Image Interpolation, Rotation and Straightening
Computer Vision 2006, Lab Exercise 1
Andreas van Cranenburgh (0440949), Willem Bressers (5687063)
January 11, 2008

Abstract

This exercise discusses implementations of algorithms for interpolation, rotation, shearing and skewing using matrix transformations.

1 Problem specification

...

2 Theory and implementation

2.1 Interpolation

When an image has been transformed then its projection usually lies on a different pixel grid (resolution) from the original. For example, if an image has been scaled upwards (increasing the picture size), the resulting image will have more pixels than the original. To calculate a pixel value of the projection, it is necessary to make an estimate of what this value should be, supposing that there is a continuous function behind the original. There are at least two methods to this.

2.1.1 Nearest neighbor

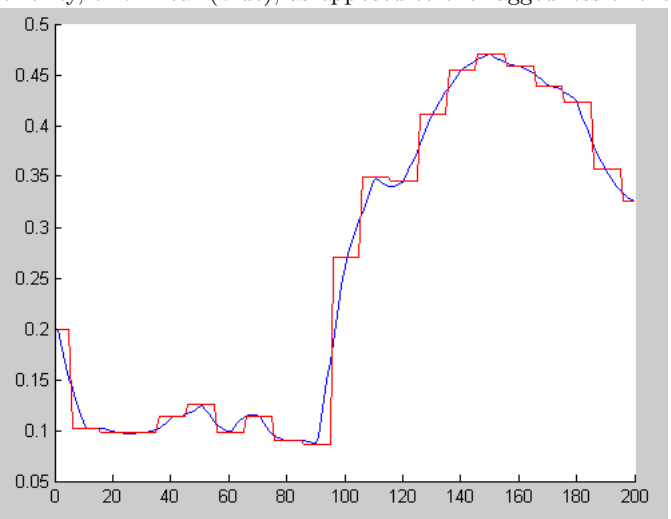
The nearest neighbor interpolation method, looks for the pixel in the original that is closest to the pixel in the projection. This method is very fast and easy to understand, but it assumes that the color value of the closest original pixel is the best. This is tricky especially when we look at "corners" or "gradients"; repeated use increases the loss of information dramatically.

2.1.2 Bilinear interpolation

The bilinear interpolation method, takes the four pixels surrounding point to be interpolated, and returns a weighted-average of them. This method is not so fast as nearest neighbor but it handles much better in "corners" or "gradients."

2.1.3 Nearest neighbor vs bilinear

As a comparison of these methods, we sampled the pixel values on a diagonal (1D) line with both interpolation methods. A plot of the resulting pixel values clearly shows the smoothness, and thus superiority, of bilinear (blue), as opposed to the raggedness of the nearest neighbor plot (red).



2.2 Rotations

This presents another way of measuring the performance of the two interpolation methods. By rotating an image back and forth using both nearest neighbor and bilinear interpolation, it is possible to measure the distance between the resulting images. The distance metric for images we used works by subtracting the first image from the other, and squaring the result. The squaring emphasizes big differences over small differences. Also, since interpolation errors can both increase and decrease the pixel values, squaring makes sure that all values are positive. Finally, the mean value of the whole matrix is the resulting distance. As expected, the bilinear interpolation results in an image closer to the original, at the cost of being slower.

```
>> testRotation
Elapsed time is 5.319305 seconds.

nearestdist = 0.0033

Elapsed time is 6.652623 seconds.

lineardist = 0.0024
```

At first the "testRotation" compared the full images, and for some reason this resulted in the bilinear interpolation having a bigger difference with the original compared to nearest neighbor interpolation. After being puzzled by this form some time the solution turned out to be to frame a sample in the middle of the image, so that the black borders created by the rotation do not interfere with the result.

2.3 Affine transformations

2.4 Projective transformations

There is an even more generalized version of the affine transformations. While affine transformations are based on three points, specifying more points results in a projective transformation. Because the transformation is overdetermined, its solution can only be approximated.

3 Conclusion