions with global support when registering images with local geometric differences. Remote sensing images of a 3-D scene captured from different views or serial images of a patient captured by a medical scanner have local geometric differences. Compactly supported radial basis functions, by modeling the geometric difference between corresponding local neighborhoods in images, use a small number of points within corresponding areas to transform the geometry of the sensed image locally to resemble that of the reference image. In this manner, global registration is achieved via local registration.

A comparison between globally defined radial basis functions and compactly supported radial basis functions in medical image registration has been provided by Fornefett et al. [24]. Improved registration accuracy has been reported with compactly supported radial basis functions over globally defined radial basis functions in the registration of serial brain images.

Although not a radial function, tensor-product functions that have local support, such as B-splines, can be used in approximation/interpolation also. Lee et al. [57] used multi-level B-splines with varying local support to interpolate data in the plane. The control points of a B-spline surface are determined by the least-squares method in such a way that the surface would interpolate given points. By using B-spline basis functions with different support levels, different levels of detail are reproduced in

the surface. By adding together B-spline basis functions at different support levels, a multi-level B-spline interpolation to scattered data is obtained.

For very large and irregularly spaced data, Bozzini et al. [7] laid a regular grid over the approximation domain and estimated the data at each grid point from the noisy and irregularly spaced data around it. Then, a B-spline surface was fitted to data at the regular grid. To produce B-splines that interpolate scattered data, Greiner et al. [39] first found parameter coordinates at the points to guarantee existence of an interpolating B-spline. Then, the control vertices of the interpolating B-spline surface were determined by an optimization process formulated in terms of surface fairness.

B-splines are a family of grid functions that are defined over regular grids of nodes (parameter coordinates). The process of generating grid functions that approximate scattered data is known as *gridding*. In spite of their limitations, in certain engineering applications, grid functions are preferred over other functions because of their ability to easily modify and visualize an approximation. Arge et al. [3] developed a three-step process for approximating scattered data by grid functions. The steps are: (1) Regularization: Identifying a subset of grid nodes in regions where density of data is high. (2) Approximation: Finding values at the grid nodes using approximation to nearby data. (3) Extrapolation: Extending the data values defined on the grid subset to the entire grid.

9.2.1.4 Moving Least-Squares

Suppose data points $\{\mathbf{p}_i = (x_i, y_i) : i = 1, \ldots, n\}$ with associating data values $\{F_i : i = 1, \ldots, n\}$ are given. A moving least-squares approximation is a function $f(\mathbf{p})$ that minimizes [52]:

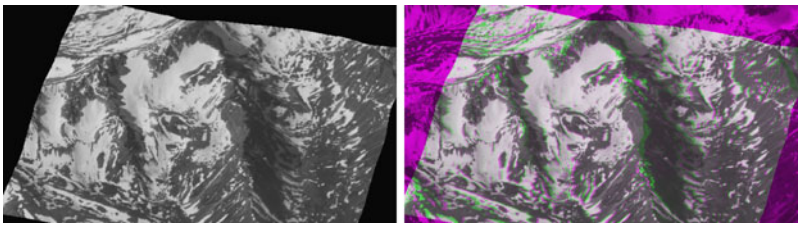
$$\sum_{i=1}^{n} \big[f(\mathbf{p}_i) - F_i \big]^2 W_i(\mathbf{p}) \tag{9.65}$$

at each $\mathbf{p} = (x, y)$. $W_i(\mathbf{p})$ is a non-negative monotonically decreasing radial function centered at $\mathbf{p}_i$. This weight function ensures that a data point closer to $\mathbf{p}$ will influence the estimated value more than a data point that is farther away. If function f is a polynomial in x and y, the best polynomial for point (x, y) is determined by the weighted least squares in such a way as to minimize (9.65).

Note that relation (9.65) is specific to point $\mathbf{p}$. Therefore, function f determined according to (9.65) will be specific to point $\mathbf{p}$ and vary from point to point. Since the parameters of a new function have to be determined for each point in the approximation domain, f cannot be a very complex function. Typically, it is a polynomial of degree 1 or 2.

For interpolating moving least squares, it is required that the weight functions assume value ∞ at $\mathbf{p} = \mathbf{p}_i$. Some of the suggested weight functions are [53]:

$$W_i(\mathbf{p}) = \frac{1}{\|\mathbf{p} - \mathbf{p}_i\|^2}, \tag{9.66}$$



(a) (b)

Fig. 9.10 Registration with moving least-squares using linear polynomials and weight functions of (9.66). (**a**) Resampling of the sensed image to overlay the reference image. (**b**) Overlaying of the reference and the resampled sensed images

$$W_i(\mathbf{p}) = \frac{1}{\|\mathbf{p} - \mathbf{p}_i\|^4}, \tag{9.67}$$

$$W_i(\mathbf{p}) = \frac{\alpha \exp(-\beta \|\mathbf{p} - \mathbf{p}_i\|^2)}{\|\mathbf{p} - \mathbf{p}_i\|^k}, \quad \alpha, \beta, k > 0. \tag{9.68}$$

To make the computations local, compactly supported weight functions are used. Examples are [53]:

$$W_i(\mathbf{p}) = \begin{cases} a\|\mathbf{p} - \mathbf{p}_i\|^{-k}(1 - \|\mathbf{p} - \mathbf{p}_i\|/d)^2, & \text{for } \|\mathbf{p} - \mathbf{p}_i\| \le d, \\ 0, & \text{for } \|\mathbf{p} - \mathbf{p}_i\| > d, \end{cases} \tag{9.69}$$

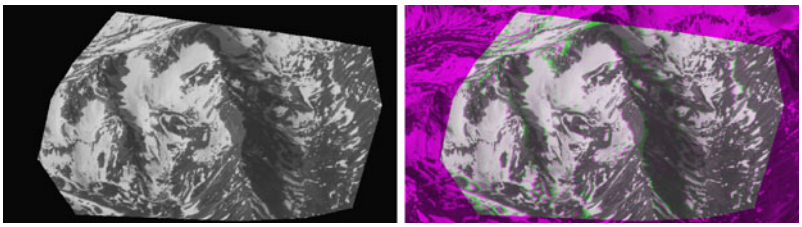
$$W_i(\mathbf{p}) = \begin{cases} a\|\mathbf{p} - \mathbf{p}_i\|^{-k} \cos(\pi \|\mathbf{p} - \mathbf{p}_i\|/2d), & \text{for } \|\mathbf{p} - \mathbf{p}_i\| \le d, \\ 0 & \text{for } \|\mathbf{p} - \mathbf{p}_i\| > d. \end{cases} \tag{9.70}$$

When f represents a polynomial of degree 1, the surface obtained by moving least-squares will be continuous and smooth everywhere in the approximation domain [51]. Levin [58] has found that moving least-squares are not only suitable for interpolation but are also useful in smoothing and derivatives estimation. For further insights into moving least-squares and its variations, see the excellent review by Belytschko et al. [4].

An example of image registration by moving least squares using the Mountain image set, linear polynomials, and weight functions of (9.66) is given in Fig. 9.10. The transformation is well-behaved outside the convex hull of the control points, and registration is acceptable at and near the control points; however, registration error is relatively large away from the control points.

9.2.1.5 Piecewise Polynomials

If control points in the reference image are triangulated [56, 91], by knowing the correspondence between the control points in the sensed and reference images, corresponding triangles will be known in the sensed image. This makes it possible to determine a transformation function for corresponding triangles and map triangles in the sensed image one by one to the corresponding triangles in the reference image.



(a) (b)

Fig. 9.11 Registration of the Mountain image set using the piecewise linear transformation. (**a**) Resampling of the sensed image to the space of the reference image. (**b**) Overlaying of the reference and the resampled sensed images

If a linear function is used to do the mapping, the transformation becomes piecewise linear.

If coordinates of the vertices of the ith triangle in the reference image are (x_{i1}, y_{i1}), (x_{i2}, y_{i2}), and (x_{i3}, y_{i3}) and coordinates of the corresponding vertices in the sensed image are (X_{i1}, Y_{i1}), (X_{i2}, Y_{i2}), and (X_{i3}, Y_{i3}), the ith triangular regions in the images can be related by an affine transformation as described by (9.19) and (9.20). The six parameters of the transformation, a–f, can be determined by substituting the coordinates of three corresponding triangle vertices into (9.19) and (9.20) and solving the obtained system of linear equations.

Finding an affine transformation for each corresponding triangle produces a composite of local affine transformations or an overall piecewise linear transformation. An example of image registration by piecewise linear interpolation is depicted in Fig. 9.11. Registration is shown within the convex hull of the control points in the reference image. Although affine transformations corresponding to the boundary triangles can be extended to cover image regions outside the convex hull of the control points, registration errors outside the convex hull of the points could be large and so is not recommended. Piecewise linear transformation has been used in image registration before [31]. The method was later extended to piecewise cubic [32] to provide a smooth as well as continuous mapping within the convex hull of the control points.

Within the convex hull of the control points, registration by piecewise linear is comparable to surface spline or moving least-squares. Although piecewise linear transformation is continuous within the convex hull of the points, it is not smooth across the triangle edges. The affine transformations obtained over triangles sharing an edge may have different gradients, producing an overall transformation that is continuous but not smooth.

To ensure that a transformation is smooth as well as continuous across a triangle edge, a polynomial of degree two or higher is required to represent the component of a transformation over each triangle. The parameters of the polynomial are determined in such a way that adjacent triangular patches join smoothly and produce the same gradient at the two sides of an edge, and all patches sharing a vertex produce the same gradient at the vertex. Various triangular patches that provide this property have been proposed [2, 12, 14–17, 48, 50, 54, 63, 73, 76, 89, 97].

A factor that affects the registration accuracy is the choice of triangulation. As a general rule, elongated triangles should be avoided. Algorithms that maximize the minimum angle in triangles is know as Delaunay triangulation [38, 54]. A better approximation accuracy is achieved if triangulation is obtained in 3-D using the data values as well as the data points. Various data-dependent triangulation algorithms have been proposed [5, 8, 21, 22, 83, 92].

If the points are triangulated in 3-D, a subdivision method may be used to create a smooth approximation or interpolation to the triangle mesh. A subdivision method typically subdivides each triangle into four smaller triangles with a limiting smooth surface that approximates or interpolates the mesh vertices [64, 65, 74, 88, 98].

Loop [60] proposed a recursive subdivision algorithm that approximates a smooth surface to a triangle mesh, while Dyn et al. [22] proposed a recursive algorithm that generates a smooth surface interpolating the vertices of a triangle mesh. Doo [18] and Doo and Sabin [19] described a subdivision scheme that can approximate a mesh with triangular, quadrilateral, and, in general, n-sided faces. Subdivision surfaces contain B-spline, Bézier, and non-uniform B-spline (NURBS) as special cases [90]. Therefore, transformation functions can be created with each component representing a piecewise surface composed of B-spline, Bézier, or NURBS patches.

In the following, two of the popular subdivision algorithms that work with triangle meshes are described. The subdivision scheme developed by Loop [44, 60] generates an approximating surface, while the subdivision scheme developed by Dyn et al. [22] creates an interpolating surface. The Loop subdivision scheme is depicted in Fig. 9.12. Given a triangle mesh, at each iteration of the algorithm a triangle is replaced with four smaller triangles by (1) inserting a new vertex near the midpoint of each edge, (2) refining the old vertex positions, and (3) replacing each old triangle with four new triangles obtained by connecting the new and refined triangle vertices.

Assuming triangle vertices at iteration r surrounding vertex $\mathbf{v}^r$ are $\mathbf{v}_1^r, \mathbf{v}_2^r, \ldots, \mathbf{v}_k^r$ (Fig. 9.12d), new vertex $\mathbf{v}_i^{r+1}$ is inserted midway between $\mathbf{v}^r$ and $\mathbf{v}_i^r$ for $i = 1, \ldots, k$. The location of a newly inserted vertex is computed from

$$\mathbf{v}_i^{r+1} = \frac{3\mathbf{v}^r + 3\mathbf{v}_i^r + \mathbf{v}_{i-1}^r + \mathbf{v}_{i+1}^r}{8}, \quad i = 1, \ldots, k. \tag{9.71}$$

Then, vertex $\mathbf{v}^r$ is replaced with

$$\mathbf{v}^{r+1} = (1 - k\beta)\mathbf{v}^r + \beta\left(\mathbf{v}_1^2 + \cdots + \mathbf{v}_k^r\right), \tag{9.72}$$

where according to Loop [60]

$$\beta = \frac{1}{k}\left(\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}\cos(2\pi/k)\right)^2\right). \tag{9.73}$$

A different set of subdivision rules are used along the boundary of the mesh to prevent the approximating open surface from shrinking towards its center after a number of iterations. Only points along the boundary are used in the rules as de-

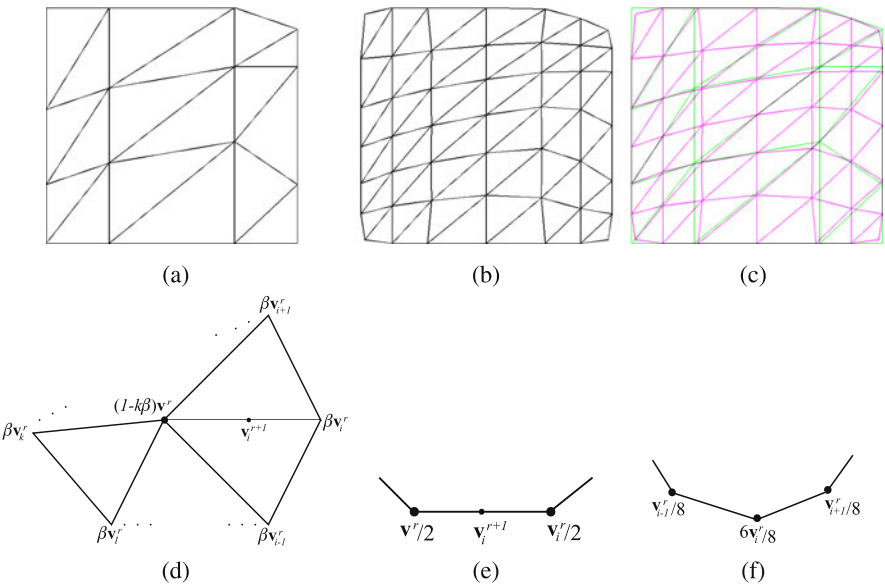


Fig. 9.12 (**a**) A triangle mesh. (**b**) The mesh after one iteration of Loop subdivision. (**c**) Overlaying of (**a**) and (**b**). (**d**) Loop vertex insertion and refinement rules for interior edges and vertices. (**e**) Loop vertex insertion for boundary edges and (**f**) vertex refinement for boundary vertices

picted in Figs. 9.12e, f. The vertex inserted between $\mathbf{v}^r$ and $\mathbf{v}_i^r$ along the boundary is computed from

$$\mathbf{v}_i^{r+1} = \frac{\mathbf{v}^r + \mathbf{v}_i^r}{2} \tag{9.74}$$

and vertex $\mathbf{v}_i^r$, which is between $\mathbf{v}_{i-1}^r$ and $\mathbf{v}_{i+1}^r$ along the boundary, is replaced with

$$\mathbf{v}_i^{r+1} = \frac{\mathbf{v}_{i-1}^r + 6\mathbf{v}_i^r + \mathbf{v}_{i+1}^r}{8}. \tag{9.75}$$

At the limit, the surface generated by Loop subdivision is C^1 continuous everywhere [95, 103]. That is, not only is the created surface continuous over the approximation domain, its first derivative is also continuous everywhere. For image registration purposes, the insertion and refinement steps should be repeated until the surface at iteration $r+1$ is sufficiently close to that obtained at iteration r. Sufficiently close is when the maximum refinement among all vertices in an iteration is less than half a pixel and all newly inserted vertices are less than half a pixel away from their edge midpoints. This ensures that subdivision surfaces at two consecutive iterations produce the same resampled image when using the nearest-neighbor resampling rule.

