C H A P T E R 13

Mosaics: Using Translations to Register Overlapping Images

Generalizing the procedure of Section 3.4.1 is really useful. Back in the days when photographs were printed on paper by special stores, one way to make a photograph of a large object was to take several different, overlapping pictures; print them; place one printed image down on a corkboard; then slide the other printed pictures around on a corkboard until the overlapping sections of each pair of pictures show the same things; and then pin them down. The result is a *mosaic* – a collection of pictures that have been registered (Figure 13.2). Two pictures are *registered* when the overlapping sections of picture show the same things as much as possible. There are a number of reasons to build mosaics. You might simply not have the right camera, and so have to assemble a big picture out of small ones. You might be able to improve resolution in the overlapping portions, because there you have multiple estimates of pixel values. You might be able to identify moving objects or important changes by comparing registered images.

13.1 REGISTERING IMAGES BY TRANSLATION

For the moment, assume we have two images $\mathcal{A}$ (which is $a_M \times a_N$) and $\mathcal{B}$ (which is $b_M \times b_N$). These two images are overlapping views of a scene that can be aligned exactly by a translation. The fact that they can be aligned means that there is some t_x, t_y so that $\mathcal{A}_{ij} = \mathcal{B}_{i-t_x,j-t_y}$ for those pixels where the images overlap. These are locations such that $1 \leq i \leq a_M$, $1 \leq j \leq a_N$, $1 \leq j - t_x \leq b_M$ and $1 \leq j - t_y \leq b_N$. Visualize this as placing $\mathcal{B}$ on top of $\mathcal{A}$, then sliding $\mathcal{B}$ by t_x, t_y; then the parts of $\mathcal{A}$ and $\mathcal{B}$ that overlap look the same. We are given $\mathcal{A}$ and $\mathcal{B}$ and must find t_x, t_y. For the moment, assume that t_x, t_y are integers.

13.1.1 Building Mosaics by Search

Registering $\mathcal{A}$ and $\mathcal{B}$ follows the recipe of Section 3.4.1: choose a function that is smallest when $\mathcal{A}$ is registered with $\mathcal{B}$; now search for the best value of the cost function.

If the two images really do agree on the overlap, then the SSD is a good choice of cost function. Find the minimum by computing the value of the SSD for each translation that results in an overlap, then taking the smallest. If the range of translations is large, this procedure could be inefficient (improvements in Section 13.1.2), but it is a reasonable model for the moment. Notice that there is no point in translating both images (exercises). Choosing one image to stay in a fixed position is equivalent to choosing the coordinate system for the mosaic. The other image is then translated to the right position in that coordinate system.

