Chapter 10
Image Resampling and Compositing

10.1 Image Resampling

The transformation

$$X = f(x, y), \tag{10.1}$$

$$Y = g(x, y) \tag{10.2}$$

relates the coordinates of points in the reference image to the coordinates of corresponding points in the sensed image. Given the (x, y) coordinates of a point in the reference image, relations (10.1) and (10.2) determine the (X, Y) coordinates of the same point in the sensed image. By reading the intensity at (X, Y) in the sensed image and saving at (x, y) in a new image, the sensed image is point-by-point resampled to the geometry of the reference image. Therefore, to resample the sensed image, the reference image is scanned and, for each pixel (x, y), the corresponding point (X, Y) is determined in the sensed image. Although coordinates (x, y) are integers, coordinates (X, Y) are floating-point numbers. Since intensities at only integer coordinates are available in the sensed image, the intensity at point (X, Y) has to be estimated from the intensities of a small number of pixels surrounding (X, Y).

If the sensed image were a continuous image, the intensity of any point (X, Y) in the image would be known. However, a sensed image $I(u, v)$ contains only uniformly spaced samples of a continuous image $C(X, Y)$. A resampling method has to estimate the intensity at (X, Y) from the intensities at discrete locations surrounding it. Figure 10.1 depicts the resampling process. Pixel a in the reference image maps to point A in the continuous sensed image. To estimate the intensity at A, intensities in a small neighborhood of A in the discrete sensed image are used. Different methods to achieve this estimation have been developed. In the following sections, nearest-neighbor, bilinear interpolation, cubic convolution, cubic spline interpolation, and radially symmetric resampling methods are discussed.

A.A. Goshtasby, *Image Registration*,
Advances in Computer Vision and Pattern Recognition,
DOI 10.1007/978-1-4471-2458-0_10, © Springer-Verlag London Limited 2012

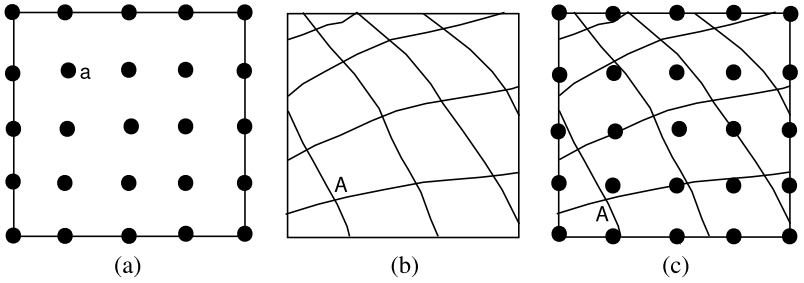


Fig. 10.1 The resampling process. (**a**) Pixels in the reference image. (**b**) A continuous sensed image. The *grid points* in this image correspond to the pixels in the reference image. (**c**) Overlaying of the continuous and discrete sensed images. The continuous sensed image is determined from intensities in the discrete sensed image. Resampling involves scanning the reference image and, for each pixel a, determining the intensity of the corresponding grid point A in the continuous sensed image. Intensity at A in the continuous image is estimated from the intensities of a small number of pixels surrounding A in the discrete sensed image

10.1.1 Nearest-Neighbor

Given the coordinates (X, Y) of a point, where X and Y are floating-point numbers, and assuming u is the integer part of X and v is the integer part of Y, the rectangular neighborhood defined by pixels (u, v), $(u, v+1)$, $(u+1, v)$, and $(u+1, v+1)$ contain point (X, Y). Among the four pixels, the one closest to (X, Y) is determined and its intensity is used as the intensity at (X, Y). There is actually no need to calculate the distances to achieve this. If $X - u < 0.5$, u is considered the X-coordinate of the pixel to be used, otherwise $u + 1$ is used. If $Y - v < 0.5$, the Y-coordinate of the pixel to be used is v, otherwise $v + 1$ is used. This is actually a rounding operation; therefore, the intensity at pixel $[round(X), round(Y)]$ is taken and considered the intensity at (X, Y).

Nearest neighbor resampling preserves the image intensities. Therefore, the histograms of an image before and after resampling will be very similar. If certain intensities do not exist in an image, they will not be present in the resampled image. The process does not blur an image, but it may produce aliasing effects. Horizontal and vertical edges in an image may appear jagged after rotation of the image by angles that are not a multiple of 90 degrees. Figure 10.2 shows a resampling example by the nearest-neighbor method. Figure 10.2a is an image of a penny and Fig. 10.2b is the penny after being rotated by 10 degrees counterclockwise about the image center and resampled by the nearest-neighbor method. The jagged appearance of the penny, especially along its boundary, is quite evident in the resampled image.

