Geometric Transformation Interpolation Methods

Overview of Interpolation Methods

The Geometric Transformations library of Video and Image Processing Blockset software contains blocks that perform geometric transformations. These blocks use interpolation to calculate the appropriate pixel values so that images appear rotated, translated, resized, or sheared.

Note  The examples in the following sections are illustrations of interpolation methods. The block algorithms are implemented in a slightly different way so that they are optimized for speed and memory.

Nearest Neighbor Interpolation

For nearest neighbor interpolation, the block uses the value of nearby translated pixel values for the output pixel values.

For example, suppose this matrix,

1  2  3  
4  5  6  
7  8  9  

represents your input image. You want to translate this image 1.7 pixels in the positive horizontal direction using nearest neighbor interpolation. The Translate block's nearest neighbor interpolation algorithm is illustrated by the following steps:

1. Zero pad the input matrix and translate it by 1.7 pixels to the right.

   Translated zero-padded matrix

   0 1 0 2 1 3 2 0 3 0 0
   0 4 0 5 4 6 5 0 6 0 0
   0 7 0 8 7 9 8 0 9 0 0

   Original zero-padded matrix

   1.7 pixels

2. Create the output matrix by replacing each input pixel value with the translated value nearest to it. The result is the following matrix:

   0 0 1 2 3
   0 0 4 5 6
   0 0 7 8 9

Note  You wanted to translate the image by 1.7 pixels, but this method translated the image by 2 pixels. Nearest neighbor interpolation is computationally efficient but not as accurate as bilinear or bicubic interpolation.
### Bilinear Interpolation

For bilinear interpolation, the block uses the weighted average of two translated pixel values for each output pixel value.

For example, suppose this matrix,

```
1 2 3
4 5 6
7 8 9
```

represents your input image. You want to translate this image 0.5 pixel in the positive horizontal direction using bilinear interpolation. The Translate block's bilinear interpolation algorithm is illustrated by the following steps:

1. Zero pad the input matrix and translate it by 0.5 pixel to the right.

   Zero pad the input matrix:

   ![Zero padded matrix](image)

   Output matrix:

   

2. Create the output matrix by replacing each input pixel value with the weighted average of the translated values on either side. The result is the following matrix where the output matrix has one more column than the input matrix:

   

### Bicubic Interpolation

For bicubic interpolation, the block uses the weighted average of four translated pixel values for each output pixel value.

For example, suppose this matrix,

```
1 2 3
4 5 6
7 8 9
```

represents your input image. You want to translate this image 0.5 pixel in the positive horizontal direction using bicubic interpolation. The Translate block's bicubic interpolation algorithm is illustrated by the following steps:

1. Zero pad the input matrix and translate it by 0.5 pixel to the right.

   Zero pad the input matrix:

   ![Zero padded matrix](image)

   Output matrix:

   


2. Create the output matrix by replacing each input pixel value with the weighted average of the two translated values on either side. The result is the following matrix where the output matrix has one more column than the input matrix:

\[
\begin{array}{cccc}
0.375 & 1.5 & 3 & 1.625 \\
1.875 & 4.875 & 6.375 & 3.125 \\
3.375 & 8.25 & 9.75 & 4.625 \\
\end{array}
\]