One more step is required to go from two registered images to a mosaic. For

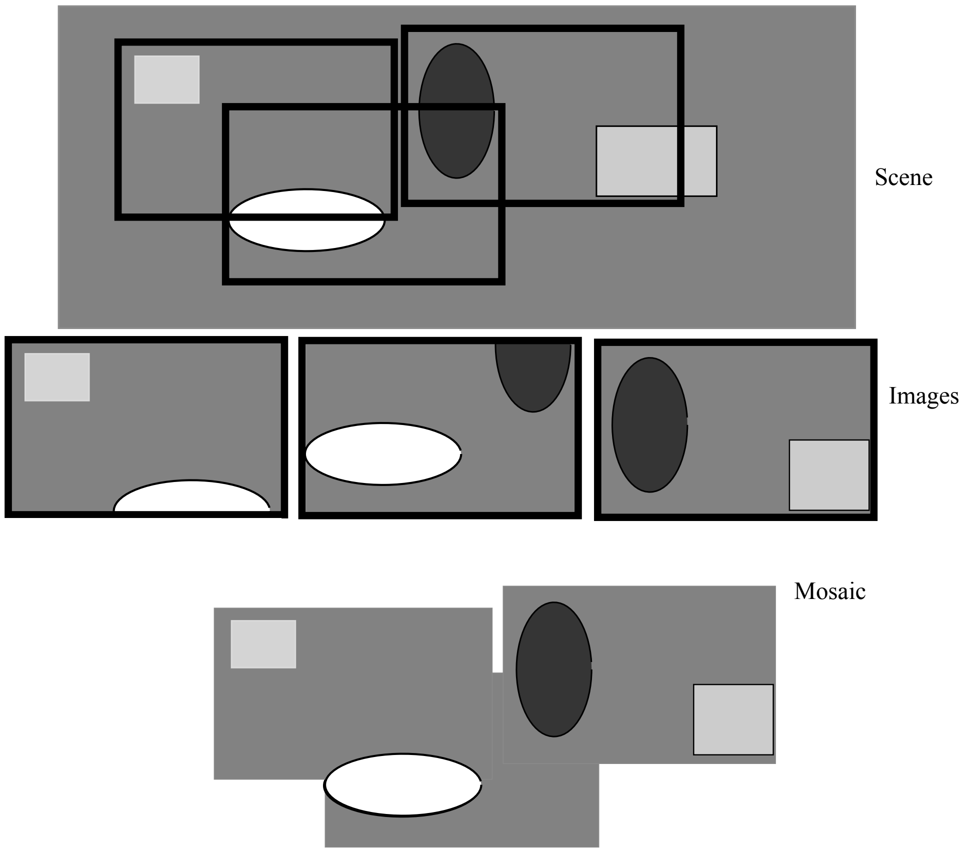

FIGURE 13.1: **Top** *shows a set of overhead images of a simple scene. The dark boundaries show each of three image frames.* **Center** *shows the actual images obtained in these frames.* **Bottom** *shows the mosaic that can be recovered by sliding images with respect to one another. This mosaic can't show features that haven't been imaged, but does show the relative configuration of scene components.*

pixels in the overlapping region, there are two estimates of the pixel value (one from $\mathcal{A}$ and one from $\mathcal{B}$). These need to be combined in some way. Not much can be done with two values, but a mosaic will be generally be constructed out of N different images, so there might be as many as N different estimates at a given location. It turns out that there are a variety of interesting possibilities here (Section **??**).

Here is a simple procedure to build a mosaic from a set of N images $\{\mathcal{I}_1, \ldots, \mathcal{I}_N\}$, sketched in Figure 13.2.

- **Start** by attaching an *offset* of zero to each image. This serves as the translation of that image with respect to the mosaic coordinate system. Choose one image to serve as the *root image* – the image that is never translated.

- Now iterate the following step until there are no images left:

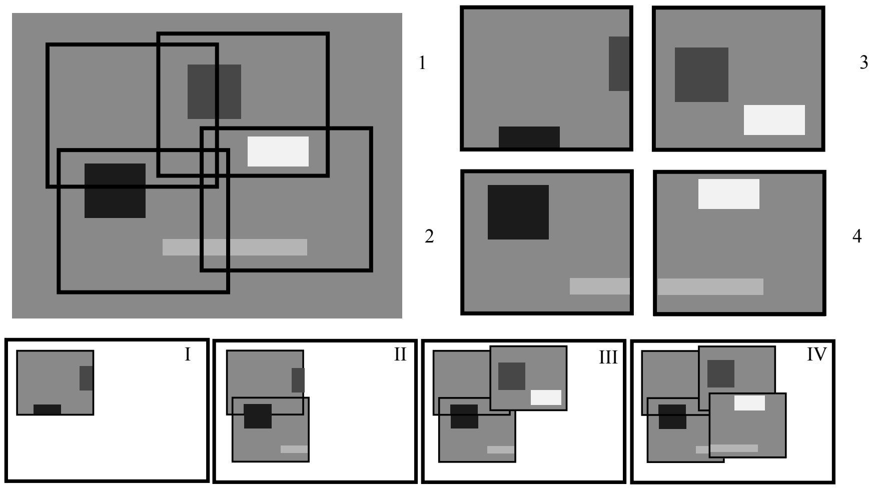

FIGURE 13.2: **Top left** *shows a simple scene with the location of four images; these are shown on the* **top right**. *The steps of creating a mosaic are sketched in the Roman numeral panes on the* **bottom**. *In step I, $\mathcal{I}_1$ is placed in the mosaic; in step II, $\mathcal{I}_2$ which is registered to $\mathcal{I}_1$. This proceeds until there aren't any images that overlap (step III, $\mathcal{I}_3$ to $\mathcal{I}_2$; step IV, $\mathcal{I}_4$ to $\mathcal{I}_3$). Finally, the overlapping image information is turned into pixel values.*

  - **Place another image** by finding an image $\mathcal{I}_k$ which overlaps with at least one image in the mosaic and register it to that image, then update the mosaic using its pixels.