Registration of the Mountain data set using the Loop subdivision surface is shown in Fig. 9.13. Although Loop subdivision surface produces a smoother resampled image due to gradient continuity of the transformation function within the

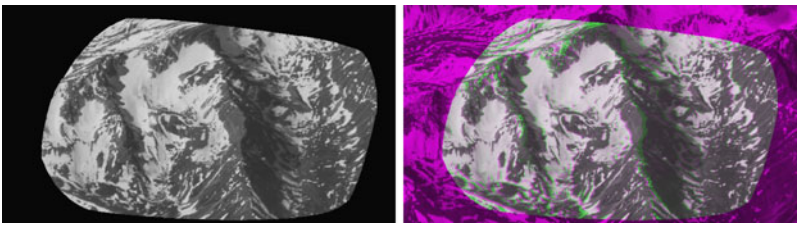


Fig. 9.13 Registration of the Mountain image set using Loop subdivision surfaces as the components of the transformation. (**a**) Resampling of the sensed image to the space of the reference image. (**b**) Overlaying of the reference and the resampled sensed images

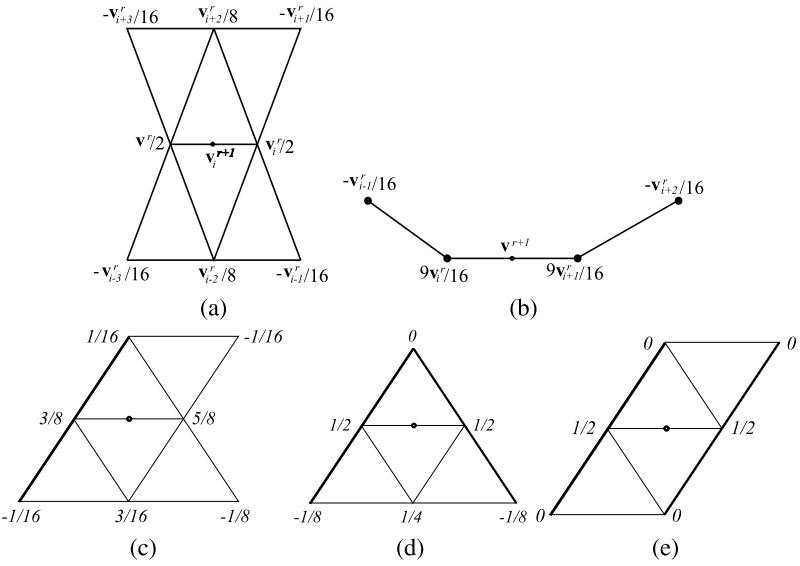


Fig. 9.14 Butterfly subdivision rules for (**a**) interior edges, (**b**) boundary edges, and (**c**)–(**e**) interior edges that touch the boundary or a crease

convex hull of the control points when compared with piecewise linear, there isn't a significant difference between the registration accuracy of the two methods.

The interpolative subdivision surface described by Dyn et al. [22] uses a neighborhood that has the shape of a butterfly as shown in Fig. 9.14a. Subdivision requires vertex insertion only. Existing vertices are not repositioned after each iteration because the original and newly inserted vertices are on the limiting surface. Vertex $\mathbf{v}_i^{r+1}$, which is newly inserted between vertices $\mathbf{v}^r$ and $\mathbf{v}_i^r$ when surrounded by the vertices shown in Fig. 9.14a, is computed from

$$\mathbf{v}_i^{r+1} = \frac{\mathbf{v}^r + \mathbf{v}_i^r}{2} + \frac{\mathbf{v}_{i-2}^r + \mathbf{v}_{i+2}^r}{8} - \frac{\mathbf{v}_{i-3}^r + \mathbf{v}_{i-1}^r + \mathbf{v}_{i+1}^r + \mathbf{v}_{i+3}^r}{16}. \tag{9.76}$$

Subdivision rules along the boundary are slightly different. Vertex $\mathbf{v}^{r+1}$, which is inserted between vertices $\mathbf{v}_i^r$ and $\mathbf{v}_{i+1}^r$ along the boundary, is computed from

$$\mathbf{v}^{r+1} = \frac{-\mathbf{v}_{i-1}^r + 9\mathbf{v}_i^r + 9\mathbf{v}_{i+1}^r - \mathbf{v}_{i+2}^r}{16}. \tag{9.77}$$

Vertex insertion at interior edges that touch the boundary or a crease is obtained using the rules shown in Figs. 9.14c–e.

The limiting surface produced by the butterfly subdivision scheme of Dyn et al. [22] is C^1-continuous everywhere when a regular mesh is provided. The surface, however, is not smooth at mesh vertices of valance $k = 3$ or $k > 7$ when an irregular mesh is given [103]. Zorin [103, 104] proposed a modified butterfly subdivision scheme that at the limit interpolates a smooth surface to any triangle mesh. Qu and Agarwal [77] described a 10-point interpolatory subdivision scheme over an arbitrary triangle mesh that has a limiting surface that is smooth everywhere, including at the mesh vertices.

9.2.2 Parametric

Parametric functions are of form

$$\mathbf{P}(u, v) = \mathbf{f}(u, v). \tag{9.78}$$

$\mathbf{P}(u, v)$ is the surface point at (u, v), defined as a function of parameters u and v. $\mathbf{f}(u, v)$ is a function with three independent components, each a function of (u, v); therefore,

$$x(u, v) = f_x(u, v), \tag{9.79}$$

$$y(u, v) = f_y(u, v), \tag{9.80}$$

$$F(u, v) = f_F(u, v). \tag{9.81}$$

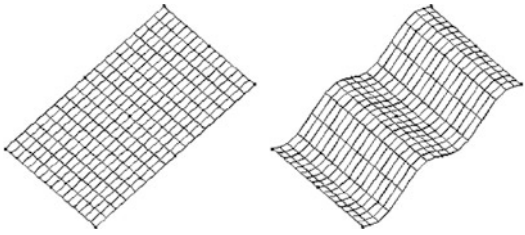
Since the three components of a parametric surface are independent of each other, each can be determined separately.

Given $\{(x_i, y_i, F_i) : i = 1, \ldots, n\}$, to determine the surface value at (x, y), first, the corresponding (u, v) coordinates are determined from (9.79) and (9.80). Knowing (u, v), surface value F is then calculated. The nonlinear nature of the equations makes determination of exact surface values very time consuming. For image registration purposes, however, we will see that approximations to the surface values can be determined efficiently with sufficient accuracy.

Parametric surfaces used in geometric modeling require a regular grid of control points. The control points available in image registration are, however, irregularly spaced. Below, parametric surfaces suitable for interpolation/approximation to scattered data are explored.

Table 9.1 Coordinates of 9 uniformly spaced points in the xy domain with associating data values

i	1	2	3	4	5	6	7	8	9
x_i	0	1	2	0	1	2	0	1	2
y_i	0	0	0	1	1	1	2	2	2
F_i	0	1	2	0	1	2	0	1	2



(a) (b)

Fig. 9.15 Interpolation of the data in Table 9.1 by Shepard's method. (**a**) The ideal surface and (**b**) the surface obtained by Shepard's method

9.2.2.1 Parametric Shepard Interpolation

One of the earliest methods for the interpolation of scattered data is proposed by Shepard [96]. This is a weighted mean method with rational weights. Given data sites $\{(x_i, y_i) : i = 1, \ldots, n\}$ with associating data values $\{F_i : i = 1, \ldots, n\}$, Shepard's interpolation is defined by

$$f(x, y) = \sum_{i=1}^{n} W_i(x, y) F_i, \tag{9.82}$$

where

$$W_i(x, y) = \frac{R_i(x, y)}{\sum_{j=1}^{n} R_j(x, y)}, \tag{9.83}$$

and

$$R_i(x, y) = \left\{(x - x_i)^2 + (y - y_i)^2\right\}^{-\frac{1}{2}}. \tag{9.84}$$

The surface interpolates the points, yet it does not require the solution of a system of equations. The interpolating surface is obtained immediately by substituting the coordinates of the data sites and the data values into (9.82).

Shepard's method is known to produce flat spots in the surface at and near the data sites. Consider the data in Table 9.1, showing coordinates of 3-D points in a plane as depicted in Fig. 9.15a. Shepard's method, however, produces the surface depicted in Fig. 9.15b.

The reason for the flat spots is the nonlinear relation between xy and f. The flat spots show increased surface point density near the data sites. This weakness can be overcome by subjecting x and y to the same nonlinear transformation that f is subjected to. By letting $(u_i, v_i) \propto (x_i, y_i)$ and defining the components of the

parametric Shepard similarly by formula (9.82), we obtain

$$x(u,v) = \sum_{i=1}^{n} W_i(u,v)x_i, \tag{9.85}$$

$$y(u,v) = \sum_{i=1}^{n} W_i(u,v)y_i, \tag{9.86}$$

$$f(u,v) = \sum_{i=1}^{n} W_i(u,v)F_i, \tag{9.87}$$

where

$$W_i(u,v) = \frac{R_i(u,v)}{\sum_{j=1}^{n} R_j(u,v)}, \tag{9.88}$$

$$R_i(u,v) = \left\{(u-u_i)^2 + (v-v_i)^2\right\}^{-\frac{1}{2}}, \tag{9.89}$$

$u_i = x_i/(n_c - 1)$, and $v_i = y_i/(n_r - 1)$. n_c and n_r are, respectively, the number of columns and number of rows in the reference image. As x varies between 0 and $n_c - 1$, u will vary between 0 and 1, and as y varies between 0 and $n_r - 1$, v will vary between 0 and 1.

Parametric Shepard, however, requires the solution of two nonlinear equations to find (u, v) for a given (x, y). Then, it uses the obtained (u, v) to find the surface value F. For image registration purposes though, this is not necessary since exact surface coordinates are not required. Surface coordinates that are within half a pixel of the actual coordinates are sufficient to resample the sensed image to align with the reference image when using nearest neighbor resampling.

The following algorithm determines a component of a transformation function by the parametric Shepard method.

Algorithm PSI (Parametric Shepard Interpolation) Given points $\{(x_i, y_i, F_i) : i = 1, \ldots, n\}$, calculate image $F[x, y]$, showing the surface interpolating the points when quantized at discrete pixel coordinates in the reference image.

1. Let $u_i = x_i/(n_c - 1)$ and $v_i = y_i/(n_r - 1)$. This will ensure parameters in the image domain vary between 0 and 1.
2. Initially, let increments in u and v be $\Delta u = 0.5$ and $\Delta v = 0.5$.
3. For $u = 0$ to 1 with increment Δu and for $v = 0$ to 1 with increment Δv, repeat the following.

   - If $[x(u,v) + x(u+\Delta u, v)]/2! \in [x(u + \Delta u/2, v) \pm 0.5]$ or $[y(u,v) + y(u + \Delta u, v)]/2! \in [y(u + \Delta u/2, v) \pm 0.5]$ or $[F(u,v) + F(u + \Delta u, v)]/2! \in [F(u + \Delta u/2, v) \pm 0.5]$ or
     $[x(u,v) + x(u, v + \Delta v)]/2! \in [x(u, v + \Delta v/2) \pm 0.5]$ or $[y(u,v) + y(u, v + \Delta v)]/2! \in [y(u, v + \Delta v/2) \pm 0.5]$ or $[F(u,v) + F(u, v + \Delta v)] \neq [F(u, v + \Delta v/2) \pm 0.5]$ or
     $[x(u,v) + x(u + \Delta u, v) + x(u, v + \Delta v) + x(u + \Delta u, v + \Delta v)]/4! \in [x(u + \Delta u/2, v + \Delta v/2) \pm 0.5]$ or $[y(u,v) + y(u + \delta u, v) + y(u, v + \Delta v) + y(u + \Delta u, v + \Delta v)]/4! \in [y(u + \Delta u/2, v + \Delta v/2) \pm 0.5]$ or

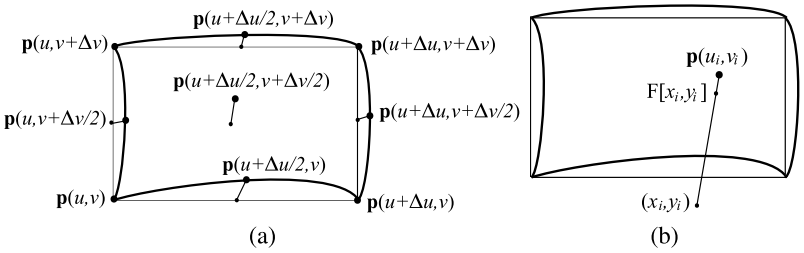


Fig. 9.16 (**a**) The subdivision scheme at Step 3. (**b**) Ensuring the approximating surface passes within half a pixel of the given points in Step 5

$[F(u,v)+F(u+\delta u,v)+x(u,v+\Delta v)+F(u+\Delta u,v+\Delta v)]/4! \in [F(u+\Delta u/2, v+\Delta v/2) \pm 0.5]$, then reduce Δu and Δv by a factor of 2 and go to Step 3.

4. If $F_i! \in [F[x_i, y_i] \pm 0.5]$ for any $i = 1, \ldots, n$, reduce Δu and Δv by a factor of 2 and repeat this step.
5. For $u = 0$ to 1 with increment Δu and for $v = 0$ to 1 with increment Δv, repeat the following.
   - Calculate $[x(u,v), y(u,v), F(u,v)]$, $[x(u+\Delta u, v), y(u+\Delta u, v), F(u+\Delta u, v)]$, $[x(u+\Delta u, v+\Delta v), y(u+\Delta u, v+\Delta v), F(u+\Delta u, v+\Delta v)]$, $[x(u, v+\Delta v), y(u, v+\Delta v), F(u, v+\Delta v)]$. This defines a local patch. Estimate values within the patch using bilinear interpolation of values at its four corners.

By notation "$a! \in [b \pm 0.5]$," it is implied "if $a < b - 0.5$ or $a > b + 0.5$." In Step 3, for each patch defined within parameters (u, v) to $(u + \Delta u, v + \Delta v)$, the distances of the midpoints of the four sides and at the center of the patch to its bilinear approximation (Fig. 9.16a) are determined. Subdivision is continued until all distances become smaller than half a pixel.

Step 4 ensures that the obtained approximation is within half a pixel of the points it is supposed to interpolate. If it is not, subdivision is continued until the approximating surface falls within half a pixel of the given points. Note that in Step 3 the patches are not generated. Values at only edge midpoints and patch centers are calculated. In most situations, this finds the required increment in u and v that will obtain the required surface. In some rare cases, the process may not produce a surface sufficiently close to the given points. In such cases, Step 4 ensures that the obtained surface does, in fact, pass within half a pixel of the points it is supposed to interpolate (Fig. 9.16b).

