Department of Computer Science
Technical University of Cluj-Napoca

Image Processing

Laboratory activity 2019-2020

Project title: Zooming and Shrinking

Name: Kelemen Máté
Group: 30432
Email: mate.kelemen009@gmail.com

Contents

Chapter 1

Overview

In computer graphics and digital imaging, image scaling (zooming and shrinking) refers to the resizing of a digital image. In video technology, the magnification of digital material is known as upscaling or resolution enhancement. When scaling a raster graphics image, a new image with a higher or lower number of pixels must be generated. In the case of decreasing the pixel number (scaling down) this usually results in a visible quality loss. The issue at hand is to obtain the up or downscaled image (by a given amount) with the best visual quality. For this purpose, three simple, well-known image processing algorithms were implemented in code. My approach was to

1. Create an image of size (height * zoomFactorY, width * zoomFactorX)
2. Project the source image to the destination image by inserting zoomFactor sized gaps
3. To find the color of the pixels in the gaps, interpolate using multivariate interpolation (Nearest-Neighbor, Bilinear, Bicubic) between these pixels.

Chapter 2

Theoretical background

There are many algorithms designed to up- or downscale an image. The 3 algorithms eimplemented in this project are the Nearest - neighbor interpolation, the Bilinear interpolation and the Bicubic interpolation. Interpolation is the problem of approximating the value of a function for a non-given point in some space when given the value of that function in points around (neighboring) that point. In a nutshell, all the 3 algorithms work by pixel replication and inserting empty spaces in between. The way the empty gaps are filled with color are algorithm specific, and all three algorithms give different results.

2.1 Nearest-neighbor interpolation

The nearest neighbor algorithm selects the value of the nearest point and does not consider the values of neighboring points at all, yielding a piecewise-constant interpolant. The algorithm is very simple to implement and is commonly used.

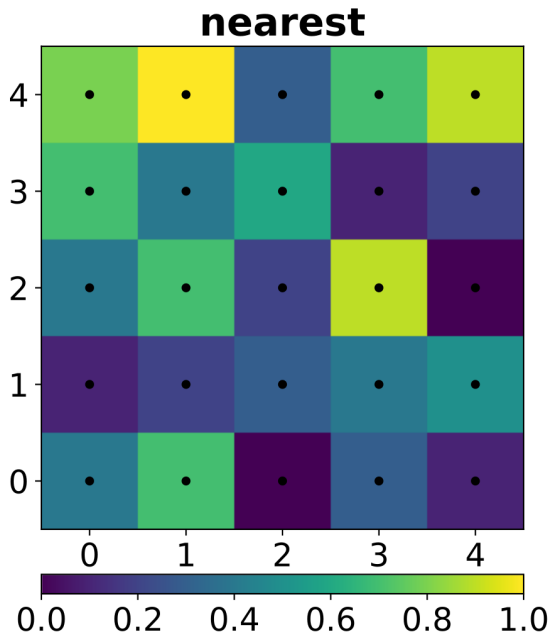

Figure 2.1: Nearest neighbor interpolation on a uniform 2D grid (black points). Each coloured cell indicates the area in which all the points have the black point in the cell as their nearest black point.

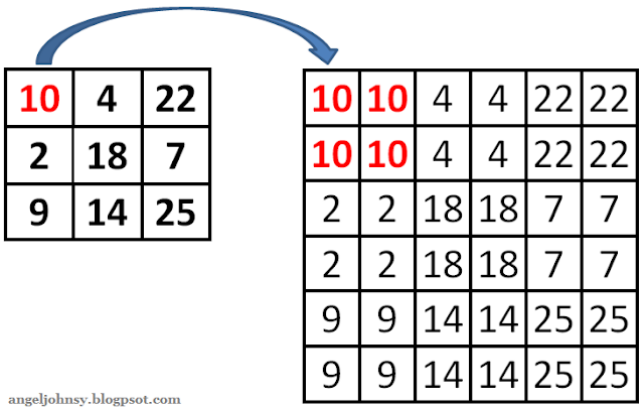

Figure 2.2: Nearest neighbor interpolation

2.2 Bilinear interpolation

Bilinear interpolation is an extension of linear interpolation for interpolating functions of two variables (e.g., x and y) on a rectilinear 2D grid. Bilinear interpolation is performed using linear interpolation first in one direction, and then again in the other direction. Thus, it uses 4 nearest neighbors (2x2 grid) and takes their weighted average to produce the output. Bilinear interpolation is one of the basic resampling techniques in computer vision and image processing, where it is also called bilinear filtering or bilinear texture mapping.