The computational complexity of nearest-neighbor resampling is on the order of n comparisons if reference image containing n pixels.

(a) (b)

Fig. 10.2 (**a**) A penny and (**b**) its rotation by 10 degrees counterclockwise when using nearest-neighbor resampling

10.1.2 Bilinear Interpolation

In bilinear interpolation, the intensity at a point is determined from the weighted sum of intensities of the four pixels closest to it. Therefore, given location (X, Y) and assuming u is the integer part of X and v is the integer part of Y, the intensity at (X, Y) is estimated from the intensities at (u, v), $(u + 1, v)$, $(u, v + 1)$, and $(u + 1, v + 1)$. This resampling involves first finding the intensity at (X, v) from the linear interpolation of intensities at (u, v) and $(u + 1, v)$. Let this intensity be $I(X, v)$. Then, finding the intensity at $(X, v + 1)$ from the linear interpolation of intensities at $(u, v + 1)$ and $(u + 1, v + 1)$. Let this intensity be $I(X, v + 1)$. Finally, finding the intensity at (X, Y) from the linear interpolation of intensities at (X, v) and $(X, v + 1)$. This can be summarized as

$$\begin{aligned} I(X, Y) = W_{u,v} I(u, v) + W_{u+1,v} I(u + 1, v) + W_{u,v+1} I(u, v + 1) \\ + W_{u+1,v+1} I(u + 1, v + 1), \end{aligned} \tag{10.3}$$

where

$$W_{u,v} = (u + 1 - X)(v + 1 - Y), \tag{10.4}$$

$$W_{u+1,v} = (X - u)(v + 1 - Y), \tag{10.5}$$

$$W_{u,v+1} = (u + 1 - X)(Y - v), \tag{10.6}$$

$$W_{u+1,v+1} = (X - u)(Y - v), \tag{10.7}$$

and $I(u, v)$, $I(u + 1, v)$, $I(u, v + 1$, and $I(u + 1, v + 1)$ are intensities at (u, v), $(u + 1, v)$, $(u, v + 1)$, and $(u + 1, v + 1)$, respectively. Figure 10.3a depicts this estimation process. Figure 10.3b shows resampling of Fig. 10.2a after rotation counterclockwise by 10 degrees about the image center by bilinear interpolation. Compared to Fig. 10.2b, we see that the aliasing effect has mostly disappeared. One should also note that this anti-aliasing is at the cost of blurring the image. Therefore, if repeated resampling of an image is required, the process will gradually smooth image details. Bilinear interpolation changes image intensities, thereby changing the histogram of the image.

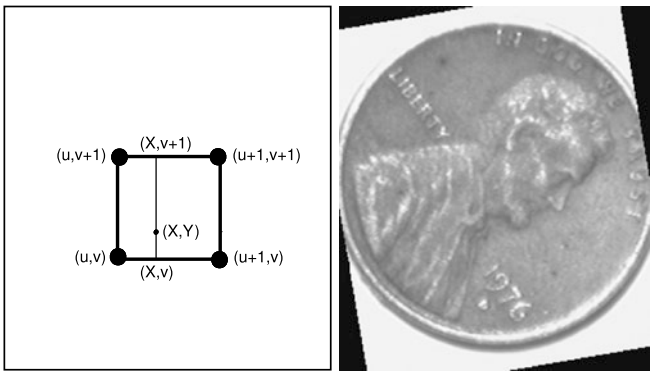


Fig. 10.3 (**a**) Estimating the intensity at (X, Y) from intensities at (u, v), $(u + 1, v)$, $(u, v + 1)$, and $(u + 1, v + 1)$ by bilinear interpolation. (**b**) The penny shown in Fig. 10.2a after rotation by 10 degrees counterclockwise about its center and resampling by bilinear interpolation

Computationally, resampling by bilinear interpolation requires on the order of n multiplications if the reference image contains n pixels. Therefore, nearest-neighbor and bilinear interpolation have the same computational complexity, although nearest-neighbor is several times faster than bilinear interpolation.