- **Summarize** the overlapping images by computing pixel values at each location in the mosaic using all the information in each vector at each location.

The choice of root image may have consequences, and the choice of which image to place and which image it overlaps will certainly have consequences. For the moment, assume that these choices have been made well. In some applications, the image with a good overlap will be obvious. For example, if you build a mosaic out of overhead aerial images, the images are going to be timestamped, and the next image will overlap rather well with the current image.

13.1.2 Coarse to Fine Search

The obvious strategy for minimizing the cost function is: apply each translation; compute the objective function; then take the translation with smallest value. This will be slow if the range of translations is large. Notice that if we smoothed and subsampled $\mathcal{A}$ and $\mathcal{B}$ to produce a very coarse scale version (say, 8×8) of each image, we could compute a coarse estimate of m, n from those fairly easily because there are very few values to search. A coarse estimate of the translation that registers two coarse scale images can be refined using larger versions of the images. This observation leads to a really powerful search strategy: register very small versions of the images, then repeatedly refine the registration estimate using increasingly

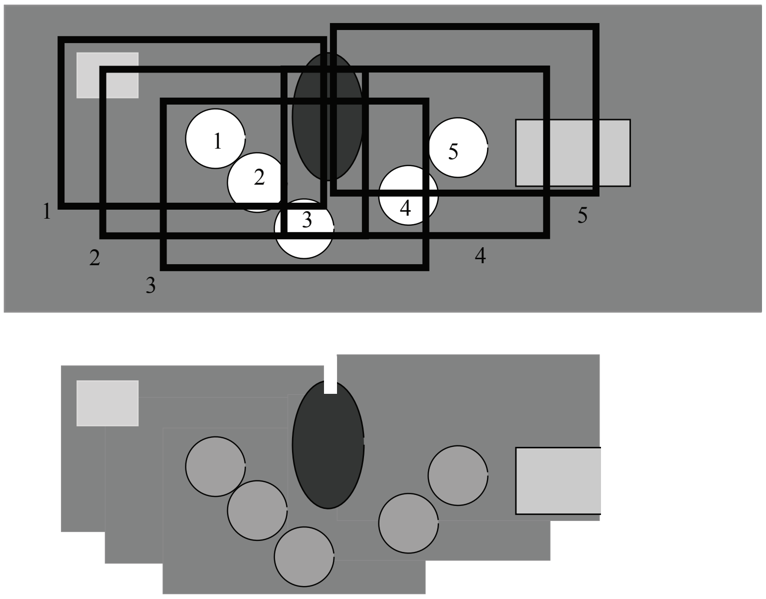

FIGURE 13.3: *Summarizing the mosaic with a mean shows a ghostly path of moving objects.* **Top** *shows a set of images of a simple scene with a moving object (the white circle). This is at location 1 in frame 1, and so on (location number on the object, frame number next to the frame). The background does not move. The frames are registered to one another to produce a mosaic.* **Bottom** *shows what happens if one averages. The circles affect the average, and appear – one can both background and object. Compare with Figure 13.4 and Figure 13.5.*

large versions until the full size images are registered. This strategy is known as *coarse to fine search*.

For example, look at the zebra's muzzle in Figure 2.11, and think about registering this image to itself. The 8×8 version has very few pixels, and looks like a medium dark bar, darker at the muzzle end. Finding a translation to register this image to itself should be fairly straightforward, and unambiguous. Assume we find m_8, n_8. In the 16×16 version, some stripes are visible. Registering this image to itself might be more difficult, because the stripes will create local minima of the cost function (check you follow this remark; think about what happens if you have the images registered, and then shift the muzzle perpendicular to the stripes). But if we have an estimate of the translation from the 8×8 version, we do not need to search a large range of translations to register the 16×16 version. We need to look only at four translations: $2 * m_8, 2 * n_8$; $2 * m_8 + 1, 2 * n_8$; $2 * m_8, 2 * n_8 + 1$; and $2 * m_8 + 1, 2 * n_8 + 1$. The same reasoning applies when going from the 16×16 version to the 32×32 version, and so on. It should be obvious that this is a natural application of a gaussian pyramid (Section **??**). Notice that the recipe applies even if one subsamples by less than two, though some details change (exercises).

