Sikender Shahid
0981476
02/04/2019

Introduction

Image resizing is a common tool utilized in many software to enhance a image resolution. There are well known algorithms that vary in performance and result while serving a practical use case. These common algorithm are the following: nearest-neighbor interpolation, bi-linear interpolation and bi-cubic interpolation.

Findings

The algorithm with the highest performance is the nearest-neighbor interpolation. This algorithm performs a piece-wise operation on calculating each unknown value in the input. The output of each value is based on the spatial arrangement of the neighboring values in a 2 dimensional space. The nearest neighboring value is selected as the output. The result of the images are bit sharp. That is because a neighboring pixel with equal intensity is copied over in the current location. The use for this algorithm allows for quick large image generation and usage is well known for finding edges due to the sharpness of the image.

The algorithm with the lowest performance however with the highest quality is produced by the bi cubic interpolation with respect to the list mentioned above. It performs a third order computation in calculating each value.

The algorithm that falls in the middle of these three mentioned is bi-linear interpolation. This algorithm performs a second order computation in calculating each pixel value. The result of the image is bit blurry however each pixel value is calculating by interpolating linearly in one dimensional space in the horizontal direction then interpolating linearly in one dimensional space in the vertical direction consider the result from the horizontal direction. The previous results into a quadratic operation. This algorithm allows for interpolating of unknown values creating an estimation of pixel strength. This is a useful tool to use when zooming in pictures and trying to interpret information.

Observation

It is fascinating how resizing allowed images to become bigger. When using a graphical tool to spread an image usually results in pix-elated image. Mathematically this process can be performed more effectively.

Looking at cell 1 and 2. Imediate observation can be seen at the pixel level. The nearest-neighbor has a textured arrangement , while the bilinear has a smooth arrangement.

Performed both operations one image, by increasing its size by bi-linear and then increasing it by nearest-neighbor. There is a slight improvement on blur and sharpness. This makes me think that these algorithms can be used interchangeably as well depending on the use case. Performed bi-linear increase and then decrease the size by a half using both algorithms the result was the same by visual inspection. In some cases the second order solution isn't the best choice.

It looks like some image viewer softwares also utilize these algorithm. When I zoom in using a software I can see jagged lines that eventually smooth out. I open the images on the browser to continue evaluating the images.

I was not able to zoom in much in the original images compared to the resized images. Looking at cell 2 image, information can from both algorithms sort of help make out the structure cell. My opinion is that jaggedness helps with edges and depth by visual inspection. The blurry image after looking at the jagged image helps determining how spatial arrangement ties together.

Road blocks

the structure of the resize package was a bit confusing in the beginning. However it makes more sense after playing around with the functions. Although one can argue that the initial structure influence a programmer to follow a specific way to program. The setup can be done in various ways which could server other benefits. The current setup is beneficial with modularity. More interpolation algorithms can be added in the package while more scaling algorithms can be added in the scale algorithm package.

The bit size wasn't specified for the intensity values. My outputs are casted to unsigned integer 16bits.

I saw in literature algorithms would copy the initial images over to a new scaled space with spacing first then perform interpolating operations. This would mean the program would need to raster scan twice. I wrote it to where the images are scanned and operations performed in one sweep. The difficulty arose when trying to determine the next pixel locations. This was solved by math leveraging the known ratios, flooring and ceiling operations.

Difficulties with understanding the problem

The cell images are used when resizing. Questioned if the problem requires the analysis of the cell structure. If so the cell name would be helpful. Then looking into the layout and structure of the cell in literature would help assess the varying cell morphology displayed.