10.1.3 Cubic Convolution

In cubic convolution, the intensity at point (X, Y) is estimated from the intensities of a 4×4 grid of pixels closest to it as depicted in Fig. 10.4a. Just like a separable 2-D convolution that can be performed via 1-D convolutions row-by-row and then column-by-column, cubic convolution can be carried out in 1-D first along the rows and then along the columns [8]. Therefore, first, 1-D interpolation is carried out along the four image rows in the 4×4 neighborhood of (X, Y). This will determine values at $(X, v - 1)$, (X, v), $(X, v + 1)$, and $(X, v + 2)$. Then interpolation is carried out along column X to find the intensity at (X, Y). Figure 10.4a depicts this computation graphically.

If u denotes the integer part of X in a 1-D image, and assuming intensities at $u - 1$, u, $u + 1$, and $u + 2$ are $I(u - 1)$, $I(u)$, $I(u + 1)$, and $I(u + 2)$, respectively, a function $f(t)$ that interpolates the four intensities should satisfy $f(t_i) = I(t_i)$, where t_i is one of $u - 1$, u, $u + 1$, or $u + 2$. Function f is defined in terms of a weighted sum of four local functions, the weights representing the intensities at the pixels. That is,

$$f(X) = I(u - 1) f_{-1} + I(u) f_0 + I(u + 1) f_1 + I(u + 2) f_2, \tag{10.8}$$

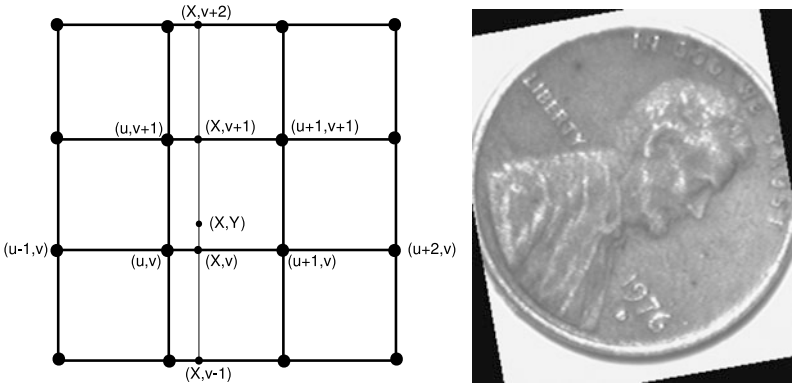


Fig. 10.4 (**a**) Cubic convolution uses intensities in a 4×4 neighborhood to estimate the intensity at a point. To determine the intensity at (X, Y), first 1-D interpolation is carried out along the 4 rows to determine intensities at X in each row. Then 1-D interpolation is carried out along column X to determine the intensity at (X, Y). (**b**) Resampling of Fig. 10.2a by cubic convolution

where

$$f_{-1} = -\frac{1}{2}t^3 + t^2 - \frac{1}{2}t, \tag{10.9}$$

$$f_0 = \frac{3}{2}t^3 - \frac{5}{2}t^2 + 1, \tag{10.10}$$

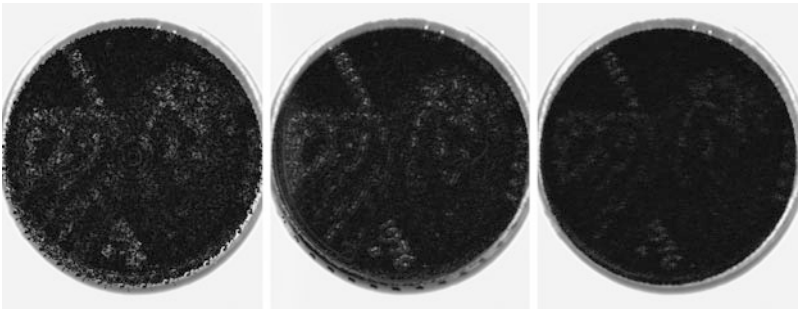
$$f_1 = -\frac{3}{2}t^3 + 2t^2 + \frac{1}{2}t, \tag{10.11}$$

$$f_2 = \frac{1}{2}t^3 - \frac{1}{2}t^2, \tag{10.12}$$

and $t = X - u$. Note that the sum of the local functions for any t in the range from 0 to 1 is 1. $f(X)$ evaluates to $I(u)$ when $X = u$ and it evaluates to $I(u + 1)$ when $X = u + 1$. In a 1-D image with n pixels, the pixel positions are $u = 0, 1, \ldots, n - 1$ with corresponding intensities $I(u)$. At the two image borders, it is assumed that $I(-1) = 3I(0) - 3I(1) + I(2)$ and $I(n) = 3I(n - 1) - 3I(n - 2) + I(n - 3)$. Resampling of the coin image of Fig. 10.2a after rotation counterclockwise by 10 degrees by cubic convolution is shown in Fig. 10.4b.