Coarse-to-fine search suggests a way to find pairs of images with a good over-

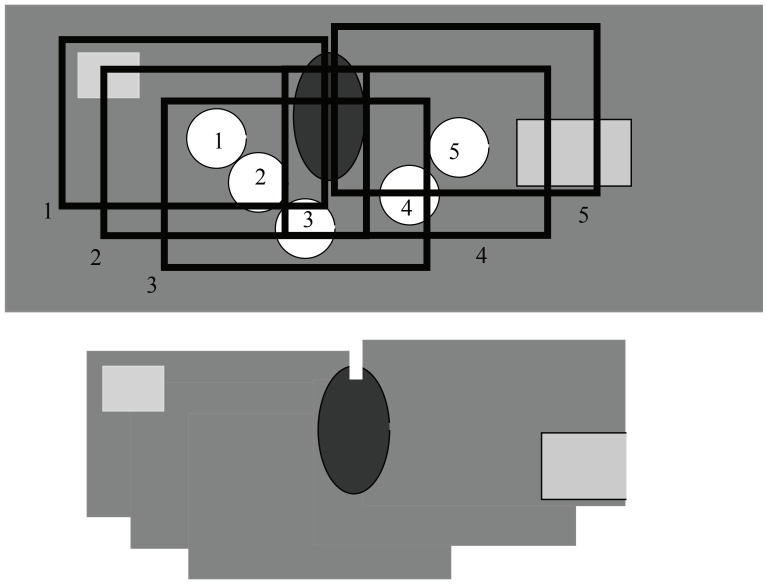

FIGURE 13.4: *Summarizing the mosaic with a median suppresses moving objects.* **Top** *shows a set of images of a simple scene with a moving object (the white circle). This is at location 1 in frame 1, and so on (location number on the object, frame number next to the frame). The background does not move. The frames are registered to one another to produce a mosaic.* **Bottom** *shows what happens if one summarizes using a median. The moving object spends little time at each pixel, and so does not change the median – the result shows the background. Compare with Figure 13.3 and Figure 13.5.*

lap: try to register coarse scale versions of the images with one another. Not all pairs will register, but those that do with a small enough translation will likely have a good overlap, and you can use this to obtain an estimate of the extent to which images overlap.

13.1.3 Summarizing Registered Images to form a Mosaic

There are a number of useful ways to summarize the registered images. At many locations, the recipe will produce a vector containing many components that are not unknown. Each is an estimate of the "true" pixel value for that location, but there is some error in this estimate. For example, if a small object is moving in the scene, it will affect some pixels in each frame. The main options to use as a summary of this vector are:

- The **mean**, best if there is no moving object, and one expects that the registration is very good. Averaging will tend to blur details if the images aren't precisely registered, and will show a blurry trail of moving objects, as Figure 13.3 shows and Figure 13.7 confirms.