The interpolating parametric Shepard defined in this manner may produce sharp edges and corners at the interpolating points. This problem can be alleviated by replacing the radial function defined in (9.89) by

$$R_i(u,v) = \left\{(u-u_i)^2 + (v-v_i)^2 + d^2\right\}^{-\frac{1}{2}}. \tag{9.90}$$

d^2 is a small positive number. The larger its value, the smoother the obtained surface will be, but also the farther the surface will fall from some of the points. Note that

this is an inverse multiquadric weight. Therefore, Shepard weights can be considered rational inverse multiquadric weights. When $d^2 = 0$, the surface will interpolate the points and when $d^2 > 0$, the surface will approximate the points. W_i is a rational function in u and v when parametric Shepard is used with $u_i = x_i/(n_c - 1)$ and $v_i = y_i/(n_r - 1)$ for $i = 1, \ldots, n$.

Letting

$$R_i(u, v) = \exp\left\{-\frac{(u - u_i)^2 + (v - v_i)^2}{2(s\sigma_i)^2}\right\}, \tag{9.91}$$

the obtained surface will be a rational Gaussian (RaG) surface [37] that approximates the points. The standard deviation of the Gaussian at the ith point, σ_i, shows spacing between the points surrounding it. It can be taken equal to the distance of that point to the kth point closest to it. The smoothness parameter s is a global parameter that will increase or decrease the standard deviations of all Gaussians simultaneously. The larger is the value for s, the smoother will be the obtained surface. The smaller is the s, the more closely the approximation will follow local data. Since the influence of a Gaussian vanishes exponentially, for small standard deviations and considering the digital nature of images, the weight functions, in effect, have only local support.

By setting the standard deviations of Gaussians proportional to the spacing between the points, the surface is made to automatically adapt to the spacing between the points. In areas where density of points is high, narrow Gaussians are used to keep the effect of the points local. In areas where the points are sparse, wide Gaussians are used to cover large gaps between the points.

As the standard deviations of Gaussians are increased, the surface gets smoother and moves away from some of the points. To ensure that a surface interpolates the points, new data values $\{A_i : i = 1, \ldots, n\}$ at $\{(u_i, v_i) : i = 1, \ldots, n\}$ are determined such that the surface obtained from the new data values will evaluate to the old data values at the parameter coordinates corresponding to the data sites. That is, the surface is obtained by solving

$$x_j = \sum_{i=1}^{n} A_i W_i(u_j, v_j), \tag{9.92}$$

$$y_j = \sum_{i=1}^{n} B_i W_i(u_j, v_j), \tag{9.93}$$

$$F_j = \sum_{i=1}^{n} C_i W_i(u_j, v_j), \tag{9.94}$$

for $\{A_i, B_i, C_i : i = 1, \ldots, n\}$, where $j = 1, \ldots, n$, and

$$W_i(u_j, v_j) = \frac{G_i(u_j, v_j)}{\sum_{k=1}^{n} G_k(u_j, v_j)} \tag{9.95}$$

is the ith basis function of the RaG surface evaluated at (u_j, v_j), and $G_i(u_j, v_j)$ is a 2-D Gaussian of standard deviation $s\sigma_i$ centered at (u_i, v_i) when evaluated at (u_j, v_j).

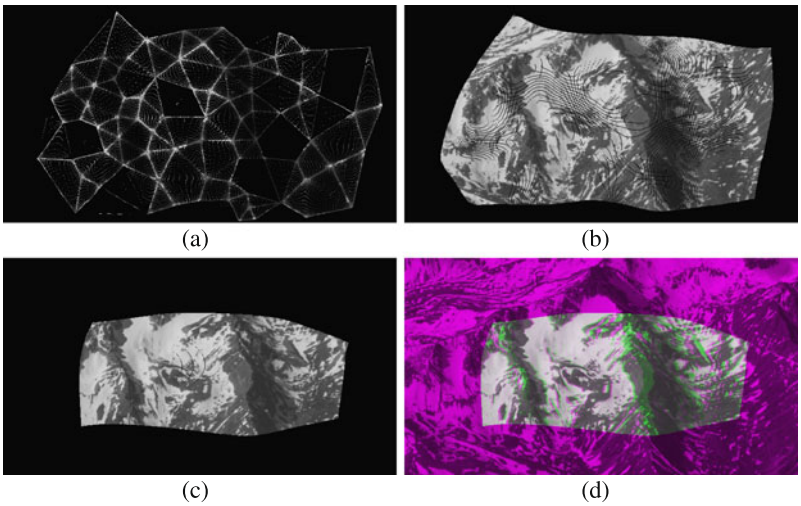


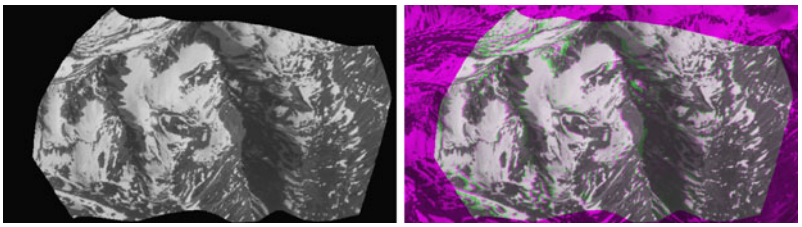
Fig. 9.17 (**a**)–(**c**) Resampling of the sensed image to the space of the reference image as the smoothness parameters is increased. Density of surface points is high at and near the control points as well as along edges connecting the points when smoothness parameter s is very small. Missing surface values are estimated by bilinear interpolation as outlined in Algorithm PSA. (**d**) Registration with parametric Shepard approximation when $s = 2.5$

It is important to note that due to the nonlinear relation between (x, y) and (u, v), by varying u and v from 0 to 1, x may not vary between 0 and $n_c - 1$ and y may not vary between 0 and $n_r - 1$. Consequently, it may be necessary to start u and v slightly below 0 and continue slightly past 1. If u and v are varied between 0 and 1, the sensed image may leave some gaps near the borders of the reference image.

Examples of parametric Shepard approximation using RaG weights are given in Fig. 9.17. The standard deviation of a Gaussian at a control point is set proportional to the distance of that control point to the control point closest to it in the reference image. Therefore, $k = 1$. Figure 9.17a shows resampling of the sensed image when $s = 0.25$. That is, the standard deviation at a control point is set to 0.25 times the distance of that control point to the control point closest to it. At such low standard deviations, the approximation is close to piecewise linear, and for uniformly spaced u and v, surface points primarily concentrate along edges and at vertices of the triangle mesh obtained from the points.

By increasing the smoothness parameter to 1, a smoother surface is obtained and for uniformly spaced u and v, points on the surface become more uniformly spaced as shown in (b). Increasing the smoothness parameter to 2.5 will further increase the smoothness of the surface, but it will shrink the surface at the same time when varying u and v from 0 to 1, as depicted in (c). It also moves the surface farther from some of the points, increasing approximation error. The registration result when $s = 2.5$ is depicted in (d).

In order to create a smooth surface that interpolates the points, we will find new coordinates $\{(A_i, B_i, C_i) : i = 1, \ldots, n\}$ such that the obtained surface would inter-



(a) (b)

Fig. 9.18 Registration of the Mountain image set using parametric Shepard interpolation as the components of the transformation. (**a**) Resampling of the sensed image to the space of the reference image. (**b**) Overlaying of the reference and the resampled sensed images

polate 3-D points $\{(x_i, y_i, F_i) : i = 1, \ldots, n\}$. Doing so, we obtain the resampled image shown in Fig. 9.18a and the registration result shown in Fig. 9.18b. Ignoring its rough boundary, the quality of registration obtained by interpolative parametric Shepard is as good as any of the methods discussed so far.

Examining Shepard's interpolation as described by (9.82), we see that the surface that interpolates a set of points is obtained by a weighted sum of horizontal planes passing through the points. The plane passing through point (x_i, y_i, F_i) is $F(x, y) = F_i$. The reason for obtaining a high density of points near (x_i, y_i) is that many points near (x_i, y_i) produce values close to F_i. This formulation ignores the surface gradient at (x_i, y_i) and always uses horizontal plane $F(x, y) = F_i$ at (x_i, y_i). One remedy to this problem is to use a plane with a gradient equal to that estimated at (x_i, y_i) rather than using gradient 0 at every point.

Gradient vectors at the data points, if not given, can be estimated directly from the data. Typically, a surface is fitted to the points and the gradient vectors of the surface at the points are determined. Stead [99] found that gradient vectors produced by multiquadric surface fitting is superior to those estimated by other methods when using randomly spaced points. Goodman et al. [30] triangulated the points with their associating data values in 3-D and used a convex combination of gradient vectors of the triangle planes sharing a point as the gradient vector at the point.

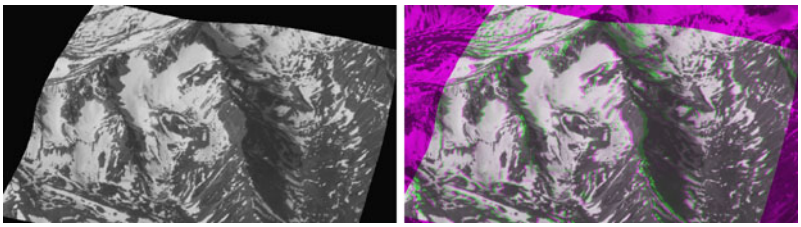
To find the gradient vector at a point, we fit a plane to the that point and $k > 2$ other points nearest to it by the least-squares method. The gradients of the plane are then taken as estimates to the gradients of the surface at the point. Assuming the plane fitting to point (x_i, y_i, F_i) and a small number of points around it by the least-squares method is

$$F(x, y) = a_i x + b_i y + c_i, \tag{9.96}$$

we recalculate c_i in such a way that $F(x_i, y_i) = F_i$. Doing so, we find $c_i = F_i - a_i x_i - b_i y_i$. Therefore, the equation of the plane passing through the ith point will be

$$L_i(x, y) = a_i(x - x_i) + b_i(y - y_i) + F_i. \tag{9.97}$$

In the Shepard interpolation of (9.82), we replace F_i, which is a horizontal plane passing through point (x_i, y_i, F_i), with $L_i(x, y)$, which is a plane of a desired gra-



(a) (b)

Fig. 9.19 Registration of the Mountain image set using weighted linear approximation as the components of the transformation. (**a**) Resampling of the sensed image to the space of the reference image. (**b**) Overlaying of the reference and the resampled sensed images. The smoothness parameter $s = 1$ in this example

dient passing through the same point. The weighted sum of such planes produces a weighted linear interpolation to the points:

$$f(x, y) = \frac{\sum_{i=1}^{n} R_i(x, y) L_i(x, y)}{\sum_{i=1}^{n} R_i(x, y)}. \tag{9.98}$$

This weighted linear function [34, 36] interpolates the points and provides desired gradients at the points. To make the surface approximate the points, instead of (9.89) we let the radial functions be (9.91) but define it in the xy space. If necessary, this surface can be made to interpolate the points by finding new data values at the points in such a way that the obtained surface would evaluate to the old data values at the control points by solving a system of equations similar to (9.94) but as a function of (x, y) rather than (u, v). Note that this new formulation is in explicit form; therefore, revising Shepard's method to use gradients at the points will make it possible to avoid formation of horizontal flat spots in the created surface without parametrizing it.

An example of the use of weighted linear approximation as the components of the transformation function in image registration is given in Fig. 9.19. RaG weights are used with the standard deviation of Gaussian at a point proportional to the distance of that point to the point closest to it. The smoothness parameter s is set to 1 in Fig. 9.19. Since this is an approximating surface, increasing s will create a smoother surface that gets farther from some of the given points. As s is decreased, the surface will more resemble a piecewise linear interpolation. Being an approximation method, weighted linear is particularly suitable in image registration when a large number of point correspondences is given. Registration results are better than those obtained by multiquadric and surface spline and are comparable to those obtained by parametric Shepard interpolation.

A number of modifications to the Shepard interpolation have been proposed. These modifications replace a data point with a function. Franke and Nielson [27] fitted a quadratic function, Renka and Brown [80] fitted a cubic function, Lazzaro and Montefusco [55] fitted a radial function, and Renka and Brown [81] fitted a 10-parameter cosine series to a small number of points in the neighborhood of a point as the nodal function at the point. The weighted sum of the functions were then used

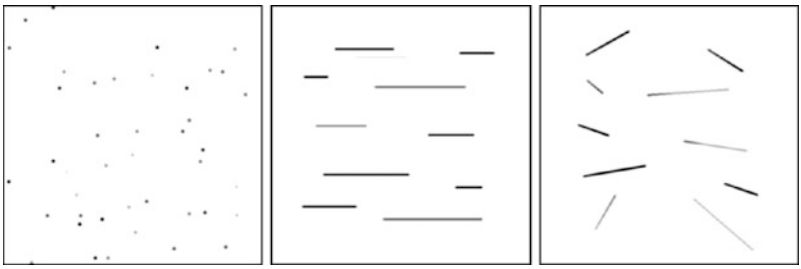


Fig. 9.20 (**a**) Scattered data points in the plane, showing 3-D points. (**b**) Scattered horizontal data lines, showing 3-D lines parallel to the x-axis. (**c**) Scattered data lines of arbitrary orientation with values along a line varying linearly. These represent scattered 3-D lines. Higher values are shown brighter in these images

to obtain the interpolation. Rational weights with local support are used, vanishing at a fixed distance of the data sites. Renka [78] further allowed the width of each weight function to vary with the density of local data and vanish at a distance equal to the distance of a data site to the kth data site closest to it.

Weights with local support are attractive because they are computationally efficient and do not allow a local deformation or inaccuracy to spread over the entire approximation domain. Weights with local support, however, may produce a surface with holes if spacing between the points varies greatly across the image domain.

9.2.2.2 Surface Approximation to Scattered Lines

Image registration methods rely on the coordinates of corresponding points in images to find the transformation function. Transformation functions defined in terms of points, however, cannot represent sharp geometric differences along edges, as found in images of man-made scenes taken from different views.

Line segments are abundant in images of indoor and outdoor scenes and methods to find correspondence between them have been developed [13, 46, 101]. Therefore, rather than defining a transformation function in terms of corresponding points, we would like to formulate the transformation function in terms of corresponding lines in images.

Suppose n corresponding line segments are obtained in two images. Let's denote the coordinates of the end points of the ith line segment by $(x_{i_1}, y_{i_1}, F_{i_1})$ and $(x_{i_2}, y_{i_2}, F_{i_2})$. We want to find a function $F = f(x, y)$ that approximates the lines.

The surface approximating a set of scattered lines is obtained by extending the equation of a surface that approximates a set of points [35]. Consider fitting a single-valued surface to data at scattered points in the plane $\{(x_i, y_i, F_i) : i = 1, \ldots, n\}$. An example of scattered data in the plane is given in Fig. 9.20a. Intensities of the points

represent the data values at the points. A weighted mean approximation to the data will be

$$f(x, y) = \sum_{i=1}^{n} F_i g_i(x, y). \tag{9.99}$$

$g_i(x, y)$ can be considered a rational basis function centered at (x_i, y_i) defined in such a way that the sum of n basis functions everywhere in the approximation domain is 1. One such example is rational Gaussian (RaG) basis functions [37]:

$$g_i(x, y) = \frac{w_i G_i(x, y)}{\sum_{j=1}^{n} w_j G_j(x, y)}, \tag{9.100}$$

where $G_i(x, y)$ is a 2-D Gaussian centered at (x_i, y_i) and w_i is the weight associated with the ith data point. For point data, we let $w_i = 1$ for $i = 1, \ldots, n$. For a line, we let a weight be proportional to the length of the line it represents. The standard deviations of the Gaussians can be varied to generate surfaces at different levels of detail.