Although resampling by cubic convolution appears visually similar to that of bilinear interpolation, by a closer examination it becomes apparent that intensities estimated by the two methods are not the same. An example is given in Fig. 10.5. The image of Fig. 10.2a is rotated by 10 degrees 36 times. This is expected to produce the original image if no resampling errors existed. However, due to resampling errors, the obtained image is different from the original one. The absolute difference between the intensities of the obtained image and the intensities of the original image show the magnitude of resampling error.

Figures 10.5a–c show absolute errors when nearest-neighbor, bilinear interpolation, and cubic convolution, respectively, are used. Only errors within the circle



(a) (b) (c)

Fig. 10.5 (**a**)–(**c**) Resampling errors when rotating the image in Fig. 10.2a 36 times with 10-degree increments and computing the absolute difference between corresponding pixels in the original and resampled images using nearest-neighbor, bilinear interpolation, and cubic convolution, respectively. Higher intensities show larger errors

touching the image borders can be computed. As the image is rotated, areas near image corners move outside the image, causing them to be cut off. Consequently, after a 360-degree rotation, only pixels within the largest circle that can be contained in the image remain. Pixels outside this circle will all assume a value of zero. Therefore, when subtracting the resampled image from the original image, intensities of pixels outside this circle become equal to the intensities of the original image. This experiment reveals that among the three methods, cubic convolution is the most accurate method, while nearest-neighbor is the least accurate method.

Computational complexity of cubic convolution is $O(n)$ multiplications when reference image contains n pixels. Cubic convolution is several times slower than bilinear interpolation.

10.1.4 Cubic Spline

Given intensities $\{I_i : i = -1, 0, 1, 2\}$ of pixels at $\{u_i : i = -1, 0, 1, 2\}$ in a 1-D image, the intensity at point $0 \leq u < 1$ in the image can be estimated using a B-spline curve of order four (degree three) from:

$$f(u) = \sum_{i=-1}^{2} I_i b_i(u), \tag{10.13}$$

where

$$b_{-1}(u) = \left(-u^3 + 3u^2 - 3u + 1\right)/6, \tag{10.14}$$

$$b_0(u) = \left(3u^3 - 6u^2 + 4\right)/6, \tag{10.15}$$

$$b_1(u) = \left(-3u^3 + 3u^2 + 3u + 1\right)/6, \tag{10.16}$$

$$b_2(u) = u^3/6 \tag{10.17}$$

are the B-spline basis functions of order 4 and I_is are the control points of the curve. Note that the sum of the B-spline basis functions is 1 everywhere in the range $0 \leq u \leq 1$. The basis functions, when evaluated at a particular point u, show the contributions of the control points in the computation of the intensity at that point. The intensity at point u in a 1-D image is, therefore, a weighted sum of the intensities of the four pixels closest to it.

Formula (10.13) shows an approximation, thus $f(u)$ will only approximately evaluate to the intensities at the pixels. In order to interpolate the intensities, it is required to find new intensities such that, when they are used as the control points, the obtained B-spline curve will evaluate to the intensities at the pixels. That is $\{I'_j : j = -1, \ldots, 2\}$ should be determined such that

$$I_i = \sum_{j=-1}^{2} I'_j b_j(u_i), \quad i = -1, \ldots, 2. \tag{10.18}$$

Since two adjacent B-spline segments of order four share three control points (pixel intensities in our case), we cannot determine the control points of an interpolating B-spline by repeated use of formula (10.18). Instead, it is required to determine the entire set of control points collectively. This would require the solution of a system of n equations if a 1-D image with n pixels is given.

Estimation of the intensity at point (u, v) requires use of intensities of pixels at the 4×4 grid closest to the point. To make this estimation, first, new intensities should be determined so that when used as the control points in a bicubic B-spline surface, the obtained surface will interpolate the given intensities. Computation of the control points of a bicubic B-spline surface interpolating the intensities in an image with n pixels requires the solution of a system of n equations. This computation is on the order of n^2 multiplications. A parallel algorithm to calculate the control points of an interpolating B-spline surface with a smaller computational complexity is given by Cheng and Goshtasby [1].