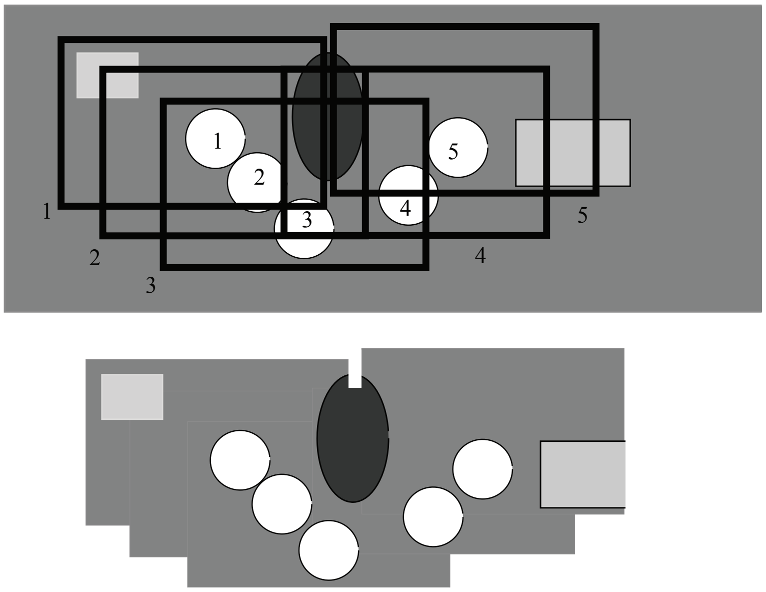

FIGURE 13.5: *Summarizing the mosaic with the value furthest from the median shows the path of moving objects.* **Top** *shows a set of images of a simple scene with a moving object (the white circle). This is at location 1 in frame 1, and so on (location number on the object, frame number next to the frame). The background does not move. The frames are registered to one another to produce a mosaic.* **Bottom** *shows what happens if one summarizes using the value most different from the median. The moving object spends little time at each pixel, and so is likely very different from the median – the result shows the moving object in different locations against the background. Compare with Figure 13.3 and Figure 13.4.*

- The **median**, best if there are moving objects and you want to suppress them; the median will tend to show the background (because more of the values from the registered images will be background), and does not blur as much in the face of misregistration, as Figure 13.4 shows and Figure 13.8 confirms.

- A **preferred image**, chosen in some way. Details will mostly not be blurred, and moving objects will largely be frozen in place (but might appear in more than one place). The outer edges of images can be prominent, but this can be suppressed by blending (Figures **??** and 13.10) Figure 13.3 shows and Figure 13.7 confirms.

- The **value most different from the median**, if you want to show object motion. This will tend to accentuate details, and show a moving object in its different locations (Figures 13.5 and 13.9).

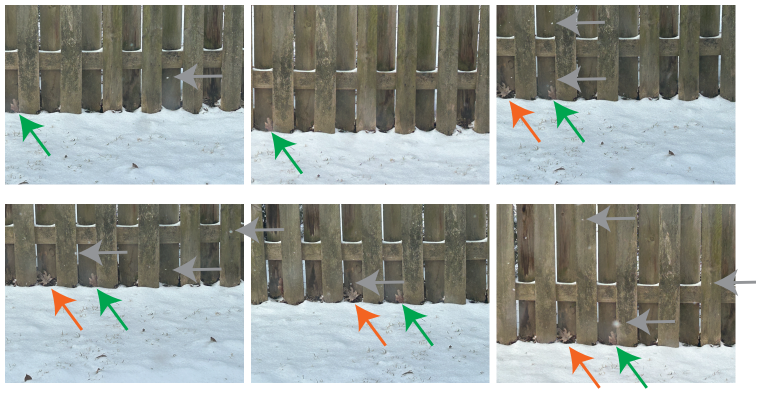

FIGURE 13.6: *Six images of a snowy yard I used to make the mosaics of Figures 13.7, 13.8, 13.9 and 13.10.* **Colored arrows** *point to distinctive leaves at the base of the fence posts, which you can use to register the images by eye (the colors correspond).* **Gray arrows** *point to some drifting snowflakes.* Image credit: *Figure shows my photographs of a snowy yard.*

13.2 IMPROVEMENTS AND APPLICATIONS

13.2.1 Sub-Pixel Estimates and Interpolation

The translation that registers images can be estimated at a resolution that is finer than a single pixel. You should expect this – each image has very large numbers of pixels that are being used to estimate the translation. The main question is how to get estimates of the translation at a fine resolution. One straightforward procedure is to upsample the images, using an interpolate, and register the results. The coarse-to-fine search procedure still applies. For example, you could add layers to the gaussian pyramid by simply upsampling the images to appropriate sizes, then proceed as before.

An alternative strategy is to interpolate the cost function. Recall the original translations were estimated by evaluating the cost function at a set of translations, then finding the translation with the best cost. Now interpolate the cost function, and obtain the minimum of the interpolate. This works only if the interpolate has a minimum that isn't on a grid point, so at least the bicubic interpolate of Exercise **** is required. A minor issue results. If the translation isn't an integer, then the sampled values of one of two registered images aren't on grid points. This is easily dealt with by interpolation.