Now, consider using a data line in place of a data point. For the sake of simplicity, let's first assume that data along a line does not vary and all lines are parallel to the x-axis. An example of such data lines is given in Fig. 9.20b. Therefore, instead of point (x_i, y_i), we will have a line with end points (x_{i_1}, y_{i_1}) and (x_{i_2}, y_{i_2}) and the same data value F_i everywhere along the line. To fit a surface to these lines, we will horizontally stretch the Gaussian associated with a line proportional to its length.

If the coordinates of the midpoint of the ith line are (x_i, y_i), since a 2-D Gaussian can be decomposed into two 1-D Gaussians, we have

$$G_i(x, y) = \exp\left\{-\frac{(x - x_i)^2 + (y - y_i)^2}{2\sigma^2}\right\}, \tag{9.101}$$

$$= \exp\left\{-\frac{(x - x_i)^2}{2\sigma^2}\right\} \exp\left\{-\frac{(y - y_i)^2}{2\sigma^2}\right\}, \tag{9.102}$$

$$= G_i(x) G_i(y). \tag{9.103}$$

To stretch $G_i(x, y)$ along the x-axis, we scale σ by a factor proportional to the length of the line. Let's denote this scaling by $m_i > 1$. Then, we replace $G_i(x)$ with

$$H_i(x) = \exp\left\{-\frac{(x - x_i)^2}{2(m_i\sigma)^2}\right\}, \tag{9.104}$$

where $m_i = (1 + \varepsilon_i)$ and ε_i is proportional to the length of the ith line. After this stretching, relation (9.99) becomes

$$f(x, y) = \frac{\sum_{i=1}^{n} w_i F_i H_i(x) G_i(y)}{\sum_{i=1}^{n} w_i H_i(x) G_i(y)}. \tag{9.105}$$

Now suppose data values along a line vary linearly, but the projections of the lines to the xy plane are still parallel to the x-axis. To fit a surface to such lines, instead of using a Gaussian of a fixed height F_i, we let the height of a Gaussian vary

with data along the line. Assuming data at the endpoints of the ith line are F_{i_1} and F_{i_2} and the data value at the line midpoint is F_i, in (9.105) we will replace F_i with

$$F_i(x) = F_i + \frac{(x - x_i)}{(x_{i_2} - x_i)}(F_{i_2} - F_i). \tag{9.106}$$

This formula changes the height of the Gaussian along a line proportional to the data values on the line. The new approximation formula, therefore, becomes

$$f(x, y) = \frac{\sum_{i=1}^{n} w_i F_i(x) H_i(x) G_i(y)}{\sum_{i=1}^{n} w_i H_i(x) G_i(y)}. \tag{9.107}$$

To adapt the surface to data lines with arbitrary orientations, such as those shown in Fig. 9.20c, we rotate each data line about its center so that it becomes parallel to the x-axis. Then, we use the above formula to find its contribution to the surface. Finally, we rotate the values back. Doing this for each line and adding contributions from the lines, we obtain the approximating surface. If the projection of the ith line to the xy-plane makes angle θ_i with the x-axis, when rotating the coordinate system clockwise about the line's midpoint by θ_i so that it becomes parallel to the x-axis, denoting the coordinates of points on the line before and after this rotation by (X, Y) and (x, y), we have

$$x = (X - X_i)\cos\theta_i - (Y - Y_i)\sin\theta_i + x_i, \tag{9.108}$$

$$y = (X - X_i)\sin\theta_i + (Y - Y_i)\cos\theta_i + y_i. \tag{9.109}$$

Substituting relations (9.108) and (9.109) into the right side of (9.107), we obtain a relation in (X, Y). This relation finds the surface value at (X, Y) in the approximation domain. Renaming the approximating function by $F(X, Y)$, we will have

$$F(X, Y) = \frac{\sum_{i=1}^{n} w_i F_i(X, Y) H_i(X, Y) G_i(X, Y)}{\sum_{i=1}^{n} w_i H_i(X, Y) G_i(X, Y)}, \tag{9.110}$$

where

$$F_i(X, Y) = F_i + \frac{(X - X_i)\cos\theta_i - (Y - Y_i)\sin\theta_i}{D_i}(F_{i_2} - F_i), \tag{9.111}$$

$$H_i(X, Y) = \exp\left\{-\frac{[(X - X_i)\cos\theta_i - (Y - Y_i)\sin\theta_i]^2}{2(m_i\sigma)^2}\right\}, \tag{9.112}$$

$$G_i(X, Y) = \exp\left\{-\frac{[(X - X_i)\sin\theta_i + (Y - Y_i)\cos\theta_i]^2}{2\sigma^2}\right\}, \tag{9.113}$$

and

$$D_i = \sqrt{(x_{i_2} - x_i)^2 + (y_{i_2} - y_i)^2} = \sqrt{(X_{i_2} - X_i)^2 + (Y_{i_2} - Y_i)^2} \tag{9.114}$$

is half the length of the ith line segment in the xy or XY domain. Weight w_i of line L_i is set equal to $1 + 2D_i$. The 1 in the formula ensures that if points are used in addition to lines, the obtained surface will approximate the points as well as the lines. As the length of a line increases, the volume under the stretched Gaussian

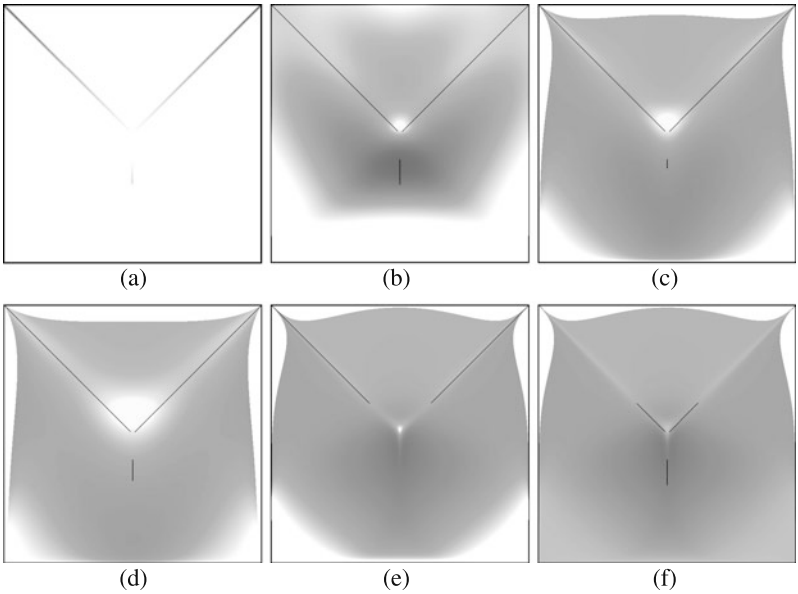


Fig. 9.21 (**a**) Data lines of Table 9.2. Higher values are shown brighter. (**b**) The single-valued surface of (9.110) approximating the data lines. (**c**) The parametric surface of (9.116)–(9.118) approximating the same data lines. (**d**) Same as (**c**) but using a larger σ. (**e**) Same as (**c**) but using a smaller σ. (**f**) Same as (**e**) but viewing from the opposite side. The lines and the approximating surface are overlaid for qualitative evaluation of the approximation

increases. To make the weight function dependent on the length of the line as well as on the data values along the line, we let

$$w_i = 1 + 2D_i = 1 + 2\sqrt{(X_{i_2} - X_i)^2 + (Y_{i_2} - X_i)^2 + (F_{i_2} - F_i)^2}. \qquad (9.115)$$

Substituting (9.111)–(9.113) into (9.110), a single-valued surface is obtained that approximates scattered line data in the plane.

An example of the kind of surfaces obtained by this method is shown in Fig. 9.21. Figure (a) shows seven data lines in the xy-plane. Intensities of points along a line show the data values. The coordinates of the line endpoints and the associating data values are shown in Table 9.2. Figure 9.21b shows the surface approximating the lines according to formula (9.110). Although the surface approximates the lines, flat spots are obtained along the lines. This is a known property of the weighted-mean method.

Since the sum of the weights is required to be 1 everywhere in the approximation domain, when the weight functions are rather narrow, flat spots are obtained at and near the data lines. To prevent such flat spots from appearing in the approximating surface, instead of a single-valued surface, as explained in the preceding section, a parametric surface should be used. Therefore, instead of the single-valued surface

Table 9.2 The coordinates of the endpoints of the lines in Fig. 9.21a and the associating data values

i	1	2	3	4	5	6	7
X_{i_1}	−50	50	50	−50	1	−1	0
Y_{i_1}	−50	−50	50	50	1	1	−10
F_{i_1}	0	0	0	0	50	50	40
X_{i_2}	50	50	−50	−50	50	−50	0
Y_{i_2}	−50	50	50	−50	50	50	−20
F_{i_2}	0	0	0	0	0	0	20

given by (9.110), we use the parametric surface defined by

$$F_x(u,v) = \frac{\sum_{i=1}^{n} w_i X_i(u,v) H_i(u,v) G_i(u,v)}{\sum_{i=1}^{n} w_i H_i(u,v) G_i(u,v)}, \tag{9.116}$$

$$F_y(u,v) = \frac{\sum_{i=1}^{n} w_i Y_i(u,v) H_i(u,v) G_i(u,v)}{\sum_{i=1}^{n} w_i H_i(u,v) G_i(u,v)}, \tag{9.117}$$

$$F_F(u,v) = \frac{\sum_{i=1}^{n} w_i F_i(u,v) H_i(u,v) G_i(u,v)}{\sum_{i=1}^{n} w_i H_i(u,v) G_i(u,v)}. \tag{9.118}$$

Doing so, we obtain the surface shown in Fig. 9.21c. F_x, F_y, and F_F are the x, y, and F components of the surface, each obtained by varying u and v from 0 to 1. Due to the nonlinear relation between (u, v) and (x, y), when varying u and v from 0 to 1, the obtained surface leaves gaps near the image borders. To recover surface values at and near the image borders, u and v need to be varied from values slightly below 0 to values slightly above 1.

In this example, parameter coordinates at the line midpoints and line end points were set proportional to the XY coordinates of the line midpoints and end points, respectively. That is,

$$u_i = (X_i - X_{\min})/(X_{\max} - X_{\min}), \tag{9.119}$$

$$u_{i_1} = (X_{i_1} - X_{\min})/(X_{\max} - X_{\min}), \tag{9.120}$$

$$u_{i_2} = (X_{i_2} - X_{\min})/(X_{\max} - X_{\min}), \tag{9.121}$$

$$v_i = (Y_i - Y_{\min})/(Y_{\max} - Y_{\min}), \tag{9.122}$$

$$v_{i_1} = (Y_{i_1} - Y_{\min})/(Y_{\max} - Y_{\min}), \tag{9.123}$$

$$v_{i_2} = (Y_{i_2} - Y_{\min})/(Y_{\max} - Y_{\min}), \tag{9.124}$$

where $X_{\min}$, $X_{\max}$, $Y_{\min}$, and $Y_{\max}$ define the range of coordinates in the approximation domain. In image registration, $X_{\min} = Y_{\min} = 0$, $X_{\max} = n_c - 1$, $Y_{\max} = n_r - 1$, and n_c and n_r are the image dimensions (i.e., number of columns and number of rows in the reference image).

The transformation with components described by (9.116)–(9.118) maps the sensed image to the reference image in such a way that corresponding lines in the images align. The transformation most naturally registers images containing sharp edges, such as close-range imagery of buildings and man-made structures. The accuracy of the method depends on the accuracy with which the endpoints of the lines are determined.

9.2.3 Implicit

Implicit functions are generally of the form

$$f(\mathbf{p}) = c. \tag{9.125}$$

Given point $\mathbf{p} = (x, y, F)$, the value at the point is $f(\mathbf{p})$. If this value happens to be c, the point will be on the surface. The process of determining an implicit surface involves producing a volumetric image and thresholding it at c. When $c = 0$, the obtained surface is called the *zero* surface or the *zero-crossing* surface.

Implicit surfaces are easy to generate, but if the function is not formulated carefully, multiple surface points can be obtained for the same (x, y), making resampling ambiguous. Implicit functions suitable for image registration are described next.

9.2.3.1 Interpolating Implicit Surfaces

If $\phi(\mathbf{p})$ is a radial function, a function of form

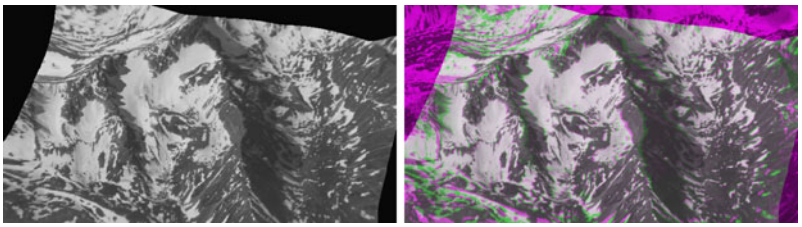
$$f(\mathbf{p}) = \sum_{i=1}^{n} A_i \phi\big(\|\mathbf{p} - \mathbf{p}_i\|\big) + L(\mathbf{p}) \tag{9.126}$$

will interpolate points $\{\mathbf{p}_i = (x_i, y_i, F_i) : i = 1, \ldots, n\}$ if it satisfies $f(\mathbf{p}_i) = h_i$ for $i = 1, \ldots, n$ [86, 100]. Since h_i can take any value, we will let it to be 0 for $i = 1, \ldots, n$. This will make the surface of interest be the zero surface of $f(\mathbf{p})$. $L(\mathbf{p})$ is an optional degree one polynomial in x, y, and F, with its coefficients determined in such a way that the surface would satisfy prespecified conditions. Carr et al. [11] used radial functions of form $\|\mathbf{p} - \mathbf{p}_i\|$, while Turk and O'Brien [100] used radial functions of form $\|\mathbf{p} - \mathbf{p}_i\|^3$. If logarithmic basis functions are used, $\phi(\|\mathbf{p} - \mathbf{p}_i\|) = \|\mathbf{p} - \mathbf{p}_i\|^2 \log(\|\mathbf{p} - \mathbf{p}_i\|^2)$.

Parameters $\{A_i : i = 1, \ldots, n\}$ are determined by letting $f(\mathbf{p}_i) = 0$ in (9.126) for $i = 1, \ldots, n$ and solving the obtained system of n linear equations. Note that the obtained system of equations will have a trivial solution $A_i = 0$ for $i = 1, \ldots, n$ when term $L(\mathbf{p})$ is not present. To avoid the trivial solution, additional constraints need to be provided. Since the surface traces the zeros of $f(\mathbf{p})$, one side of the surface will be positive, while the opposite side will be negative. To impose this constraint on the obtained surface, 2 virtual points $\mathbf{p}_{n+1}$ and $\mathbf{p}_{n+2}$ are added to the set of given points. $\mathbf{p}_{n+1}$ is considered a point below the surface and $\mathbf{p}_{n+2}$ is considered a point above the surface. Then, $f(\mathbf{p}_{n+1})$ is set to an appropriately large negative value and $f(\mathbf{p}_{n+2})$ is set to an appropriately large positive value.