Properties of B-splines as filters have been explored by Ferrari et al. [3]. Noticing that a B-spline curve acts like a digital filter when intensities of uniformly spaced pixels in a 1-D image are used as its control points; the control points of the B-spline curve interpolating the intensities can be obtained through an inverse filtering operation [6]. This involves finding the Fourier transform of the image and dividing it point-by-point by the Fourier transform of the B-spline filter. The inverse Fourier transform of the result will produce the control points of the interpolating B-spline. If FFT algorithm is used to compute the Fourier transform coefficients, it will take on the order of $n \log n$ multiplications to find the n control points of the interpolating B-spline. This is a reduction in computation time from $O(n^2)$ to $O(n \log n)$ multiplications.

The cubic spline method is computationally more expensive than cubic convolution; however, cubic spline has been found to produce more accurate results than cubic convolution [7].

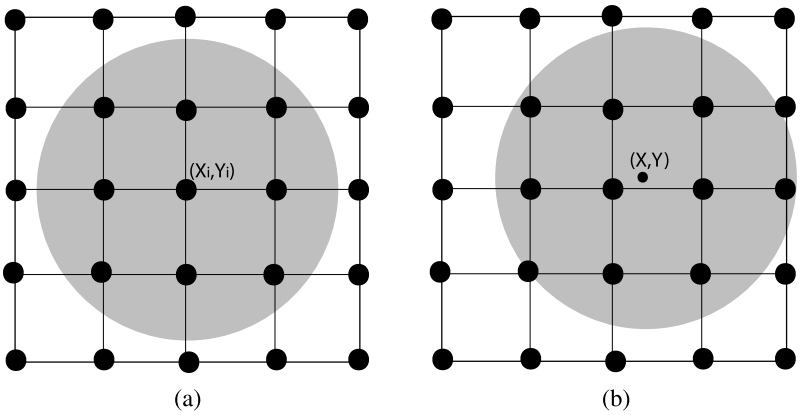


Fig. 10.6 (**a**) A radial function with local support of radius 1.8 pixels centered at pixel (X_i, Y_i). This function shows the influence of pixel (X_i, Y_i) on the interpolating function within the *shaded area*. (**b**) The intensity at (X, Y) is estimated from the intensities of pixels whose distances to (X, Y) are less than a. The pixels used in the calculation are those falling in the *shaded area* in this figure when $a = 1.8$ pixels

10.1.5 Compactly Supported Radial Functions

Cubic spline, cubic convolution, and bilinear interpolation are not radially symmetric functions. In general, locally defined separable functions are not radially symmetric because they approximate rectangular domains. The effect of such functions on an image depends on the orientation of the image. For a resampling operation to be rotationally invariant, the function used in resampling should be radially symmetric. Consider centering the compactly supported radial function,

$$f(r_i) = \begin{cases} 1 - 3r_i^2 + 2r_i^3, & 0 \le r_i \le 1, \\ 0, & r_i > 1, \end{cases} \tag{10.19}$$

at pixel (X_i, Y_i), where

$$r_i = \frac{\sqrt{(X - X_i)^2 + (Y - Y_i)^2}}{a} \tag{10.20}$$

and a is the radius of the domain where the function is nonzero (Fig. 10.6a). Function $f(r_i)$ will then show the influence of the intensity at (X_i, Y_i) on intensity at (X, Y). Note that since $df(r_i)/dr_i = 0$ at $r_i = 1$, not only does $f(r_i)$ vanish at $r_i = 1$, its derivative vanishes at $r_i = 1$. Therefore, if radial function f is centered at each pixel and a weighted sum of the functions is computed, the obtained surface will be smooth everywhere in the image domain.

Assuming intensity at (X_i, Y_i) in the sensed image is I_i, for the surface to interpolate the intensities in a circular neighborhood of radius a centered at point of

Table 10.1 Computational complexities of nearest-neighbor, bilinear interpolation, cubic convolution, cubic spline, and radial functions with local support in image resampling. It is assumed that the sensed image is resampled to a reference image of size n pixels

Type of resampling	Computational complexity
Nearest-neighbor	$O(n)$
Bilinear interpolation	$O(n)$
Cubic convolution	$O(n)$
Cubic spline, direct computation	$O(n^2)$
Cubic spline, using FFT	$O(n \log n)$
Compactly supported radial functions	$O(nm^2)$

interest (X, Y) (Fig. 10.6b), new intensities I_i' should be computed at the pixels within the neighborhood so that when

$$I(X, Y) = \sum_{r_i < a} I_i' f(r_i), \tag{10.21}$$

is evaluated at the pixels within that neighborhood produce the original intensities, that is $I(X_i, Y_i) = I_i$ for all pixels (X_i, Y_i) within the circular neighborhood of radius a centered at (X, Y). Knowing I_i' at the pixels, the intensity at (X, Y) can then be determined from (10.21).