13.2.2 Bundle Adjustment

Our simple procedure does not produce the best possible mosaic. To see this, assume you have images $\mathcal{I}_1, \ldots, \mathcal{I}_4$, as in Figure 13.11, and you introduce them into the mosaic in that order. Write $T_{2\rightarrow 1}$ for the translation to align $\mathcal{I}_2$ with $\mathcal{I}_1$ by translating $\mathcal{I}_2$ to the right place in the mosaic coordinate system. The effect of this translation is shown in step II in Figure 13.11. Now you estimate $T_{3\rightarrow 1}$ to

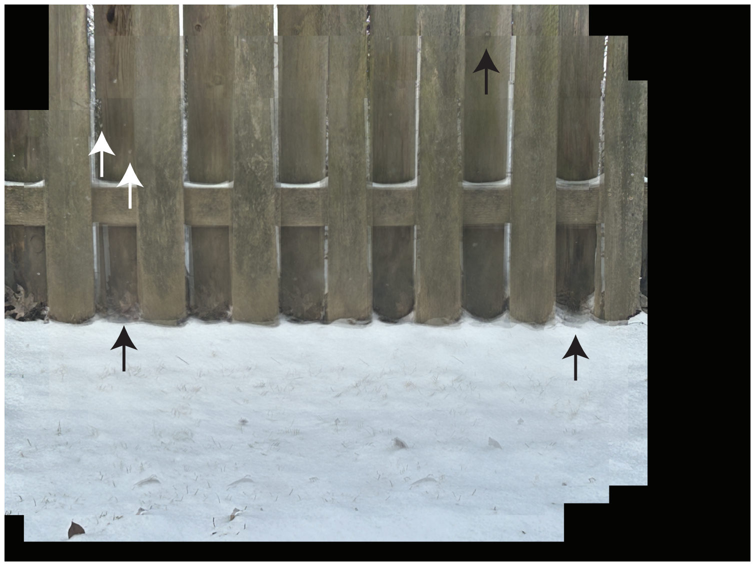

FIGURE 13.7: *Summarizing with a mean will reduce the effects of snowflakes, but can blur detail.* **Center** *shows a mosaic made by registering the six images of Figure* **??** *around it using translations. The registered images are then combined using an average. The* **dark arrows** *point to: blurring and loss of detail caused by registration errors followed by averaging (the leaf on the* **left** *is nearly gone; the base of the fence on the* **right** *is seriously blurred); and outer edges of the image visible because the images are not blended into the average. The* **light arrows** *point to snow effects. In the average, these snowflakes tend to have low contrast.* Image credit: *Figure shows my photographs of a snowy yard.*

register $\mathcal{I}_3$ to $\mathcal{I}_1$ (Step III). Notice that the patterns in the scene have been chosen to show an exaggerated case where the registration error between $\mathcal{I}_3$ and $\mathcal{I}_1$ is likely to be large (many different horizontal translations of $\mathcal{I}_3$ will register with $\mathcal{I}_1$ well). Finally, you estimate $\mathcal{T}_{4\to 3}$ (Step IV). Notice how error has cascaded. $\mathcal{I}_3$ is poorly registered to $\mathcal{I}_1$, and $\mathcal{I}_4$ is registered to $\mathcal{I}_3$, meaning that $\mathcal{I}_4$ is poorly registered to $\mathcal{I}_1$.

Notice that this isn't necessary. Some pixels of $\mathcal{I}_4$ overlap $\mathcal{I}_2$. If these pixels contributed to the registration, $\mathcal{I}_4$ and $\mathcal{I}_3$ could be properly registered. This problem occurs quite generally – it is not just a result of an odd scene – and is often referred to as a failure of *loop closure*. This refers to the idea that if 2 is registered to 1, 3 is registered to 2, 4 is registered to 3, all the way up to N is registered to $N-1$, N may not be registered to 1 at all well – the loop does not close.