Once the coefficients of the implicit surface are determined, the function is quantized within a volume where its xy domain covers the reference image and its F domain covers the columns (when $F = X$) or rows (when $F = Y$) of the sensed image. Then, the zero surface within the volume is obtained by thresholding the volume at 0 and tracing the zero values [61, 71].

An alternative approach to tracing the zero surface without creating an actual volume is to first find a point on the surface by scanning along F axis with discrete



(a) (b)

Fig. 9.22 Registration of the Mountain image set using interpolative implicit surfaces as the components of the transformation. (**a**) Resampling of the sensed image to align with the reference image. (**b**) Overlaying of the reference and resampled sensed images

steps within its possible range at an arbitrary point (x, y) in the image domain. Once the surface value F at (x, y) is determined, the surface value at a pixel adjacent to (x, y) is determined by using F as the start value and incrementing or decrementing it until a zero-crossing is detected. The process is propagated from one pixel to the next until surface points for all pixels in the reference image are determined.

An example of image registration using the interpolative implicit surface with $\phi(\|\mathbf{p} - \mathbf{p}_i\|) = \|\mathbf{p} - \mathbf{p}_i\|$ is given in Fig. 9.22. The two virtual points are assumed to be $(n_c/2, n_r/2, -n)$ and $(n_c/2, n_r/2, n)$, where n_r and n_c are the number of rows and columns, respectively, in the reference image and n is set to the number of columns of the sensed image when calculating the x-component and it is set to the number of rows of the sensed image when calculating the y-component of the transformation. A much larger n will require a longer time to calculate the surface points and a much smaller n will result in inaccurate surface values when incrementing and decrementing F by 1 to locate the zero-crossing at a particular (x, y). These virtual points are located in the middle of the image domain, one below and one above the surface. The registration result is shown in Fig. 9.22. Although the results may be acceptable within the convex hull of the control points, errors are rather large outside the convex hull of the points.

9.2.3.2 Approximating Implicit Surfaces

We are after an implicit function of form $f(x, y, F) = 0$ that can approximate points $\{\mathbf{p}_i = (x_i, y_i, F_i) : i = 1, \ldots, n\}$. If a 3-D monotonically decreasing radial function, such as a Gaussian, is centered at each point, then by adding the functions we obtain

$$f_1(x, y, F) = \sum_{i=1}^{N} g_i(\sigma, x, y, F), \tag{9.127}$$

where $g_i(\sigma, x, y, F)$ is a 3-D Gaussian of standard deviation σ centered at (x_i, y_i, F_i). f_1 in (9.127) generally increases towards the points and decreases away from the points. Therefore, by tracing locally maximum values of f_1, we can obtain a surface that passes near the points. When the points are uniformly spaced and the

standard deviations are all equal to the spacing between the points, the process will work well, but when the points are irregularly spaced, the process will produce a fragmented surface.

Usually, control points in an image are not uniformly spaced. To find a surface that approximates a set of irregularly spaced points, we center a 3-D Gaussian at each point with its standard deviation proportional to the distance of that point to the kth point closest to it. Adding such Gaussians, we obtain

$$f_2(x, y, F) = \sum_{i=1}^{n} g_i(\sigma_i, x, y, F), \tag{9.128}$$

where $g_i(\sigma_i, x, y, F)$ is a 3-D Gaussian of magnitude 1 and standard deviation σ_i centered at point (x_i, y_i, F_i). By tracing the local maxima of f_2 in the direction of maximum gradient, a surface that approximates the points will be obtained.

When isotropic Gaussians are centered at the points and the points are irregularly spaced, local maxima of f_2 in the gradient direction will again produce a fragmented surface. We have to stretch the Gaussians toward the gaps in order to avoid fragmentation. This is achieved by replacing a 3-D isotropic Gaussian with a 3-D anisotropic Gaussian oriented in such a way that it stretches toward the gaps.

Letting XYZ represent the local coordinate system of a point, with the Z-axis pointing in the direction of surface normal and XY defining the tangent plane at the point, the relation between the global coordinate system xyF of the surface and the local coordinate system XYZ of a point will be a rigid transformation. The 3-D anisotropic Gaussian centered at $\mathbf{p}_i$ in the local coordinate system of the point can be defined by

$$G_i(\sigma_X, X)G_i(\sigma_Y, Y)G_i(\sigma_Z, Z), \tag{9.129}$$

where $G_i(\sigma_X, X)$, $G_i(\sigma_Y, Y)$, and $G_i(\sigma_Z, Z)$ are 1-D Gaussians centered at the origin and laid along X-, Y-, and Z-axes, respectively.

To determine the coordinate axes at point $\mathbf{p}_i$, first, the surface normal at the point is determined by identifying the k closest points of $\mathbf{p}_i$ and calculating from them the covariance matrix [1]:

$$\mathbf{M}_i = \frac{1}{k}\sum_{j=1}^{k}\left(\mathbf{p}_i^j - \mathbf{p}_i\right)\left(\mathbf{p}_i^j - \mathbf{p}_i\right)^t, \tag{9.130}$$

where $\mathbf{p}_i^j$ denotes the jth point closest to $\mathbf{p}_i$ and t denotes matrix transpose operation. The eigenvectors of the 3×3 matrix $\mathbf{M}_i$ define three orthogonal axes, which are taken as the local coordinate axes at $\mathbf{p}_i$. The eigenvector associated with the smallest eigenvalue is taken as the surface normal at $\mathbf{p}_i$. All normals are made to point upward. The surface normal is taken as the Z-axis and the eigenvector associated with the largest eigenvalue is taken as the Y-axis of the local coordinate system. The X-axis is taken normal to both Y and Z.

Letting the eigenvalues of $\mathbf{M}_i$ from the largest to the smallest be λ_1, λ_2, and λ_3, we define

$$\sigma_X^2 = a\lambda_2, \tag{9.131}$$

$$\sigma_Y^2 = a\lambda_1, \tag{9.132}$$

$$\sigma_Z^2 = b\lambda_3. \tag{9.133}$$

This will ensure that the 3-D Gaussian is stretched toward the gaps where the density of points is low. The process will automatically adapts local averaging to the local density and organization of points. Parameters a and b are global parameters that can be varied to produce surfaces at different levels of detail. Parameters a and b smooth the surface in the tangent and normal directions. A larger a will stretch a Gaussian at a data point in the tangent direction of the approximating surface, filling large gaps between points and avoiding the creation of holes. A larger b smoothes the surface more in the normal direction, reducing noise among the correspondences and also smoothing surface details.

A local coordinate system is considered at point $\mathbf{p}_i$ with coordinate axes representing the eigenvectors of the covariance matrix $\mathbf{M}_i$. The sum of the Gaussians at point (x, y, F) in the approximation can be computed from:

$$f_3(x, y, F) = \sum_{i=1}^{n} g_i(\sigma_X, x) g_i(\sigma_Y, y) g_i(\sigma_Z, F), \tag{9.134}$$

where $g_i(\sigma_X, x)$, $g_i(\sigma_Y, y)$, and $g_i(\sigma_Z, F)$, correspondingly, represent 1-D Gaussians $G_i(\sigma_X, X)$, $G_i(\sigma_Y, Y)$, and $G_i(\sigma_Z, Z)$ after coordinate transformation from XYZ to xyF. Note that parameters σ_X, σ_Y, and σ_Z are local to point $\mathbf{p}_i$ and, thus, vary from point to point.

The surface to be recovered is composed of points where function $f_3(x, y, F)$ becomes locally maximum in the direction of surface normal. To simplify the surface detection process, rather than finding local maxima of $f_3(x, y, F)$ in the direction of surface normal, we determine the zero-crossings of the first derivative of $f_3(x, y, F)$ in the direction of surface normal. To achieve this, we orient the first-derivative of Gaussian in the direction of surface normal at each point in such a way that its positive side always points upward. Then, zero-crossings of the sum of the first-derivative Gaussians are determined and used as the approximating surface. More specifically, we use the zeros of

$$f(x, y, F) = \sum_{i=1}^{n} g_i(\sigma_X, x) g_i(\sigma_Y, y) g_i'(\sigma_Z, F) \tag{9.135}$$

as the approximating surface, where $g_i'(\sigma_Z, F)$ is the coordinate transformation of $G_i'(\sigma_Z, Z)$ from XYZ to xyF, and $G_i'(\sigma_Z, Z)$ is the first derivative of 1-D Gaussian $G_i(\sigma_Z, Z)$ centered at the origin and along the Z-axis.

Note that a zero-point of function $f(x, y, F)$ can be a locally maximum or a locally minimum point of $f_3(x, y, F)$ in the normal direction. However, only locally maximum points of function $f_3(x, y, F)$ correspond to the true surface points, and locally minimum points of $f_3(x, y, F)$ represent false surface points that have to be discarded.

Zero surface points that correspond to local minima of $f_3(x, y, F)$ in the normal direction can be easily identified by examining the sign of the second derivative of $f_3(x, y, F)$ calculated in the direction of surface normal. At the point where $f_3(x, y, F)$ is maximum in the normal direction, the second derivative of $f_3(x, y, F)$ in the normal direction will be negative, and at the point where $f_3(x, y, F)$ is minimum in the normal direction, the second derivative of $f_3(x, y, F)$ in the normal direction will be positive. Therefore, at each zero-crossing of $f(x, y, F)$, we find the sign of the second derivative of $f_3(x, y, F)$ calculated in the normal direction. If the sign is negative, the zero-crossing is retained, otherwise it is discarded.

Note that the second derivative of $f_3(x, y, F)$ in the normal direction is obtained by replacing $g_i'(\sigma_Z, F)$ in (9.135) with $g_i''(\sigma_Z, F)$, the second derivative of $g_i(\sigma_Z, F)$ in the normal direction, which is the second derivative of $G_i(\sigma_Z, Z)$ after the coordinate transformation from XYZ to xyz.

To summarize, steps in the implicit surface detection algorithm are:

1. For each point $\mathbf{p}_i$, $i = 1, \ldots, n$, repeat (a)–(c) below.
   a. Find the k closest points of $\mathbf{p}_i$.
   b. Using the points determine the eigenvalues ($\lambda_1 > \lambda_2 > \lambda_3$) and the corresponding eigenvectors ($\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$) of the covariance matrix $\mathbf{M}_i$ defined by (9.130) and use the eigenvectors to define a local coordinate system XYZ at $\mathbf{p}_i$.
   c. Let $\sigma_X^2 = a\lambda_2$, $\sigma_Y^2 = a\lambda_1$, and $\sigma_Z^2 = b\lambda_3$. a and b are globally controlled smoothness parameters.
2. Create an xyF volume of sufficient size and initialize the entries to 0.
3. For each point $\mathbf{p}_i, i = 1, \ldots, n$, add the volume representing $g_i(\sigma_X, x)g_i(\sigma_Y, y) \times g_i'(\sigma_Z, F)$ to the xyF volume.
4. Find the zero-crossings of the obtained volume.
5. Discard zero-crossings where the second derivative of $f_3(x, y, F)$ is positive, as they represent false surface points. The remaining zero-crossings define the desired surface.

The computation of the second derivative of $f_3(x, y, F)$ can be avoided by simply checking the magnitude of $f_3(x, y, F)$. If at a zero-crossing of the first derivative of $f(x, y, F)$, the magnitude of $f_3(x, y, F)$ is sufficiently large (say $> \varepsilon$) the zero-crossing is considered authentic. Otherwise, it is considered false and discarded. ε is usually a very small number, determined experimentally.

The process of centering the first-derivative of a 3-D anisotropic Gaussian at point $\mathbf{p}_i$ and adding the Gaussians to volume xyF is achieved by resampling the first-derivative of a 3-D isotropic Gaussian centered at the origin by a similarity transformation. The first-derivative (with respect to Z) of an isotropic Gaussian of standard deviation σ and magnitude 1 centered at the origin is:

$$G(\sigma, X, Y, Z) = G(\sigma, X)G(\sigma, Y)G'(\sigma, Z), \tag{9.136}$$

where

$$G(\sigma, X) = \exp\left\{-\frac{X^2}{2\sigma^2}\right\}, \qquad G(\sigma, Y) = \exp\left\{-\frac{Y^2}{2\sigma^2}\right\}, \tag{9.137}$$

and

$$G'(\sigma, Z) = -\frac{Z}{\sigma^2} \exp\left\{-\frac{Z^2}{2\sigma^2}\right\}. \tag{9.138}$$

The first-derivative isotropic Gaussian centered at the origin in the XYZ coordinate system is then transformed to the first-derivative anisotropic Gaussian at (x, y, F). This involves (1) scaling the isotropic Gaussian of standard deviation σ along X, Y, and Z by σ_X/σ, σ_Y/σ, and σ_Z/σ, respectively, (2) rotating it about X-, Y-, and Z-axes in such a way that the X-, Y-, and Z-axes align with the eigenvectors $\mathbf{v}_2$, $\mathbf{v}_1$, and $\mathbf{v}_3$ of covariance matrix $\mathbf{M}_i$, and (3) translating the scaled and rotated Gaussian to (x_i, y_i, F_i). Let's denote this similarity transformation by $\mathbf{A}_i$. Then, for each point $\mathbf{P} = (X, Y, Z)$ in the local coordinate system of point $\mathbf{p}_i$, the coordinates of the same point $\mathbf{p} = (x, y, F)$ in the xyF coordinate system will be $\mathbf{p} = \mathbf{A}_i\mathbf{P}$. Conversely, given point $\mathbf{p}$ in the xyF coordinate system, the same point in the local coordinate system of point $\mathbf{p}_i$ will be

$$\mathbf{P} = \mathbf{A}_i^{-1}\mathbf{p}. \tag{9.139}$$

Therefore, if the given points are in xyF space, create the first-derivative (with respect to Z) of an isotropic 3-D Gaussian centered at the origin in a sufficiently large 3-D array XYZ with the origin at the center of the array. Then, resample array XYZ and add to array xyF by the similarity transformation given in (9.139). This involves scanning the xyF volume within a small neighborhood of $\mathbf{p}_i$ and for each entry (x, y, F), determining the corresponding entry (X, Y, Z) in isotropic volume XYZ using (9.139), reading the value in the isotropic volume, and adding it to the value at entry (x, y, F) in the xyF volume.

Since a Gaussian approaches 0 exponentially, it is sufficient to scan the xyF space within a sphere of radius r_i centered at $\mathbf{p}_i$ to find its effect. r_i is determined to satisfy

$$\exp\left\{-\frac{r_i^2}{2\sigma_i^2}\right\} < \varepsilon \tag{9.140}$$

where σ_i is the largest of σ_X, σ_Y, and σ_Z calculated at $\mathbf{p}_i$, ε is the required error tolerance, which should be smaller than half the voxel size in the xyF volume to meet digital accuracy.

For a given a and b, the subvolume centered at each point (x_i, y_i, F_i) is determined. The isotropic first-derivative Gaussian is mapped to the subvolume with transformation $\mathbf{A}_i$, the sum of the anisotropic first-derivative Gaussians is determined, and its zero-surface is calculated by thresholding the volume at 0. The obtained zero surface will approximate points $\{(x_i, y_i, F_i) : i = 1, \ldots, n\}$.

9.3 Properties of Transformation Functions

Transformation functions carry information about scene geometry as well as the relation of cameras with respect to each other and with respect to the scene. Camera

geometry is global, while scene geometry is local. We would like to see if we can use information in a transformation function to estimate camera relations as well as scene geometry.