If there are m pixels within a circular neighborhood of radius a pixels, determination of new intensities at the pixels within the neighborhood requires on the order of m^2 multiplications. Since this should be repeated for each pixel, if reference image contains n pixels, resampling the sensed image to register with the reference image requires on the order of nm^2 multiplications.

10.1.6 Summary

Nearest-neighbor, bilinear interpolation, cubic convolution, cubic spline, and compactly supported radial functions in image resampling were discussed. The computational complexities of these resampling methods are summarized in Table 10.1. Nearest-neighbor is the fastest while compactly supported radial functions is the slowest.

Nearest-neighbor resampling does not change the image intensities; it preserves them. However, it can cause aliasing. Bilinear interpolation reduces the aliasing effect by slightly smoothing image intensities. Among all resampling methods, bilinear interpolation is perhaps the best compromise between speed and accuracy. Cubic convolution requires several times more time than bilinear interpolation, but it produces less aliasing. Cubic spline is considerably slower than cubic convolution, especially when the control points of the spline are determined by solving a system of equations. However, resampling accuracy of cubic spline is found to be better than that of cubic convolution [8].

Also discussed was use of radially symmetric kernels in image resampling. Considering the fact that the neighborhood size used in radial functions is adjustable,

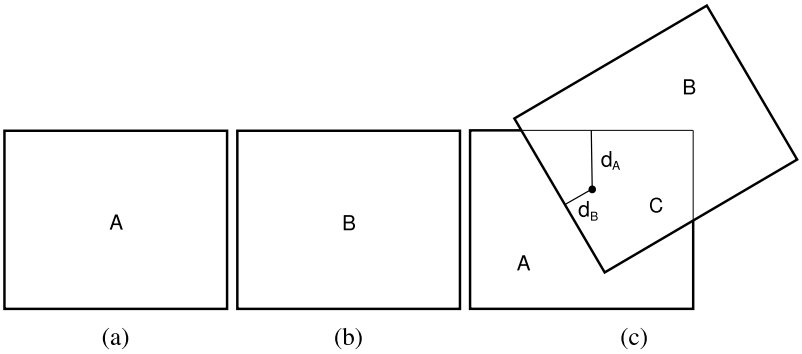


Fig. 10.7 (**a**), (**b**) Reference image A and sensed image B. (**c**) Registration and intensity blending of images A and B using inverse-distance weighting. C is the area of overlap between images A and B

this resampling method can be adapted to the level of details in an image. If cubic spline is used, since a fixed neighborhood size is used independent of the spatial frequency contents in an image, the matrix of coefficients may become singular for some images, making image resampling impossible. In resampling by compactly supported radial functions since the equations to be solved are for the determination of local intensities, the systems are relatively small and the likelihood of becoming singular reduces significantly. However, since a system of equations need be solved for resampling the intensity at each pixel, the process is the slowest among the resampling methods discussed.

10.2 Image Compositing and Mosaicking

After registering two images, there often is a need to combine the images into a larger image called a *composite* or a *mosaic*. Intensities at corresponding pixels in the overlap area of registered images may be different due to differences in environmental and sensor parameters during image acquisition. Different-view images of a scene record different intensities at the same scene point if the scene has specular characteristics. Change in scene brightness due to difference in time of day or weather conditions results in intensity differences between corresponding points in images.

The problem to be solved is as follows: Given reference image A and sensed image B, by registering B to A, we would like to create a mosaic that covers scene areas contained in both A and B as depicted in Fig. 10.7c. Denoting the area of overlap between the images by C, we would like to find intensities in C in such a way that they vary smoothly from region C to both regions A and B, creating a seamless mosaic.

