consecutive rotations by 360/N degrees will not, in general, equal the original image. In cases of multiple rotations you should consider keeping a copy of the original image as the same source for all rotations.

Image Registration

In many situations one has two or more images of the same object where the differences between the images have to do with acquisition times, dissimilar acquisition hardware or changes in the shape of the object between exposures. To facilitate comparison between such images it is necessary to register them, i.e., to adjust them so that they match each other. The **ImageRegistration** operation (see page V-398) modifies a test image to match a reference image when the key features are not too different. The algorithm is capable of subpixel resolution but it does not handle very large offsets or large rotations. The algorithm is based on an iterative processing that proceeds from coarse to fine detail. The optimization is performed using a modified Levenberg-Marquardt algorithm and results in an affine transformation for the relative rotation and translation with optional isometric scaling and contrast adjustment. The algorithm is most effective with square images where the center of rotation is not far from the center of the image.

ImageRegistration is based on an algorithm described by Thévenaz and Unser.

Mathematical Transforms

This class of transforms includes standard wave assignments, interpolation and sampling, Fourier, Wavelet, Hough, and Hartley transforms, convolution filters, edge detectors and morphological operators.

Standard Wave Operations

Grayscale image waves are regular 2D Igor waves that can be processed using normal Igor wave assignments (**Waveform Arithmetic and Assignments** on page II-74 and **Multidimensional Wave Assignment** on page II-96). For example, you can perform simple linear operations:

```
Duplicate/O root:images:blobs sample
Redimension/S sample       // create a single precision sample
sample=10+5*sample         // linear operation
NewImage sample            // keep this image displayed
```

Note that the display of the image is invariant for this linear operation.

Nonlinear operations are just as easy:

```
sample=sample^3-sample     // not particularly useful
```

You can add noise and change the background using simple wave assignment:

```
sample=root:images:blobs   // rest to original
sample+=gnoise(20)+x+2*y   // add Gaussian noise and background plane
```

As we have shown in several examples above, it is frequently necessary to create a binary image from your data. For instance, if you want an image that is set to 255 for all pixels in the image wave sample that are between the values of 50 and 250, and set to 0 otherwise, you can use the following one line wave assignment:

```
MatrixOp/O sample=255*greater(sample,50)*greater(250,sample)
```

More Efficient Wave Operations

There are several operations in this category that are designed to improve performance of certain image calculations. For example, you can obtain one plane (layer) from a multiplane image using a wave assignment like:

```
Make/N=(512,512) newImagePlane
newImagePlane[][]=root:Images:peppers[p][q][0]
```

Alternatively, you can execute:

```
ImageTransform/P=0 getPlane root:Images:peppers
```

or