Because scene geometry is local in nature, it is reflected in the gradient of a transformation function. Camera geometry is either fixed across an image or it varies gradually; therefore, it has very little influence on the gradient of a transformation function.

If the components of a transformation function are

$$X = f_x(x, y), \tag{9.141}$$

$$Y = f_y(x, y), \tag{9.142}$$

the gradients of f_x with respect to x and y are

$$\frac{\partial X}{\partial x} = \frac{\partial f_x(x, y)}{\partial x}, \tag{9.143}$$

$$\frac{\partial X}{\partial y} = \frac{\partial f_x(x, y)}{\partial y}. \tag{9.144}$$

Therefore, the gradient magnitude of X at (x, y) can be computed from

$$\left|X'(x, y)\right| = \left\{\left(\frac{\partial X}{\partial x}\right)^2 + \left(\frac{\partial X}{\partial y}\right)^2\right\}^{\frac{1}{2}}. \tag{9.145}$$

Similarly, the gradient magnitude of the Y-component of the transformation is

$$\left|Y'(x, y)\right| = \left\{\left(\frac{\partial Y}{\partial x}\right)^2 + \left(\frac{\partial Y}{\partial y}\right)^2\right\}^{\frac{1}{2}}. \tag{9.146}$$

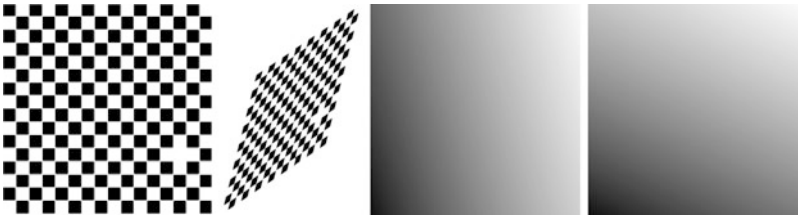
When the images are translated with respect to each other in a neighborhood, the components of the transformation that register the images in that neighborhood are defined by (9.5) and (9.6), from which we find $|X'(x, y)| = 1$ and $|Y'(x, y)| = 1$. Therefore, the gradient magnitude of each component of the transformation in the neighborhood under consideration is equal to 1 independent of (x, y).

When the images in a neighborhood have translational and rotational differences (rigid transformation) as defined by (9.9) and (9.10), the gradient magnitude for each component of the transformation in that neighborhood will be $\sqrt{\sin^2\theta + \cos^2\theta} = 1$. Therefore, the gradient magnitude of each component of the transformation in the neighborhood under consideration is also equal to 1 independent of (x, y).

When two images in a neighborhood are related by an affine transformation as defined by (9.19) and (9.20), the gradient magnitude of each component of the transformation in that neighborhood will be

$$\left|X'(x, y)\right| = \sqrt{a_1^2 + a_2^2}, \tag{9.147}$$

$$\left|Y'(x, y)\right| = \sqrt{a_3^2 + a_4^2}. \tag{9.148}$$



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

Fig. 9.23 (**a**), (**b**) An image and its transformation by an affine transformation. (**c**), (**d**) The X-component and the Y-component of the transformation, respectively. Values in the components of the transformation are appropriately scaled for viewing purposes

This shows that the X-component and the Y-component of an affine transformation have different gradient magnitudes unless $\sqrt{a_1^2 + a_2^2} = \sqrt{a_3^2 + a_4^2}$, implying the images are related by the similarity transformation. Therefore, gradient magnitudes of the two components of the similarity transformation are also the same. However, the gradient magnitude may be smaller than or larger than 1. The gradient magnitude, in fact, is equal to the $\sqrt{2}$ of the scale of the sensed image with respect to that of the reference image.

When two images are locally related by an affine transformation, gradient magnitudes $\sqrt{a_1^2 + a_2^2}$ and $\sqrt{a_3^2 + a_4^2}$, in addition to containing scale information, contain information about shearing of the sensed image with respect to the reference image. A larger shearing is obtained when the scene makes a larger angle with the direction of view. Therefore, the gradient of an affine transformation can be used to guess the orientation of the planar scene with respect to the view direction. The gradients of the X-component and the Y-component contain information about foreshortening of the scene horizontally and vertically with respect to the view.

Transforming the image in Fig. 9.23a by an affine transformation with $a_1 = 1.5$, $a_2 = 0.5, a_3 = 0, a_4 = 1, a_5 = 2$, and $a_6 = 0$, we obtain the image shown in Fig. 9.23b. The X-component and the Y-component of this transformation are shown in Figs. 9.23c and 9.23d, respectively. The gradient magnitude for the X-component transformation computed digitally is 1.581, which is the same as its theoretical value $\sqrt{a_1^2 + a_2^2} = \sqrt{2.5}$. The gradient magnitude of the Y-component transformation determined digitally is 2.236, which is the same as its theoretical value $\sqrt{a_3^2 + a_4^2} = \sqrt{5}$.

When two images are locally related by the projective transformation as defined by (9.23) and (9.24), the gradients of the two components become

$$\frac{\partial X}{\partial x} = \frac{a_1(a_7x + a_8y + 1) - a_7(a_1x + a_2y + a_3)}{(a_7x + a_8y + 1)^2}, \tag{9.149}$$

$$\frac{\partial X}{\partial y} = \frac{a_2(a_7x + a_8y + 1) - a_8(a_1x + a_2y + a_3)}{(a_7x + a_8y + 1)^2}, \tag{9.150}$$

$$\frac{\partial Y}{\partial x} = \frac{a_4(a_7x + a_8y + 1) - a_7(a_4x + a_5y + a_3)}{(a_7x + a_8y + 1)^2}, \tag{9.151}$$

$$\frac{\partial Y}{\partial y} = \frac{a_5(a_7x + a_8y + 1) - a_8(a_4x + a_5y + a_3)}{(a_7x + a_8y + 1)^2}, \tag{9.152}$$

or

$$\frac{\partial X}{\partial x} = \frac{a_1 - a_7X}{a_7x + a_8y + 1}, \tag{9.153}$$

$$\frac{\partial X}{\partial y} = \frac{a_2 - a_8X}{a_7x + a_8y + 1}, \tag{9.154}$$

$$\frac{\partial Y}{\partial x} = \frac{a_4 - a_7Y}{a_7x + a_8y + 1}, \tag{9.155}$$

$$\frac{\partial Y}{\partial y} = \frac{a_5 - a_8Y}{a_7x + a_8y + 1}, \tag{9.156}$$

or

$$\frac{\partial X}{\partial x} = A_1 + A_2X, \tag{9.157}$$

$$\frac{\partial X}{\partial y} = A_3 + A_4X, \tag{9.158}$$

$$\frac{\partial Y}{\partial x} = A_5 + A_2Y, \tag{9.159}$$

$$\frac{\partial Y}{\partial y} = A_6 + A_4Y, \tag{9.160}$$

therefore,

$$\left|X'(x, y)\right| = \sqrt{(A_1 + A_2X)^2 + (A_3 + A_4X)^2}, \tag{9.161}$$

$$\left|Y'(x, y)\right| = \sqrt{(A_5 + A_2Y)^2 + (A_6 + A_4Y)^2}. \tag{9.162}$$

The gradient magnitude for the X-component of the projective transformation is not only dependent on (x, y), it depends on X. Similarly, the gradient magnitude of the Y-component of the transformation is a function of Y as well as (x, y). Also, the gradient magnitudes of the two components of the projective transformation depend on each other. The gradient magnitudes become independent of (x, y) when $a_7 = a_8 = 0$, and that happens when the projective transformation becomes an affine transformation.

Since $\partial X/\partial x$ and $\partial X/\partial y$ are linear functions of X, their derivatives with respect to X will be constants. Denoting $\partial X/\partial x$ by X_x and denoting $\partial X/\partial y$ by X_y, we find $dX_x/dX = A_2$ and $dX_y/dX = A_4$. Let's define

$$\left|(dX)'\right| \equiv \sqrt{(dX_x/dX)^2 + (dX_y/dX)^2} = \sqrt{A_2^2 + A_4^2}. \tag{9.163}$$

Similarly, denoting $\partial Y/\partial x$ by Y_x and denoting $\partial Y/\partial y$ by Y_y, we find

$$\left|(dY)'\right| \equiv \sqrt{(dY_x/dY)^2 + (dY_y/dY)^2} = \sqrt{A_2^2 + A_4^2}, \tag{9.164}$$

Fig. 9.24 (**a**) A projective transformation of the image in Fig. 9.23a. (**b**), (**c**) The X-component and the Y-component of the transformation. (**d**), (**e**) Images representing $|(dX)'|$ and $|(dY)'|$. In (**b**)–(**e**) the values are appropriately scaled to range $[0, 255]$ for enhanced viewing

we find that $|(dX)'| = |(dY)'|$. This implies that the gradient of the X-component of a projective transformation calculated in the xy domain has a gradient magnitude with respect to X that is the same as the gradient of the Y-component of the transformation calculated in the xy domain when its gradient magnitude is calculated with respect to Y. However, this amount varies from pixel to pixel as A_2 and A_4 both depend on (x, y).

An example showing this property is given in Fig. 9.24. Using the image in Fig. 9.23a and letting the parameters of the projective transformation be $a_1 = 1.5$, $a_2 = -0.5$, $a_3 = 0$, $a_4 = 1$, $a_5 = 2$, $a_6 = 0$, $a_7 = 0.005$, and $a_8 = 0.01$, we obtain the transformed image shown in Fig. 9.24a. The X- and Y-components of this transformation are shown in 9.24b and 9.24c. The gradient magnitude of the gradient of the two components of the transformation, $|(dX)'|$ and $|(dY)'|$, as shown in 9.24d and 9.24e, are exactly the same. This property can be used to determine whether a transformation in a neighborhood is projective or not.

When the geometric difference between two images varies locally, the above mentioned properties hold within corresponding local neighborhoods in the images. At each (x, y), $|X'|$ and $|Y'|$ can be determined and based on their values, the geometric difference between the images at and in the neighborhood of (x, y) can be guessed. The parameters of the transformation mapping images in the neighborhood of (x, y) can be estimated using the X and the Y values at (x, y) and at pixels around it. Knowing the X- and the Y-components of a transformation, algorithms can be developed to examine X, Y, $|X'|$, and $|Y'|$ at each pixel and derive information about the geometry of the scene.

Consider the example in Fig. 9.25. Images (a) and (b) show the X-component and the Y-component of the transformation obtained by the weighted-linear (WLIN) method to register the Mountain image set. Images (c) and (d) represent $|X'|$ and $|Y'|$. We see a larger variation in gradient magnitudes of the X-component transformation than the Y-component transformation. This is typical of stereo images, showing a larger change in foreshortening horizontally than vertically. Variation in local geometry of the sensed image with respect to the reference image is reflected in the components of the transformation. Images $|X'|$ and $|Y'|$ not only contain information about the geometry of the scene, they contain information about the relation of the cameras with respect to each other and with respect to the scene.

Darker areas in $|X'|$ and $|Y'|$ are indicative of areas that are going out of view horizontally and vertically from the sensed image to the reference image. Brighter

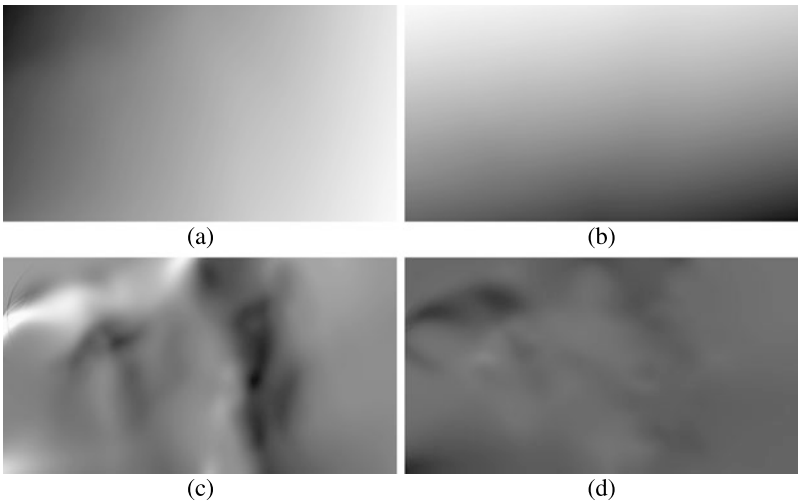


Fig. 9.25 (**a**), (**b**) The X-component and the Y-component of the transformation obtained by the weighted-linear (WLIN) method to register the Mountain images. (**c**), (**d**) Plots of $|X'|$ and $|Y'|$. Intensities in the images have been appropriately scaled for better viewing

areas show regions that are coming into view and expanding in size in the sensed image when compared to the reference image. Such regions point towards the view while darker regions point away from the view. The sensed image, therefore, has been obtained to the left of the reference image. Using the transformation function obtained for the registration of two images, some characteristics of the scene as well as the relation between the cameras and the scene can be determined.

9.4 Evaluation

Various interpolating/approximating functions suitable for representing the components of a transformation function in image registration were discussed. Each transformation has its strengths and weaknesses. It is hard to find a single transformation function that performs the best on all types of images; however, there are transformation functions that perform better than others on many image types. The desired properties of a transformation function for image registration are:

1. *Monotonicity*, *convexity*, *and nonnegativity preserving*: These properties ensure that the function is well behaved and it does not produce high fluctuations and overshoots away from the control points. The properties can be obtained by formulating the surface in terms of not only the data values but also the data gradients at the points. The properties are easier to achieve when a function is formulated in such a way that its variations can be more easily controlled. Lu and Schumaker [62] and Li [59] derived monotonicity-preserving conditions, Renka [79] and Lai [49] derived convexity-preserving conditions, and Schumaker and

Speleers [94] and Hussain and Hussain [45] derived nonnegativity preserving conditions for piecewise smooth surface interpolation to scattered data. These methods typically constrain gradient vectors at the points to ensure a desired property in the created surface.

2. *Linearity preserving*: If data values in the image domain vary linearly, the function interpolating/approximating the data should also vary linearly. This property ensures that a transformation function would not introduce nonlinearity into the resampling process when corresponding reference and sensed areas are related linearly.
3. *Adaptive to the density and organization of points*: Since control points in an image are rarely uniformly spaced, a transformation function should have the ability to adapt to the local density and organization of the points. Density of points across the image domain can vary greatly and so can the spacing between the points. If the transformation function is defined by radial basis functions, the widths of the functions should adapt to the local density of points and the shape of the basis functions should adapt to the irregular spacing of the points. Generally, monotonically decreasing rational basis functions adapt well to the organization of points. Rational basis functions, however, should be used in parametric form. If used in explicit form, flat spots appear in the components of the transformation, producing large errors in registration.

To determine the strengths and weaknesses of the transformation functions described in this chapter and to determine their performances in image registration, experiments were carried out using the images depicted in Fig. 9.26. Corresponding points in the images are also shown. The images have various degrees of geometric differences.

Images (a) and (b) are captured from different views and different distances of an art piece. They are of dimensions 520×614 and 505×549, respectively. The geometric difference between the images varies from point to point. We will refer to these images as *Face* images. The images contain 80 corresponding points. Images (c) and (d) show aerial images, again, taken from different views and different distances to the scene. They are of dimensions 412×244 and 469×274, respectively. The images contain small local and global geometric differences. We will refer to them as *Aerial* images. There are 31 corresponding points in these images.

