CS 663: Assignment 1 Question 1

Shubham Kar - 180070058, Yash Sanjeev - 180070068
Garaga V V S Krishna Vamsi - 180070020, Rishav Ranjan - 180070045

September 1, 2020

Contents

1 Image Shrinking

In this sub-problem, we were expected to shrink the image by some given scaling factor d. The algorithm behind is that we took fewer pixels into account while shrinking the image. In each direction, we skipped d pixels and hence the image's dimension reduced by a factor of d in both height and width for all three channels Red, Blue and Green.

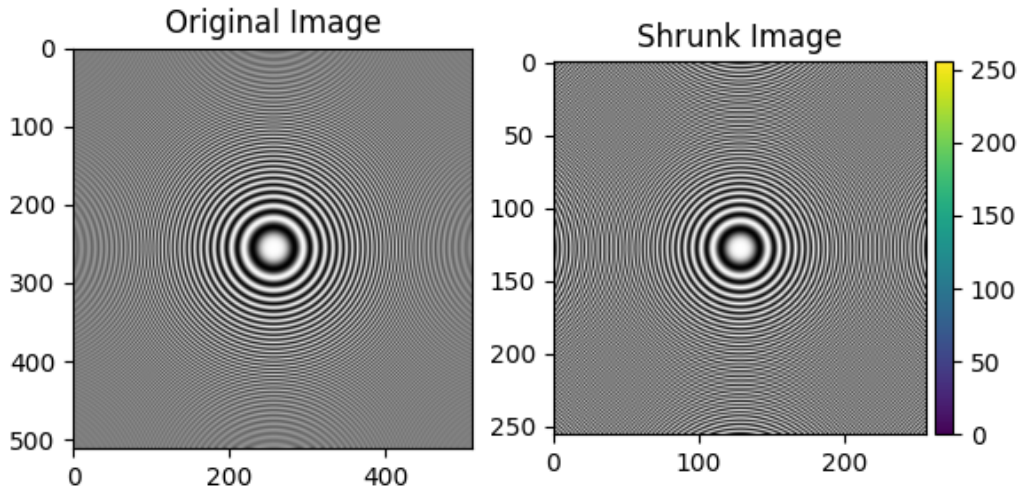

Figure 1: Image Shrinking

2 Image Enlargement

2.1 Bilinear Interpolation

In this subproblem we have to enlarge the image using Bilinear Interpolation, thus we need to calculate the values of pixels at many additional unknown points other than the values which are known to us, and hence we use the pixels data at the corners of a rectangle surrounding the point whose data we need to

calculate. We do two linear interpolation in x direction whose formula is given below :

$$f(x, y_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}),$$
$$f(x, y_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}).$$

Now, we use the two results of linear interpolation in x-direction to do a linear interpolation in y-direction whose formula is given below

$$f(x, y) \approx \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2)$$

Here, values at the corners are $Q_{11} = (x_1 1, y_1), Q_{12} = (x_1, y_2)$, $Q_{21} = (x_2, y_1), Q_{22} = (x_2, y_2)$.