10.2.1 Intensity blending

Intensities in region A in Fig. 10.7c are those of the reference image, intensities in region B are those of the sensed image, and intensities in region C are computed from a weighted sum of intensities of corresponding pixels in the reference and sensed images. To make sure that a seamless mosaic is obtained, the weights at a pixel in region C are set inversely proportional to the distances of that pixel to pixels closest to it in regions A and B. The closer the pixel in C is to region A, the higher the contribution of the pixel in the reference image will be to the computed intensity. If the distance of a pixel in region C to the pixel closest to it in region A is d_A and its distance to the pixel closest to it in region B is d_B, the intensity at the pixel in C is estimated from

$$I^C = \frac{I^A d_B^{-1} + I^B d_A^{-1}}{d_A^{-1} + d_B^{-1}}, \tag{10.22}$$

where I^A and I^B are intensities of corresponding pixels in the reference and sensed images in the overlap area, and I^C is the estimated intensity there. Distances d_A and d_B can be calculated ahead of time efficiently by finding the distance transform [2, 4, 9] of the mosaic image using the border pixels in each image as the object pixels and the remaining pixels as the background pixels.

Intensity blending by weighted averaging has been referred to as *feathering* [10]. Intensity blending in this manner can cause ghosting effects if images in the overlap area have differences due to occlusion or other factors. One way to reduce the ghosting effect is to use a higher power of distances. This will assign the intensities in most areas of C to either those of image A or image B, reducing the areas in C affected by ghosting. If too high a power is used, however, transition of intensities from one image to another becomes rather sharp and visible when the images have large intensity differences.

An alternative method for overcoming ghosting is to find a function that maps intensities of the sensed image to those of the reference image so that, rather than blending intensities of both images in the overlap area, intensities of only the reference image are used everywhere. The process assigns intensities of the reference image to region C, and since intensities in region A are already those of the reference image and intensities of region B are converted to those of the reference image, the composite image will have only intensities of the reference image. The process, in effect, by not blending intensities in region C, avoids the ghosting effect. This is explained in more detail next.

10.2.2 Intensity Mapping

Sometimes intensities in the images to be registered represent different scene properties, and averaging the properties is not meaningful. Consider the reference and sensed images shown in Figs. 10.8a and 10.8b. They represent aerial images of a city

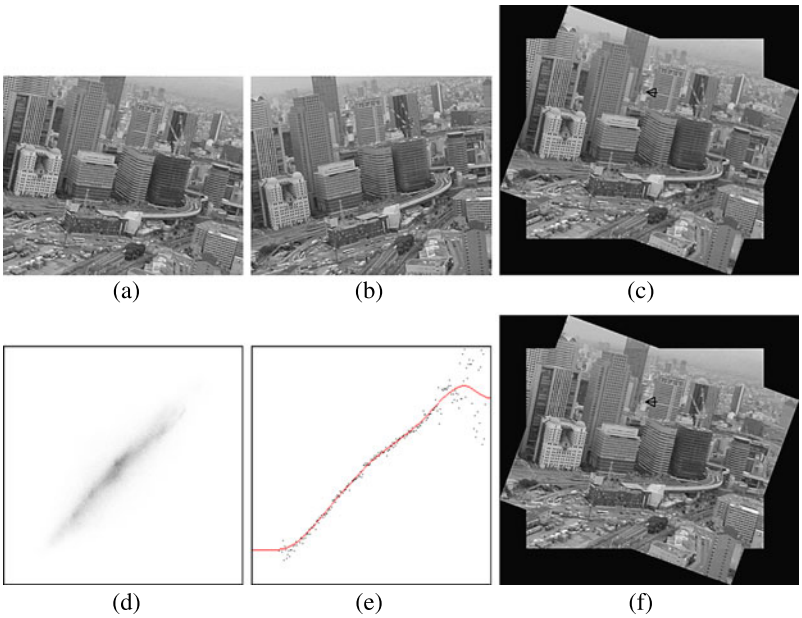


Fig. 10.8 (**a**), (**b**) Aerial images of a city downtown in different modalities. (**c**) Registration of the images and blending of image intensities by inverse-distance weighting. (**d**) Joint probability density (JPD) of intensities in sensed and reference images. (**e**) Intensity map of the sensed image with respect to the reference image. A *black dot* identifies the row with the highest value in each column of JPD. The curve shows approximation of the *black dots* by a rational Gaussian (RaG) curve. The curve maps intensities in the sensed to intensities in the reference images. (**f**) Compositing images (**a**) and (**b**) after registering them and mapping the intensities of the sensed image to those of the reference image (without intensity blending)

downtown in different modalities. While 10.8a is obtained by a sensor with spectral frequencies centered at red, the image in Fig. 10.8b is obtained by a sensor with frequencies centered at green. Registering the images and blending the intensities by the inverse-distance weighting method produces the composite image shown in Fig. 10.8c.