Images (e) and (f) show two views of a terrain. These images are of dimensions 655×438 and 677×400, respectively. There is depth discontinuity near the center of the images at about 120 degrees. There are 46 corresponding points in the images. We will call these *Terrain* images. Images (g) and (h) show a close up of a small area in the terrain. The images are of dimensions 409×531 and 402×542, respectively. There are 58 corresponding points in these images. These images will be referred to as the *Rock* images. The geometric difference between these images vary across the image domain.

Images (i) and (j) show two views of a partially snow-covered, rocky mountain. These images are of dimensions 719×396 and 565×347, respectively. There are 165 corresponding points in theses images. This is called the *Mountain* data set. The geometric difference between these images varies considerably across the image

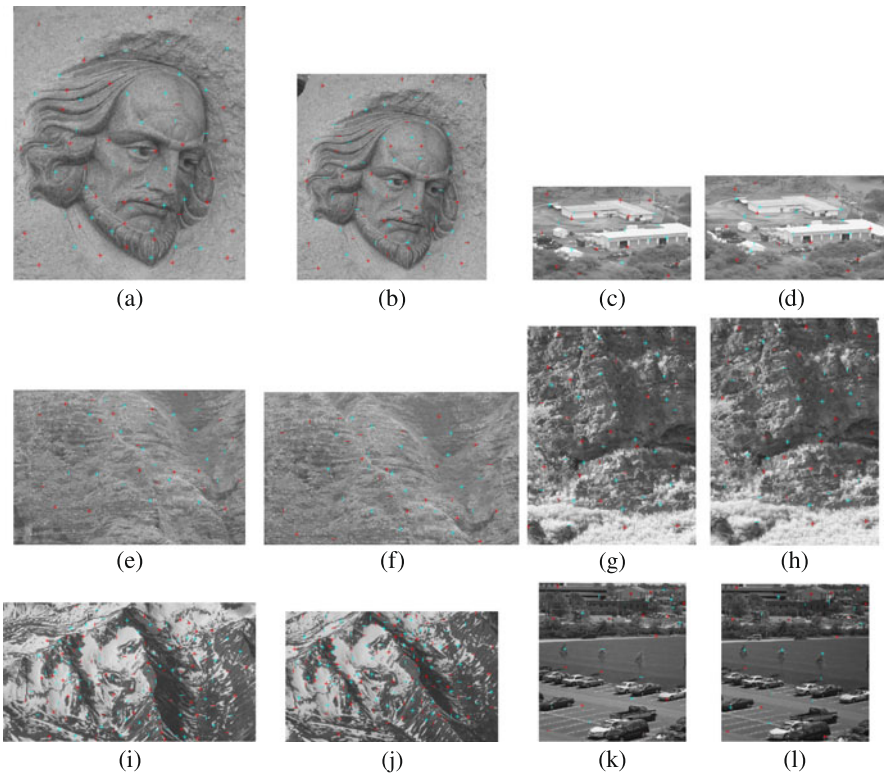


Fig. 9.26 (**a**), (**b**) Face, (**c**), (**d**) Aerial, (**e**), (**f**) Terrain, (**g**), (**h**) Rock, (**i**), (**j**) Mountain, and (**k**), (**l**) Parking images used to evaluate the performances of various transformation functions in image registration. The number of corresponding points in these image sets are 80, 31, 46, 58, 165, and 32. The control points are marked with '+' in the images. Points marked in *red* are used to determine the transformation parameters, and points marked in *light blue* are used to quantify registration accuracy

domain. Finally, (k) and (l) are images of a parking lot taken from the same viewpoint but with different view angles. These images are of dimensions 450×485 and 449×480, respectively. They contain only global geometric differences, defined by a projective transformation. Local geometric differences between the images are negligible. The images contain 32 corresponding points. We will refer to these images as the *Parking* images.

The control points in these images were determined using the Harris point detector and correspondence between the points were determined by the coarse-to-fine matching Algorithm F5 in Chap. 7 using error tolerance of 1.5 pixels.

We will compare the speeds and accuracies of various transformation functions in the registration of these images using the provided correspondences. For each transformation, the time to determine its parameters and the time to resample the sensed image to the geometry of the reference image are determined. Since the true transformation function between the images is not known, we will use half of

the correspondences to determine the transformation and use the remaining half to measure the registration accuracy. Points marked in red in Fig. 9.26 are used to determine a transformation and points marked in light blue are used to determine the registration accuracy with the obtained transformation.

The transformation functions used in this evaluation are (1) multiquadric, (2) surface or thin-plate spline, (3) Wendland's compactly supported interpolation (9.55), (9.56), (4) Maude's local weighted linear (9.61), (5) moving least squares (9.65) using polynomials of degree 1 and inverse square distance weights (9.66), (6) piecewise-linear interpolation, (7) approximating subdivision surface of Loop, (8) parametric Shepard interpolation using rational Gaussian weights with smoothness parameter $s = 0.75$ (9.92)–(9.95), (9) weighted-linear approximation (9.98), and (10) interpolating implicit surface (9.126) with Euclidean ($\|\mathbf{p} - \mathbf{p}_i\|$) basis functions without a linear term.

Results are tabulated in Table 9.3. Examining the results, we see that surface or thin-plate spline (TPS) has the highest speed in spite of the fact that it solves a global system of equations to find each component of a transformation. A single method could not produce the best RMSE for all images and methods vary in accuracy depending on the organization of the points and the severity of the geometric difference between the images.

For images with small to moderate geometric differences, Maude's weighted linear approximation (MAUD) produced the best result, while for images with large local geometric differences, Loop subdivision method (LOOP) and implicit interpolation produced the smallest MAX errors. Weighted-linear (WLIN) and parametric Shepard (SHEP) also produce low MAX errors.

Considering both speed and accuracy, overall best results are obtained by moving least-squares (MLQ) followed by weighted-linear (WLIN) and parametric Shepard (SHEP). These methods are especially attractive because they have the ability to resample image regions outside the convex hull of the control points. Registration results for the six image sets in Fig. 9.26 by the moving least-square method are shown in Fig. 9.27 for qualitative evaluation. The reference image is shown in the red and blue bands and the sensed image is shown in the green band of a color image. At pixels where the images perfectly align gray values are obtained, and at pixels where the images do not align well green or purple are obtained. Scene areas visible in only one of the images also appear in green or purple.

9.5 Final Remarks

To register two images, not only is a set of corresponding points in the images required, a transformation function is required that can use information about the correspondences to find the geometric relations between the images. A transformation function makes it possible to spatially align the images and determine the correspondence between all points in the images. It also provides the means to infer the geometric characteristics of the underlying scene.

Table 9.3 Performance measures for various transformation functions used to register the images shown in Fig. 9.26. The transformation functions tested are: multiquadric (MQ), surface or thin-plate spline (TPS), Wendland's compactly supported radial basis functions (WEND), Maude's local weighted linear formula (MAUD), moving least squares (MLQ), piecewise linear (PWL), Loop subdivision surface (LOOP), parametric Shepard interpolation (SHEP), weighted linear approximation (WLIN), and interpolative implicit surface with Euclidean basis functions (IMPL). Performance measures are: computation time (TIME) in seconds, root-mean-squared error (RMSE) in pixels, and maximum (MAX) registration error, also in pixels. The transformation parameters are determined using half of the provided control-point correspondences and registration errors are determined using the remaining correspondences. Best results are shown in **bold**

Method	Measure	Face	Aerial	Terrain	Rock	Mountain	Parking
MQ	TIME	1.34	0.19	0.73	0.70	2.48	0.39
	RMSE	4.05	6.80	10.28	4.08	4.62	5.89
	MAX	9.00	13.89	26.38	9.10	30.62	14.33
TPS	TIME	**1.09**	**0.14**	**0.58**	**0.61**	**1.93**	**0.31**
	RMSE	3.85	1.34	2.16	1.51	4.47	0.98
	MAX	10.68	2.43	4.26	3.34	32.18	1.79
WEND	TIME	1.54	0.23	0.81	0.81	3.28	0.48
	RMSE	3.59	5.22	5.59	4.22	4.57	6.71
	MAX	7.26	10.01	12.57	12.16	30.05	12.88
MAUD	TIME	4.32	1.06	3.30	2.64	5.32	2.31
	RMSE	4.09	**1.07**	**1.38**	1.50	**4.40**	**0.93**
	MAX	9.34	**1.88**	3.12	3.35	27.55	1.69
MLQ	TIME	1.98	0.41	1.15	1.06	3.35	0.67
	RMSE	3.96	1.16	1.62	1.52	5.46	0.95
	MAX	9.32	2.13	3.40	3.69	33.17	**1.45**
PWL	TIME	2.20	0.30	1.26	1.16	4.71	0.61
	RMSE	4.08	1.28	1.70	1.48	4.55	0.94
	MAX	10.94	2.49	4.33	3.23	30.10	1.47
LOOP	TIME	6.86	7.16	8.5	6.68	4.52	8.13
	RMSE	4.13	1.24	**1.45**	1.59	4.46	0.95
	MAX	10.33	2.53	3.61	3.75	**25.49**	1.64
SHEP	TIME	1.93	0.27	1.05	1.05	2.98	0.69
	RMSE	4.32	1.38	1.79	1.59	4.91	1.13
	MAX	11.64	2.35	5.10	**3.04**	33.97	1.70
WLIN	TIME	1.95	0.27	1.01	1.03	3.06	0.69
	RMSE	4.25	1.28	1.58	1.51	4.47	0.96
	MAX	12.96	2.44	**3.01**	3.33	26.29	1.72
IMPL	TIME	6.80	0.91	3.74	3.50	12.22	2.01
	RMSE	**3.48**	5.58	8.55	3.93	4.43	6.14
	MAX	**6.96**	14.23	27.75	13.46	28.8	15.30

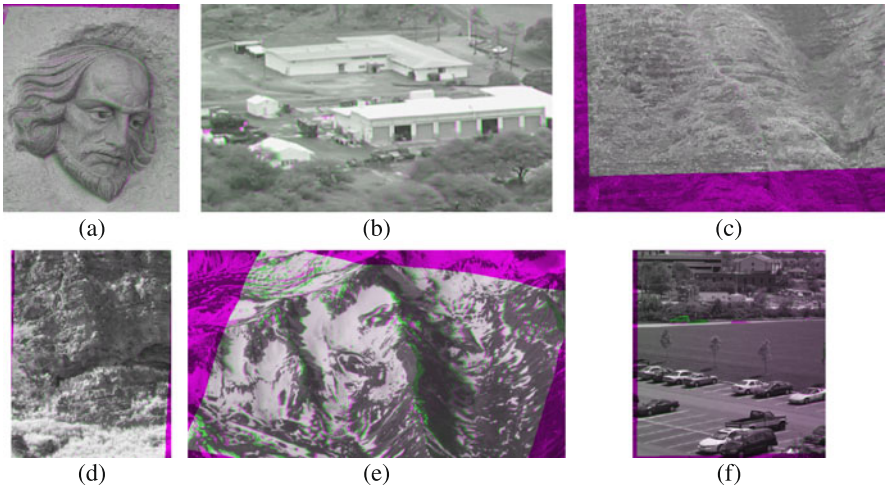


Fig. 9.27 (**a**)–(**f**) Registration of the Face, Aerial, Terrain, Rock, Mountain, and Parking images by the moving least-squares transformation function

If the geometry of a scene and the relations of the cameras to each other and to the scene are known, the type of transformation function most suitable to relate the geometries of the images can be selected. The parameters of the transformation can then be determined from the coordinates of corresponding points in the images. However, often information about the scene and the cameras is not available. In such a situation, the employed transformation function should be able to adapt to the local geometric differences between the images.

Comparing the performances of a number of adaptive transformation functions on various images with varying degrees of local and global geometric differences, we observe that although a single transformation does not exist that can outperform all other transformations, but some transformations clearly perform better than others. Among the tested transformation functions, weighted-linear, moving least-squares, and parametric Shepard methods generally perform better than other methods in both speed and accuracy.

The quality of a resampled image depends on the resampling method used. Image resampling is discussed in the next chapter. When registering two images, there is sometimes a need to combine the images into a larger image mosaic. To create a seamless mosaic, intensities in the overlap area in the images should be blended in such a way that intensities in the images smoothly merge. Image blending is also discussed in the next chapter.

References

1. Adamson, A., Alexa, M.: On normals and projection operators for surfaces defined by point sets. In: Eurographics Symposium on Point-based Graphics, pp. 149–155 (2004)

2. Akima, H.: A method of bivariate interpolation and smooth surface fitting for irregularly distributed data points. ACM Trans. Math. Softw. **4**, 148–159 (1978)
3. Arge, E., Dæhlen, M., Tveito, A.: Approximation of scattered data using smooth grid functions. J. Comput. Appl. Math. **59**, 191–205 (1995)
4. Belytschko, T., Krongauz, Y., Organ, D., Fleming, M., Krysl, P.: Meshless methods: An overview and recent developments. Comput. Methods Appl. Mech. Eng. **139**, 3–47 (1996)
5. Bertram, M., Barnes, J.C., Hamann, B., Joy, K.I., Pottmann, H., Wushour, D.: Piecewise optimal triangulation for the approximation of scattered data in the plane. Comput. Aided Geom. Des. **17**, 767–787 (2000)
6. Bookstein, F.L.: Principal warps: Thin-plate splines and the decomposition of deformations. IEEE Trans. Pattern Anal. Mach. Intell. **11**(6), 567–585 (1989)
7. Bozzini, M., Lenarduzzi, L., Rossini, M.: Polyharmonic splines: An approximation method for noisy scattered data of extra-large size. Appl. Math. Comput. **216**, 317–331 (2010)
8. Brown, J.L.: Vertex based data dependent triangulations. Comput. Aided Geom. Des. **8**, 239–251 (1991)
9. Buhmann, M.D.: A new class of radial basis functions with compact support. Math. Comput. **70**(233), 307–318 (2000)
10. Carlson, R.E., Foley, T.A.: The parameter R^2 in multiquadric interpolation. Comput. Math. Appl. **21**(9), 29–42 (1991)
11. Carr, J.C., Beatson, R.K., Cherrie, J.B., Mitchell, T.J., Fright, W.R., McCallum, B.C., Evans, T.R.: Reconstruction and representation of 3D objects with radial basis functions. In: Proc. SIGGRAPH '01 Conf., pp. 67–76 (2001)
12. Chang, L.H.T., Said, H.B.: A C^2 triangular patch for the interpolation of functional scattered data. Comput. Aided Des. **29**(6), 407–412 (1997)
13. Choi, Y.-L., Yoo, K.-W., Cho, N.-I., Lee, J.-H.: Line based image matching method. US Patent 9,884,079, Filed 20 Jun. 2001, Patented 29 Aug. 2002
14. Chui, C.K., Lai, M.-J.: Filling polygonal holes using C^1 cubic triangular spline patches. Comput. Aided Geom. Des. **17**, 297–307 (2000)
15. Constantini, P., Manni, C.: On a class of polynomial triangular macro-elements. Comput. Appl. Math. **73**, 45–64 (1996)
16. Dahmen, W., Meyling, R.H.J.G., Ursem, J.H.M.: Scattered data interpolation by bivariate C^1-piecewise quadratic functions. Approx. Theory Appl. **6**(3), 6–29 (1990)
17. Davydov, O., Schumaker, L.L.: Stable approximation and interpolation with C^1 quartic bivariate splines. SIAM J. Numer. Anal. **39**(5), 1732–1748 (2002)
18. Doo, D.W.H.: A subdivision algorithm for smoothing down irregular shaped polyhedrons. In: Proc. Interactive Techniques in Computer Aided Design, vol. 1, pp. 157–165 (1978)
19. Doo, D., Sabin, M.: Behavior of recursive division surfaces near extraordinary points. In: Computer Aided Design, pp. 356–360 (1978)
20. Duchon, J.: Splines minimizing rotation-invariant seminorms in Sobolov spaces. In: Constructive Theory of Functions of Several Variables. Lecture Notes in Math., vol. 571, pp. 85–100. Springer, Berlin (1977)
21. Dyn, N., Levin, D., Rippa, S.: Algorithms for the construction of data dependent triangulation. In: Mason, J.S., Cox, M.G. (eds.) Algorithms for Approximation II, pp. 185–192. Chapman and Hall, New York (1988)
22. Dyn, N., Levin, D., Gregory, J.A.: A butterfly subdivision scheme for surface interpolation with tension control. ACM Trans. Graph. **9**(2), 160–169 (1990)
23. Floater, M.S., Iske, A.: Multistep scattered data interpolation using compactly supported radial basis functions. J. Comput. Appl. Math. **73**, 65–78 (1996)
24. Fornefett, M., Rohr, K., Stiehl, H.S.: Radial basis functions with compact support for elastic registration of medical images. Image Vis. Comput. **19**, 87–96 (2001)
25. Franke, R.: Scattered data interpolation: Tests of some methods. Math. Comput. **38**(157), 181–200 (1982)
26. Franke, R.: Thin plate splines with tension. Comput. Aided Geom. Des. **2**, 87–95 (1985)
27. Franke, R., Nielson, G.: Smooth interpolation of large sets of scattered data. Int. J. Numer. Methods Eng. **15**, 1691–1704 (1980)