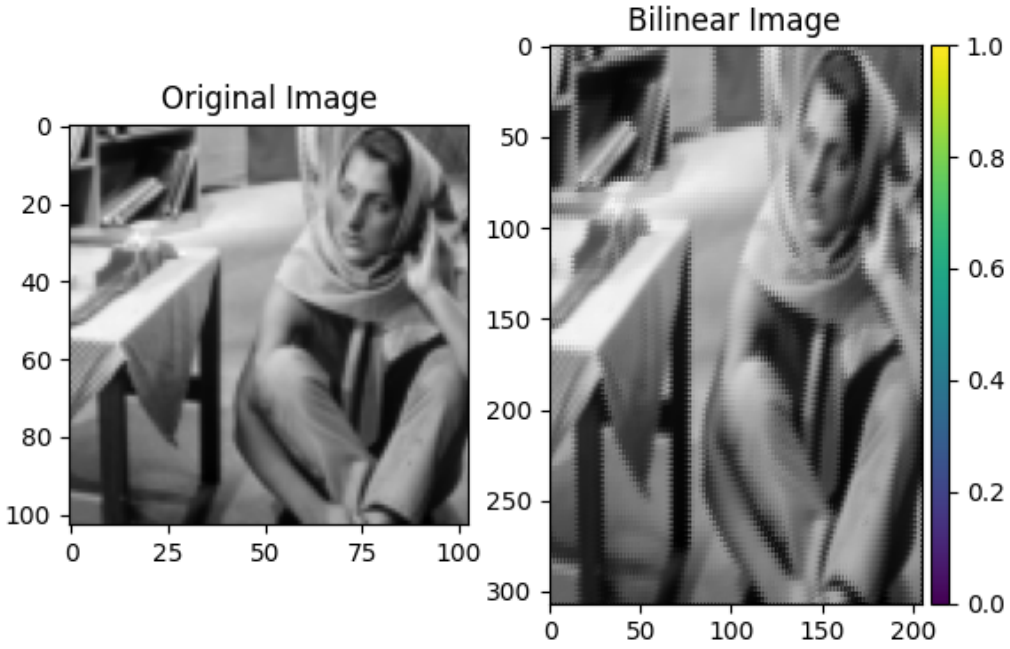

Figure 2: Bilinear Interpolation

2.2 Nearest-Neighbor Interpolation

In this subproblem we have to enlarge the image using Nearest-Neighbor Interpolation, thus we need to calculate the values of pixels at many additional unknown points other than the values which are known to us, and hence we use the pixels data of that corner of a rectangle which is more closer to the point. And hence there is a abrupt discontinuity in the newly calculated pixel values. The result is that the images looks grainy because of this discontinuity in pixel values.

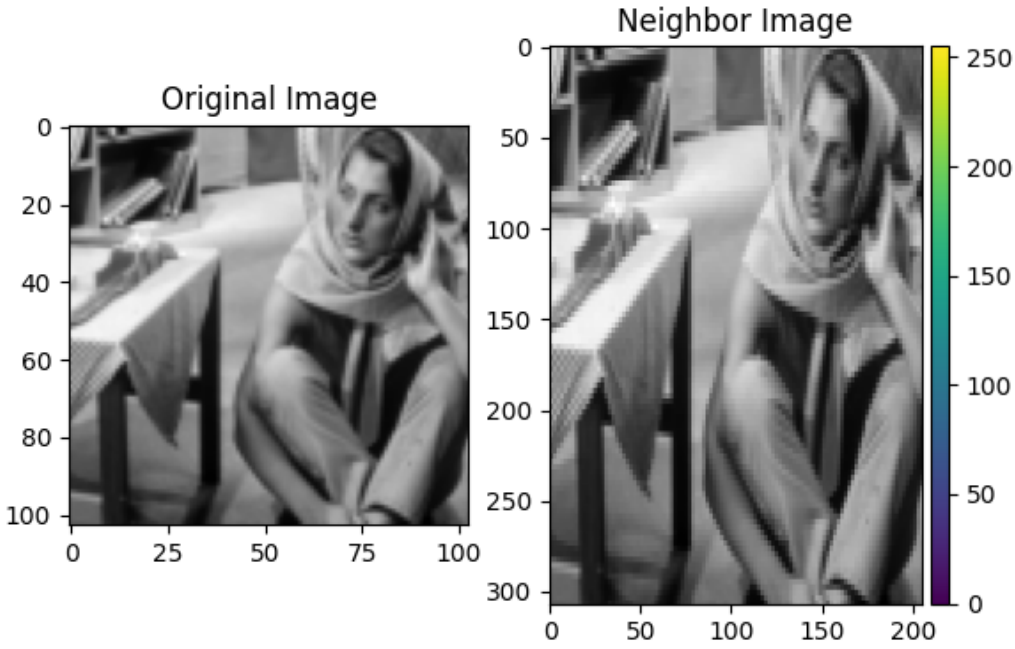

Figure 3: Nearest Neighbor Interpolation

2.3 Bicubic Interpolation

In this sub-problem we use bicubic interpolation for enlarging our image. 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. Suppose the function values f and the

derivatives f_x, f_y and f_{xy} are known re known at the four corners (0,0), (0,1), (1,0) ans(1,1) of the unit square. The interpolated surface can then be written as

$$p(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}x^i y^j$$

The 16 coefficients can be obtained by solving following equation. Matching p(x, y) with functional values can yield 4 co-efficients

$$f(0,0) = p(0,0) = a_{00} \tag{1}$$
$$f(1,0) = p(1,0) = a_{00} + a_{10} + a_{20} + a_{30} \tag{2}$$
$$f(0,1) = p(0,1) = a_{00} + a_{01} + a_{02} + a_{03} \tag{3}$$
$$f(1,1) = p(1,1) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij} \tag{4}$$