Although the images are seamlessly combined, the obtained intensities are no longer meaningful. We would like to map intensities of the sensed image to those of the reference image so that the combined image shows intensities of only the reference image. To achieve this, a mapping function that relates intensities of the sensed image to those of the reference images is required.

To find this mapping function, first, the joint probability density (JPD) of intensities in the images is calculated. A matrix is created with its entry (i, j) showing the number of pixels in the overlap area in the images where the intensity in the reference image is i and the intensity in the sensed image is j. Therefore, the columns in this matrix index to the intensities in the sensed image, while the rows index to the intensities in the reference image. The entries of the matrix are divided by the number of pixels in the overlap area to obtain probabilities.

The JPD of intensities in images in Figs. 10.8a and 10.8b is shown in Fig. 10.8d. Higher probabilities are shown darker in this image. This JPD shows that an intensity in the sensed image maps to a number of relatively close intensities in the reference image. This occurs due to noise and resampling side effects. When bilinear interpolation is used to resample the sensed image to register the reference image, most often intensities of four adjacent pixels in the sensed image are used to calculate the intensity of a pixel in the resampled image. Even when intensities in the sensed image uniquely map to intensities in the reference image, the resampling process destroys this uniqueness and makes the mapping ambiguous. Noise and other factors worsen the situation.

To obtain a unique mapping, at each column j in the JPD, the row i with the highest probability is identified. i will be the intensity in the reference image most frequently corresponding to intensity j in the sensed image. The JPD entries identified in this manner are shown by black dots in Fig. 10.8e. The black dots provide a unique mapping from intensities in the sensed image to those in the reference image, because each intensity j in the sensed image has a unique value i in the reference image. Such a mapping, however, is very noisy as can be observed in Fig. 10.8e. Adjacent intensities in the sensed image do not map to adjacent intensities in the reference image.

To reduce the effect of noise and map adjacent intensities in the sensed image to adjacent intensities in the reference image, a rational Gaussian (RaG) curve [5] is fitted to the points as shown in Fig. 10.8e. The curve provides a unique mapping from intensities in the sensed image to intensities in the reference image.

The image composite obtained after converting intensities of the sensed image to those of the reference image in this manner is shown in Fig. 10.8f. Intensities across the composite image measure the same scene properties. Although the images in Figs. 10.8c and 10.8f appear similar, looking more closely, we see that intensities in 10.8f follow the intensities in 10.8a, while those in 10.8c are a blending of the intensities in 10.8a and 10.8b. For instance, this difference is seen in Figs. 10.8c and 10.8f in the areas pointed to by the arrows.

References

1. Cheng, F., Goshtasby, A.: A parallel B-spline surface fitting algorithm. ACM Trans. Graph. **8**(1), 41–50 (1989)
2. Cuisenaire, O., Macq, B.: Fast Euclidean distance transformation by propagation using multiple neighborhoods. Comput. Vis. Image Underst. **76**(2), 163–172 (1999)
3. Ferrari, L.A., Sankar, P.V., Sklansky, J., Leeman, S.: Efficient two-dimensional filters using B-spline functions. Comput. Vis. Graph. Image Process. **35**, 152–169 (1986)
4. Goshtasby, A.: 2-D and 3-D Image Registration for Medical, Remote Sensing, and Industrial Applications. Wiley, New York (2005)
5. 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)
6. Goshtasby, A., Cheng, F., Barksy, B.A.: B-spline curves and surfaces viewed as digital filters. Comput. Vis. Graph. Image Process. **52**, 264–275 (1990)

7. Hou, H.S., Andrews, H.C.: Cubic splines for image interpolation and digital filtering. IEEE Trans. Acoust. Speech Signal Process. **26**(6), 508–517 (1978)
8. Keys, E.G.: Cubic convolution interpolation for digital image processing. IEEE Trans. Acoust. Speech Signal Process. **29**(6), 1153–1160 (1981)
9. Shih, F.Y., Wu, Y.-T.: Fast Euclidean distance transformation in two scans using a 3×3 neighborhood. Comput. Vis. Image Underst. **93**, 195–205 (2004)
10. Szeliski, R.: Computer Vision: Algorithms and Applications. Springer, London (2011)