There are several procedures to prevent or control this kind of error propagation. Start by registering the images by pairs as described. The easiest strategy is now to repeat: fix all but one image, then register that image to all the others it overlaps with. This approach can work, but may be slow, because it may take a long time for improvements to propagate across all the images. The alternative, which

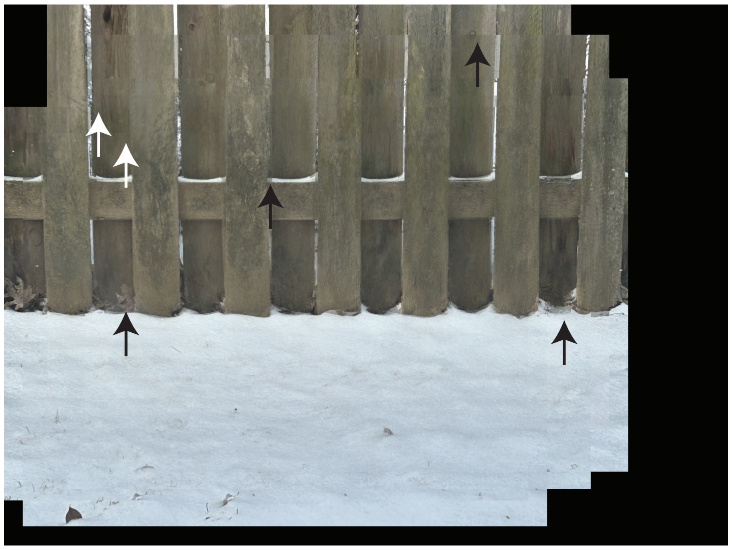

FIGURE 13.8: *Summarizing with a median will suppress snowflakes with less loss of detail, but can lead to contours breaking up.* **Center** *shows a mosaic made by registering the six images of Figure* **??** *around it using translations. The registered images are then combined using a median. The* **dark arrows** *point to: reduced blurring (the leaf on the* **left** *is now better than in Figure 13.7, as is the base of the fence on the* **right***); but outer edges of the image remain visible because the images are not blended. Notice that some contours behave strangely (***center left** *dark arrow). The* **light arrows** *point to snow effects. Snowflakes were falling when these pictures were taken. In the median, these snowflakes have largely disappeared.* Image credit: *Figure shows my photographs of a snowy yard.*

is better but can be onerous, is to fix one image in place, then adjust all others to register in every overlap. This isn't a straightforward optimization problem. There is no need to resolve it in detail yet (but see Section 41.2 if you're concerned).

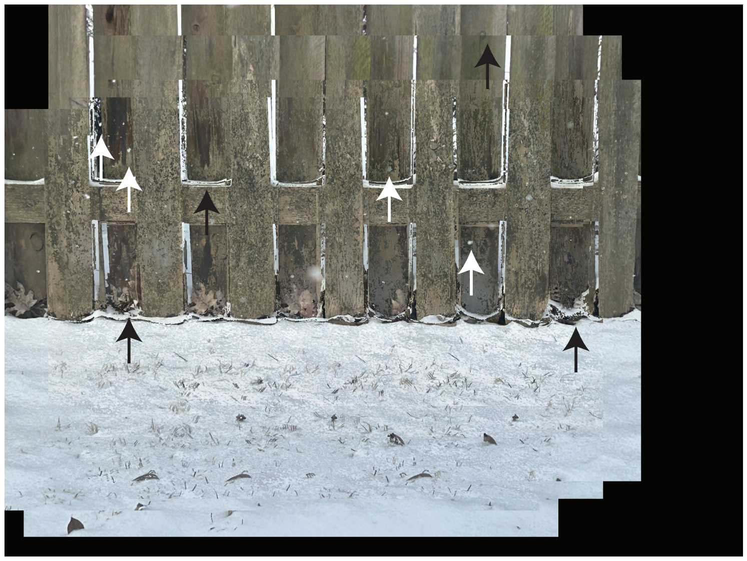

FIGURE 13.9: *Summarizing with the pixel most different from the median will show the snowflakes, but can lead to contours breaking up.* **Center** *shows a mosaic made by registering the six images of Figure* **??** *around it using translations. The registered images are then combined by choosing the pixel value most different from the median at each location. Notice the general increase in high resolution detail. The* **dark arrows** *point to: reduced blurring at some cost (there are now multiple instances of the leaf on the* **left** *for example, and the base of the fence on the* **right** *is quite strange); but outer edges of the image remain visible because the images are not blended. Notice that some contours are jittered because of misregistration (***center left** *dark arrow). The* **light arrows** *point to snow effects. Snowflakes were falling when these pictures were taken. Choosing pixels that are different from the median tends to bring these snowflakes out.* Image credit: *Figure shows my photographs of a snowy yard.*