1. Suppose we have 4 pixels located at (0,0), (1,0), (0,1) and (1,1) and we want to find value at (0.3,0.4).

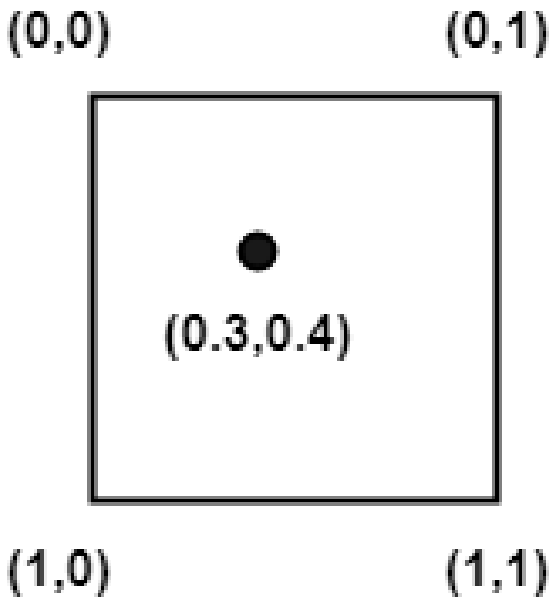

Figure 2.3: Nearest neighbor interpolation

2. First, find the value along rows i.e at position A:(0,0.4) and B:(1,0.4) by linear interpolation.

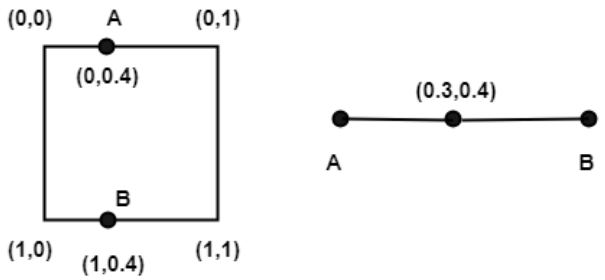

Figure 2.4: Nearest neighbor interpolation

3. After getting the values at A and B, apply linear interpolation for point (0.3,0.4) between A and B and this is the final result.

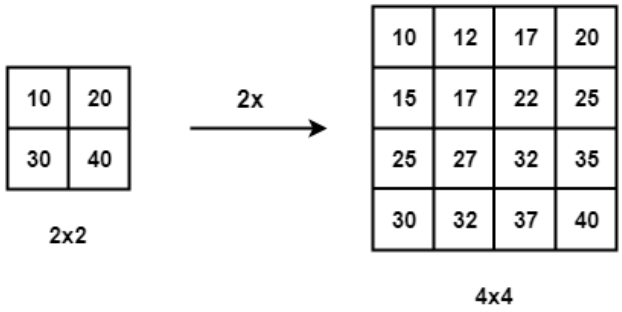

Figure 2.5: Nearest neighbor interpolation

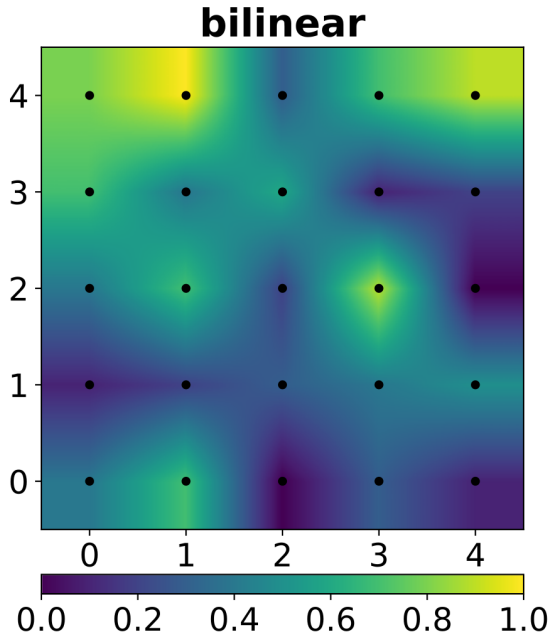

Figure 2.6: Bilinear interpolation on the same dataset as above. Derivatives of the surface are not continuous over the square boundaries.