28. Franke, R., Schumaker, L.L.: A bibliography of multivariate approximation. In: Chui, C., Schumaker, L., Utrerus, F. (eds.) Topics in Multivariate Approximation, pp. 79–98. Academic Press, San Diego (1987)
29. Franke, R., Hagen, H., Nielson, G.M.: Least squares surface approximation to scattered data using multiquadric functions. Adv. Comput. Math. **2**, 81–99 (1994)
30. Goodman, T.N.T., Said, H.B., Chang, L.H.T.: Local derivative estimation for scattered data interpolation. Appl. Math. Comput. **68**, 41–50 (1995)
31. Goshtasby, A.: Piecewise linear mapping functions for image registration. Pattern Recognit. **19**(6), 459–466 (1986)
32. Goshtasby, A.: Piecewise cubic mapping functions for image registration. Pattern Recognit. **20**(5), 525–533 (1987)
33. Goshtasby, A.: Registration of image with geometric distortion. IEEE Trans. Geosci. Remote Sens. **26**(1), 60–64 (1988)
34. Goshtasby, A.: A weighted linear method for approximation of irregularly spaced data. In: Lucian, M.M., Neamtu, M. (eds.) Geometric Modeling and Computing, pp. 285–294. Nashboro Press, Brentwood (2004)
35. Goshtasby, A.: Surface approximation to scattered lines. Comput-Aided Des. Appl. **4**(1–4), 277–286 (2007)
36. Goshtasby, A.: Registration of multi-view images. In: LeMoigne, J., Netanyahoo, N.S., Eastman, R.D. (eds.) Image Registration for Remote Sensing, pp. 153–178. Cambridge University Press, Cambridge (2011)
37. Goshtasby, A.: Design and recovery of 2-D and 3-D shapes using rational Gaussian curves and surfaces. Int. J. Comput. Vis. **10**(3), 233–256 (1993)
38. Green, P.J., Sibson, R.: Computing Dirichlet tessellation in the plane. Comput. J. **21**, 168–173 (1978)
39. Greiner, G., Kolb, A., Riepl, A.: Scattered data interpolation using data dependent optimization techniques. Graph. Models **64**, 1–18 (2002)
40. Grosse, E.: A catalogue of algorithms for approximation. In: Mason, J., Cox, M. (eds.) Algorithms for Approximation II, pp. 479–514. Chapman and Hall, London (1990)
41. Harder, R.L., Desmarais, R.N.: Interpolation using surface splines. J. Aircr. **9**(2), 189–191 (1972)
42. Hardy, R.L.: Multiquadric equations of topography and other irregular surfaces. J. Geophys. Res. **76**(8), 1905–1915 (1971)
43. Hardy, R.L.: Theory and applications of the multiquadric-biharmonic method—20 years of discovery—1969–1988. Comput. Math. Appl. **19**(8/9), 163–208 (1990)
44. Hoppe, H., DeRose, T., Duchamp, T., Halstead, M., Jin, H., McDonald, J., Schweitzer, J., Stuetzle, W.: Piecewise smooth surface reconstruction. In: SIGGRAPH'94: Proc. 21st Annual Conference on Computer Graphics and Interactive Techniques, pp. 295–302 (1994)
45. Hussain, M.Z., Hussain, M.: C^1 positive scattered data interpolation. Comput. Math. Appl. **59**, 457–567 (2010)
46. Kamgar-Parsi, B., Kamgar-Parsi, B.: Algorithms for matching 3D line sets. IEEE Trans. Pattern Anal. Mach. Intell. **26**(5), 582–593 (2004)
47. Kansa, E.J., Carlson, R.E.: Improved accuracy of multiquadric interpolation using variable shape parameters. Comput. Math. Appl. **24**, 99–120 (1992)
48. Klucewicz, I.M.: A piecewise C^1 interpolant to arbitrarily spaced data. Comput. Graph. Image Process. **8**, 92–112 (1978)
49. Lai, M.-J.: Convex preserving scattered data interpolation using bivariate C^1 cubic spline. J. Comput. Appl. Math. **119**, 249–258 (2000)
50. Lai, M.-J., Wenston, P.: L_1 spline methods for scattered data interpolation and approximation. Adv. Comput. Math. **21**, 293–315 (2004)
51. Lancaster, P.: Moving weighted least-squares methods. In: Sahney, B.N. (ed.) Polynomial and Spline Approximation, pp. 103–120 (1979)
52. Lancaster, P., Šalkauskas, K.: Surfaces generated by moving least squares methods. Math. Comput. **37**(155), 141–158 (1981)

53. Lancaster, P., Šalkauskas, K.: Curve and Surface Fitting: An Introduction. Academic Press, San Diego (1986), pp. 55–62, 225–244
54. Lawson, C.L.: Software for C^1 surface interpolation. In: Rice, J.R. (ed.) Mathematical Software III, pp. 161–194. Academic Press, San Diego (1977)
55. Lazzaro, D., Montefusco, L.B.: Radial basis functions for the multivariate interpolation of large scattered data sets. J. Comput. Appl. Math. **140**, 521–536 (2002)
56. Lee, D.T., Schachter, B.J.: Two algorithms for constructing a Delaunay triangulation. Int. J. Comput. Inf. Sci. **9**, 219–242 (1980)
57. Lee, S., Wolberg, G., Shin, S.Y.: Scattered data interpolation with multilevel B-splines. IEEE Trans. Vis. Comput. Graph. **3**(3), 228–244 (1997)
58. Levin, D.: The approximation power of moving least-squares. Math. Comput. **67**(224), 1517–1531 (1998)
59. Li, A.: Convexity preserving interpolation. Comput. Aided Geom. Des. **16**, 127–147 (1999)
60. Loop, C.: Smooth subdivision surfaces based on triangles. Master's thesis, Department of Mathematics, University of Utah (1987)
61. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3D surface construction algorithm. In: Proc. SIGGRAPH, pp. 71–78. ACM Press/ACM SIGGRAPH, New York (1992)
62. Lu, H., Schumaker, L.L.: Monotone surfaces to scattered data using C^1 piecewise cubics. SIAM J. Numer. Anal. **34**(2), 569–585 (1997)
63. Luo, Z., Peng, X.: A C^1-rational spline in range restricted interpolation of scattered data. J. Comput. Appl. Math. **194**, 255–266 (2006)
64. Maillot, J., Stam, J.: A unified subdivision scheme for polygonal modeling. In: Chalmers, A., Rhyne, T.-M. (eds.) EUROGRAPHICS, vol. 20(3) (2001)
65. Marinov, M., Kobbelt, L.: Optimization methods for scattered data approximation with subdivision surfaces. Graph. Models **67**, 452–473 (2005)
66. Maude, A.D.: Interpolation–mainly for graph plotters. Comput. J. **16**(1), 64–65 (1973)
67. McLain, D.H.: Two dimensional interpolation from random data. Comput. J. **19**(2), 178–181 (1976)
68. Meijering, E.: A chronology of interpolation: From ancient astronomy to modern signal and image processing. Proc. IEEE **90**(3), 319–342 (2002)
69. Meinguet, J.: An intrinsic approach to multivariate spline interpolation at arbitrary points. In: Sahney, B.N. (ed.) Polynomial and Spline Approximation, pp. 163–190. Reidel, Dordrecht (1979)
70. Nejhum, S.M.S., Chi, Y.-T., Yang, M.-H.: Higher-dimensional affine registration and vision applications. IEEE Trans. Pattern Anal. Mach. Intell. **33**(7), 1324–1338 (2011)
71. Nielson, G.M.: Dual marching cubes. In: Proc. IEEE Visualization, pp. 489–496 (2004)
72. Ohtake, Y., Belyaev, A., Seidel, H.-P.: 3D scattered data interpolation and approximation with multilevel compactly supported RBFs. Graph. Models **67**, 150–165 (2005)
73. Percell, P.: On cubic and quartic Clough-Tocher finite elements. SIAM J. Numer. Anal. **13**, 100–103 (1976)
74. Peters, J., Reif, U.: The simplest subdivision scheme for smoothing polyhedra. ACM Trans. Graph. **16**(4), 420–431 (1997)
75. Powell, M.J.D.: Radial basis functions for multivariate interpolation: A review. In: Mason, J.C., Cox, M.G. (eds.) Algorithms for Approximation, pp. 143–167. Clarendon Press, Oxford (1987)
76. Powell, M.J.D., Sabin, M.A.: Piecewise quadratic approximation on triangles. ACM Trans. Math. Softw. **3**, 316–325 (1977)
77. Qu, R., Agarwal, R.P.: Smooth surface interpolation to scattered data using interpolatory subdivision algorithms. Comput. Math. Appl. **32**(3), 93–110 (1996)
78. Renka, R.J.: Multivariate interpolation of large sets of scattered data. ACM Trans. Math. Softw. **14**(2), 139–148 (1988)
79. Renka, R.J.: Algorithm 833: CSRFPAXK—Interpolation of scattered data with a C^1 convexity-preserving surface. ACM Trans. Math. Softw. **30**(2), 200–211 (2004)
80. Renka, R.J., Brown, R.: Algorithm 790: CSHEP2D: Cubic Shepard method for bivariate interpolation of scattered data. ACM Trans. Math. Softw. **25**(1), 70–73 (1999)

81. Renka, R.J., Brown, R.: Algorithm 791: TSHEP2D: Cosine series Shepard method for bivariate interpolation of scattered data. ACM Trans. Math. Softw. **25**(1), 74–77 (1999)
82. Renka, R.J., Brown, R.: Algorithm 792: Accuracy tests of ACM algorithms for interpolation of scattered data in the plane. ACM Trans. Math. Softw. **25**(1), 78–94 (1999)
83. Rippa, S.: Scattered data interpolation using minimum energy Powell-Sabin elements and data dependent triangulations. Numer. Algorithms **5**, 577–587 (1993)
84. Rivlin, T.J.: Least-squares approximation. In: An Introduction to the Approximation of Functions, pp. 48–61. Dover, New York (1969)
85. Rohr, K., Stiehl, H.S., Sprengel, R., Buzug, T.M., Weese, J., Kuhn, M.H.: Landmark-based elastic registration using approximating thin-plate splines. IEEE Trans. Med. Imaging **20**(6), 526–534 (2001)
86. Savchenko, V.V., Pasko, A.A., Okunev, O.G., Kunii, T.L.: Function representation of solids reconstructed from scattered surface points and contours. Comput. Graph. Forum **14**(4), 181–188 (1995)
87. Schagen, I.P.: The use of stochastic processes in interpolation and approximation. Int. J. Comput. Math., Sect. B **8**, 63–76 (1980)
88. Scheib, V., Haber, J., Lin, M.C., Seidel, H.-P.: Efficient fitting and rendering of large scattered data sets using subdivision surfaces. Eurographics **21**(3), 353–362 (2002)
89. Schmidt, J.W.: Scattered data interpolation applying regional C^1 splines on refined triangulations. Math. Mech. **80**(1), 27–33 (2000)
90. Schröder, P., Zorin, D.: Subdivision for modeling and animation. SIGGRAPH Course No. 36 Notes (1998)
91. Schumaker, L.L.: Triangulation methods. In: Chui, C.K., Schumaker, L.L., Utreras, F. (eds.) Topics in Multivariate Approximation, pp. 219–232. Academic Press, San Diego (1987)
92. Schumaker, L.L.: Computing optimal triangulations using simulated annealing. Comput. Aided Geom. Des. **10**, 329–345 (1993)
93. Schumaker, L.L.: Multivariate spline bibliography. In: Chui, C., Neamtu, M., Schumaker, L.L. (eds.) Approximation Theory XI: Gatlinburg. Nashboro Press, Brentwood (2005)
94. Schumaker, L.L., Speleers, H.: Nonnegativity preserving macro-element interpolation of scattered data. Comput. Aided Geom. Des. **27**(3), 245–261 (2010)
95. Schweitzer, J.E.: Analysis and application of subdivision surfaces. Ph.D. Dissertation, University of Washington, Seattle (1996)
96. Shepard, D.: A two-dimensional interpolation function for irregularly spaced data. In: Proc. 23rd Nat'l Conf. ACM, pp. 517–524 (1968)
97. Shirman, L.A., Sequin, C.H.: Local surface interpolation with shape parameters between adjoining Gregory patches. Comput. Aided Geom. Des. **7**, 375–388 (1990)
98. Stam, J.: On subdivision schemes generalizing uniform B-spline surfaces of arbitrary degree. Comput. Aided Geom. Des. **18**, 383–396 (2001)
99. Stead, S.E.: Estimation of gradients from scattered data. Rocky Mt. J. Math. **14**, 265–279 (1984)
100. Turk, G., O'Brien, J.F.: Modeling with implicit surfaces that interpolate. ACM Trans. Graph. **21**(4), 855–873 (2002)
101. Wang, L., Neumann, U., You, S.: Image matching using line signature. US Patent 12,486,506, Filed 17 Jun. 2009, Patented 23 Dec. 2010
102. Wendland, H.: Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. Adv. Comput. Math. **4**, 389–396 (1995)
103. Zorin, D.: Subdivision and multiresolution surface representations. Ph.D. Dissertation, Caltech, Pasadena (1997)
104. Zorin, D., Schröder, P., Sweldens, W.: Interpolating subdivision for meshes with arbitrary topology. In: Computer Graphics Proceedings (SIGGRAPH 96), pp. 189–192 (1996)