Likewise, eight equations for the derivatives in the x and the y directions:

$$f_x(0,0) = p_x(0,0) = a_{10} \tag{5}$$
$$f_x(1,0) = p_x(1,0) = a_{10} + 2a_{20} + 3a_{30} \tag{6}$$
$$f_x(0,1) = p_x(0,1) = a_{10} + a_{11} + a_{12} + a_{13} \tag{7}$$
$$f_x(1,1) = p_x(1,1) = \sum_{i=1}^{3}\sum_{j=0}^{3} a_{ij}i \tag{8}$$
$$f_y(0,0) = p_y(0,0) = a_{01} \tag{9}$$
$$f_y(1,0) = p_y(1,0) = a_{01} + a_{11} + a_{21} + a_{31} \tag{10}$$
$$f_y(0,1) = p_y(0,1) = a_{01} + 2a_{02} + 3a_{03} \tag{11}$$
$$f_y(1,1) = p_y(1,1) = \sum_{i=0}^{3}\sum_{j=1}^{3} a_{ij}j \tag{12}$$

And four equations for the xy mixed partial derivative:

$$f_{xy}(0,0) = p_{xy}(0,0) = a_{11} \tag{13}$$
$$f_{xy}(1,0) = p_{xy}(1,0) = a_{11} + 2a_{21} + 3a_{31} \tag{14}$$
$$f_{xy}(0,1) = p_{xy}(0,1) = a_{11} + 2a_{12} + 3a_{13} \tag{15}$$
$$f_{xy}(1,1) = p_{xy}(1,1) = \sum_{i=1}^{3}\sum_{j=1}^{3} a_{ij}ij \tag{16}$$

The expressions above have used the following identities:

$$p_x(x,y) = \sum_{i=1}^{3}\sum_{j=0}^{3} a_{ij} i x^{i-1} y^j \tag{17}$$

$$p_y(x,y) = \sum_{i=0}^{3}\sum_{j=1}^{3} a_{ij} x^i j y^{j-1} \tag{18}$$

$$p_{xy}(x,y) = \sum_{i=1}^{3}\sum_{j=1}^{3} a_{ij} i x^{i-1} j y^{j-1} \tag{19}$$

Thus, taking into the account the continuity and derivatives continuity at the boundaries, we can easily compute the pixel values for arbitrary grid. And by using such interpolation we can be assured of getting a smooth image upon enlargement.

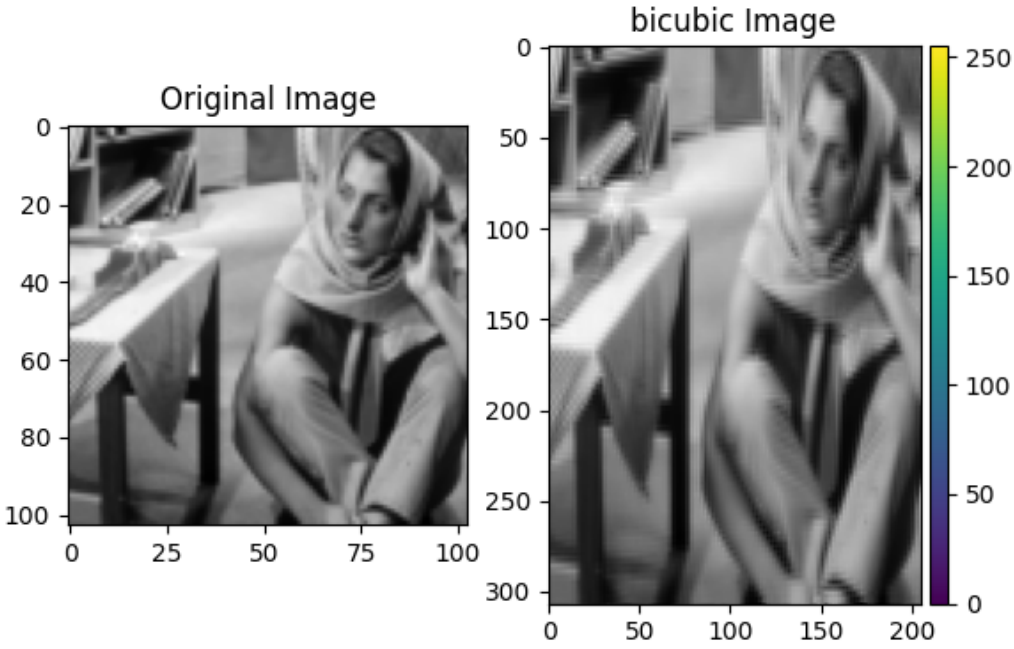

Figure 4: Bicubic Interpolation

2.4 Comparison

When all the three image enlargement methods are applied, we observe that the one with bicubic interpolation was very smooth and the one with nearest neighbor interpolation is very grainy. The one with bilinear interpolation is somewhat between them. The reason of nearest-neighbor one being grainy is that, the interpolated pixels takes the value of the corner that is closest to it, thus creating an observable stark discontinuity in pixel values. The bicubic one is smooth because while calculating the values of interpolated pixels, we take into account the constraint of continuity and derivative continuity at the boundaries. Thus, the resulting pixel values are very much approximated with respect to the corners and hence the image is smooth.