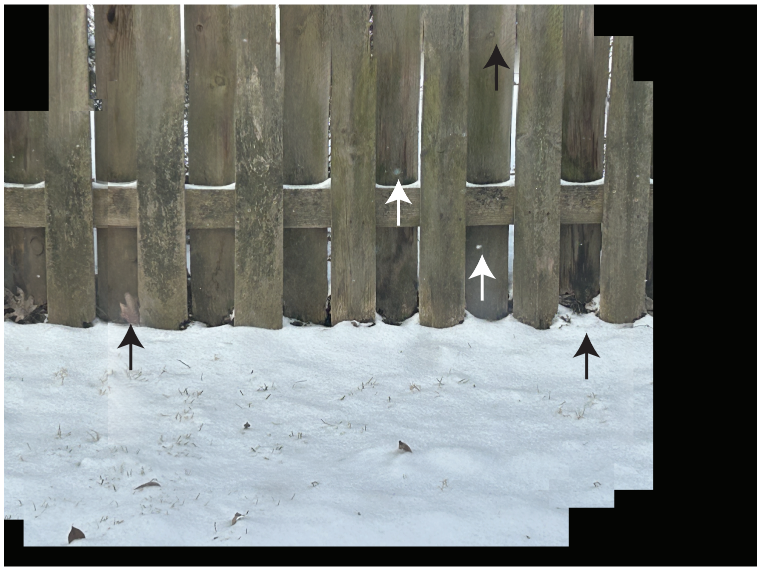

FIGURE 13.10: *Summarizing by choosing which image to show will suppress blur and contour effects, but won't suppress moving objects.* **Center** *shows a mosaic made by registering the six images of Figure* **??** *around it using translations. The registered images are then combined by choosing the first image that has a pixel value at that location; if there is more than one image, and the pixel is close to a boundary, the first and second images are blended. Notice the general increase in high resolution detail because most pixels come from a specific image. The* **dark arrows** *point to: reduced blurring (the leaf on the* **left** *is now better than in Figure 13.7, as is the base of the fence on the* **right***); but outer edges of the image remain visible because the images are not blended. The* **light arrows** *point to snow effects. Snowflakes were falling when these pictures were taken, so the mosaic shows whatever snowflakes were in the image chosen.* Image credit: *Figure shows my photographs of a snowy yard.*

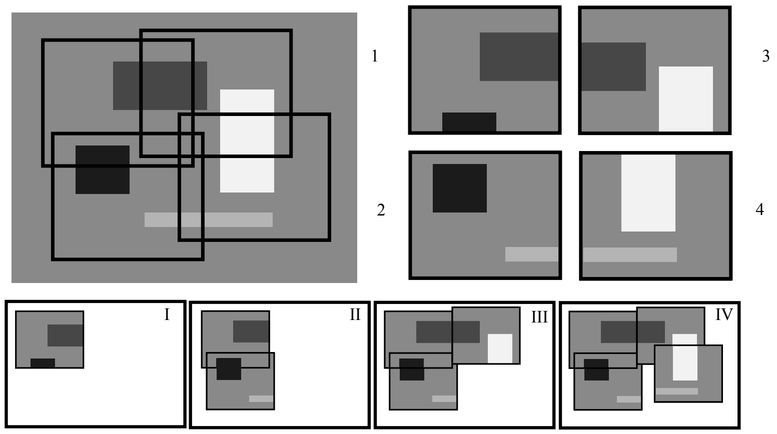

FIGURE 13.11: **Top left** *shows a simple scene with the location of four images; these are shown on the* **top right**. *The steps of creating a mosaic are sketched in the Roman numeral panes on the* **bottom**. *In this case, the translation of the fourth image with respect to the first is poorly estimated; more detail in the text.*