2.3 Bicubic interpolation

In mathematics, bicubic interpolation is an extension of cubic interpolation for interpolating data points on a two-dimensional regular grid. The interpolated surface is smoother than corresponding surfaces obtained by bilinear interpolation or nearest-neighbor interpolation. Bicubic interpolation can be accomplished using either Lagrange polynomials, cubic splines, or cubic convolution algorithm. In image processing, bicubic interpolation is often chosen over bilinear or nearest-neighbor interpolation in image resampling, when speed is not an issue. In contrast to bilinear interpolation, which only takes 4 pixels (2x2) into account, bicubic interpolation considers 16 pixels (4x4). Images resampled with bicubic interpolation are smoother.

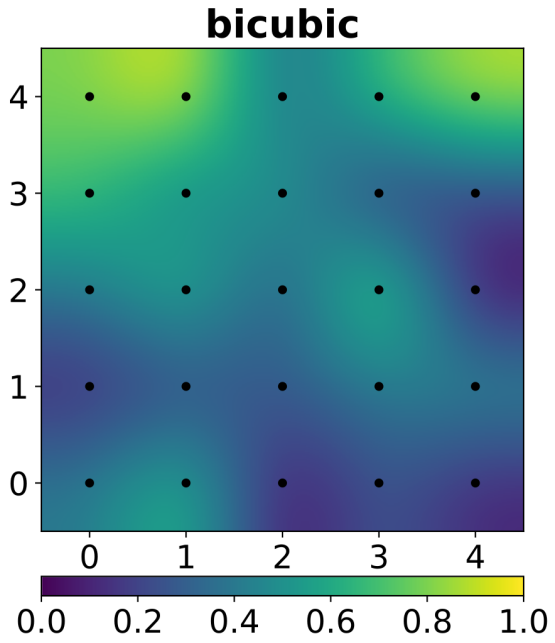

Figure 2.7: Bicubic interpolation on the square consisting of 25 unit squares patched together. The black dots are the locations of the prescribed data being interpolated.

Chapter 3

Design and Implementation

All the 3 algorithms were implemented as functions in the code. While the nearest-neighbor interpolation requires only finding the nearest image pixel, in case of both the Bilinear and Bicubic interpolation, we have to find the data points. The projection of a pixel from the destination image onto the source image is floor(iterator / zoomFactor) (iterator is either widthwise or heightwise). By subtracting this value from iterator / zoomFactor, we get the value between 0 and 1, this will become the interpoland. We find the rest of the data points (2x2 for bilinear, 4x4 for bicubic), by knowing that the top-left corner has the coordinates (floor(row / zoomFactorX), floor(column / zoomFactorY)). For the bicubic interpolation, the interpolation formulas are:

$$f(p_0, p_1, p_2, p_3, x) = (-\frac{1}{2}p_0+\frac{3}{2}p_1-\frac{3}{2}p_2+\frac{1}{2}p_3)x^3+(p_0-\frac{5}{2}p_1+2p_2-\frac{1}{2}p_3)x^2+(-\frac{1}{2}p_0+\frac{1}{2}p_2)x+p_1$$

Figure 3.1: Cubic interpolation formula in 1 dimension

$$g(x, y) = f(f(p_{00}, p_{01}, p_{02}, p_{03}, y), f(p_{10}, p_{11}, p_{12}, p_{13}, y), f(p_{20}, p_{21}, p_{22}, p_{23}, y), f(p_{30}, p_{31}, p_{32}, p_{33}, y), x)$$

Figure 3.2: Cubic interpolation formula in 2 dimensions, where pij are rows and columns from the image window