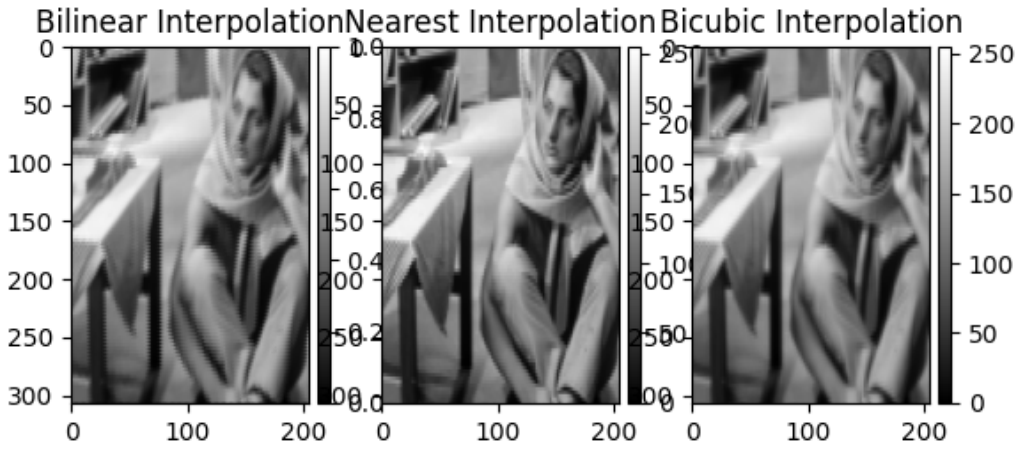

Figure 5: Comparison between different techniques of interpolation

We duly observed that the nearest neighbor interpolation was giving a more pleasing output than bilinear or bicubic interpolation. We believe that this is due to the sharp contrast features of the original image which are diluted by the bilinear and bicubic interpolations resulting in a blurred output.

3 Rotation using Bilinear Interpolation

In this subproblem, we try to rotate the image by a fixed angle *deg*. We calculate the dimensions of the image, and then for each pixel, we calculate the integer value of the pixel number if we rotate the point wrt middle pixel of the image by angle *-deg* . Now, if the rotated point is inside the image box, we calculate the bilinearly interpolated pixel value of that point. if the point is outside the image box, we ignore it. The observation is such that the part of the images at the corners are cut due to rotation and we get a rotated image.

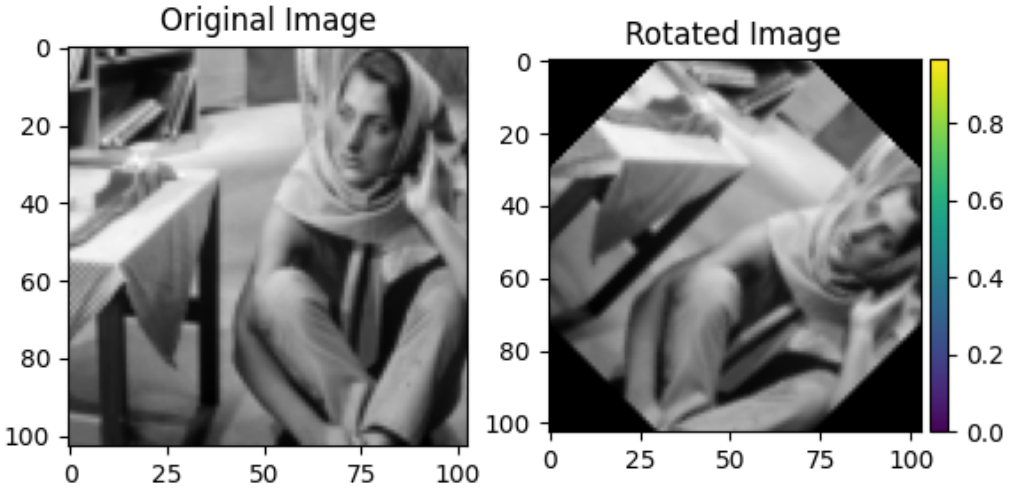

Figure 6: 45° rotation

4 Conclusion

By doing this question, we get to know about various interpolation methods and we observed their pros and cons, also we were able to learn about shrinking and rotating the image which was overall a good experience.