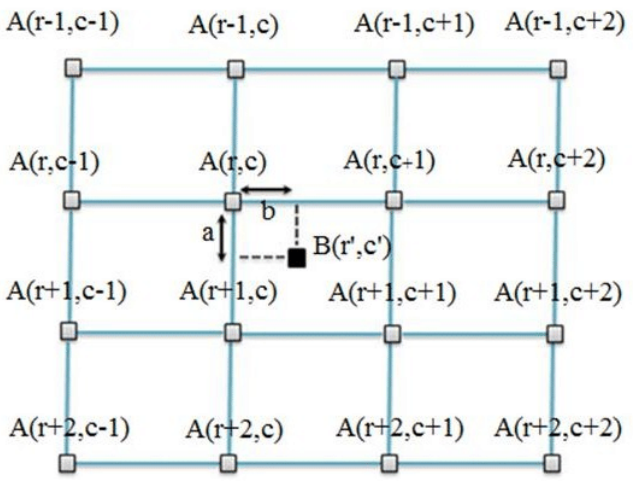

Figure 3.3: Bicubic interpolation window

Chapter 4

Results

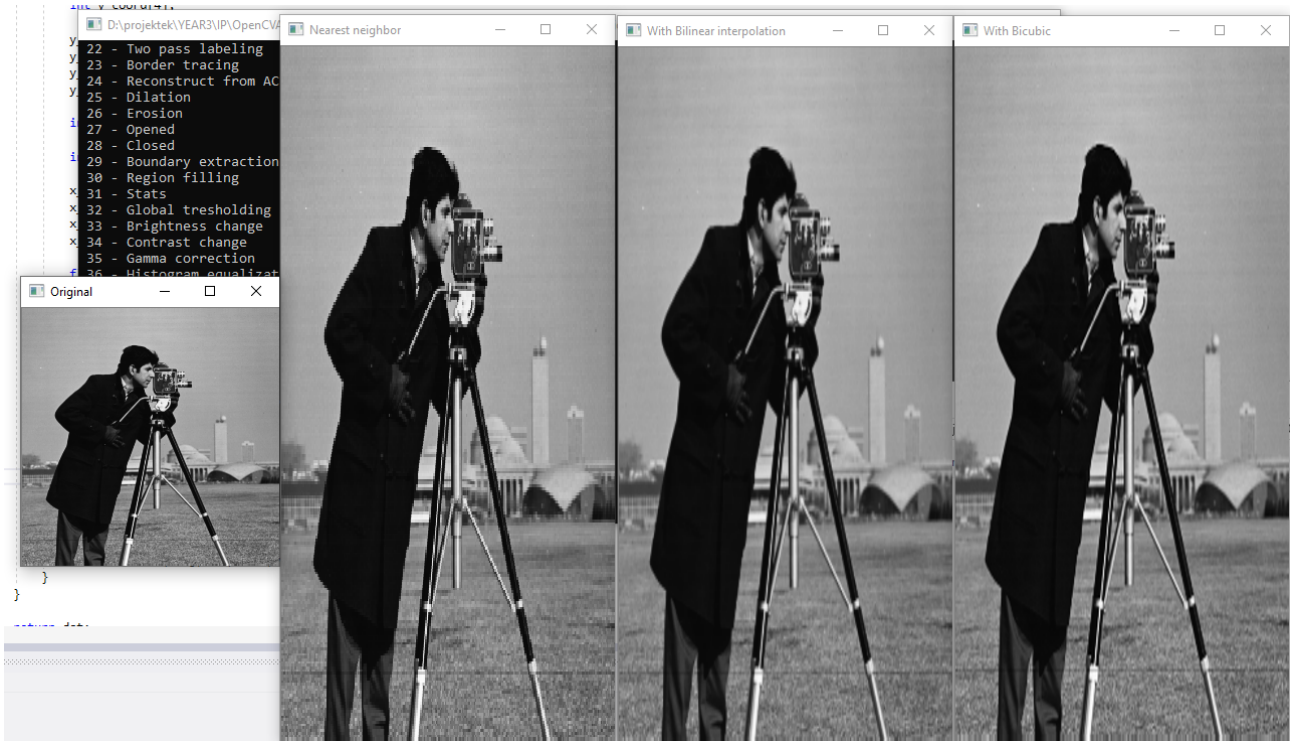

Figure 4.1: Cameraman.png with 2.7 and 1.3 zoom factors

Figure 4.2: Lena.png, Nearest-neighbor, 4x upscaled

Figure 4.3: Lena.png, Bilinear, 4x upscaled

Figure 4.4: Lena.png, Bicubic, 4x upscaled

Chapter 5

Source of Data

https://www.paulinternet.nl/?page=bicubic
https://en.wikipedia.org/wiki/Bilinear_interpolation
https://en.wikipedia.org/wiki/Bicubic_interpolation
https://en.wikipedia.org/wiki/Nearest-neighbor_interpolation
Computerphile - Bicubic Interpolationhttps://www.youtube.com/watch?v=poY_nGzEEWM
Computerphile - Resizing Imageshttps://www.youtube.com/watch?v=AqscP7rc8_M&t=1sc