Homework 1

Carla Andrade
Introduction to Signal and Image Processing

April 4, 2023

1 Regular Tesselation

1.1 Identify all shapes that satisfy the two conditions above.

The shapes that satisfy the two conditions are equilateral triangles, hexagons, and rectangles.

1.2 Formally show that no other shape satisfies the above conditions.

Only these 3 shapes satisfy this condition, because to tessellate, a form must have all its interior angles that can divide 360 and give a whole number. Therefore all shape interior angle must be in the following list: 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 18, 20, 24, 30, 36, 40, 45, 60, 72, 90, 120, 180, and 360. Only the 3 former shapes can have all its angles from this list.

2 Chamfer distances

To implement the Chamfer Algorithm, I did the following steps for each figure:

1. Loaded the edge map and created a distance map of the same shape as the edge map.
2. Initialized the distance map with 0 corresponding to the edges of the edge map, and ∞ otherwise.
3. Created a function Chamfer distance that calculates the new value for a specific pixel. This function takes 3 arguments: the matrix, a list of AL or BR neighbor pixels of x (see figure 2) and if it is BR or not. The function loops through each pixel of the image, ignoring the border of the image (respectively range(1, height-1) and range(1, width-1), because for implement Chamfer the pixel x need to have 8 neighbors. it also check if it's BR or AL because on start top to bottom, left to right (AL) and the other is start from bottom to top, right to left (BR). Then for each x it create a list and loop through all the AL or BR neighbors of x and calculates the Manhattan distance (L1) between the neighbor pixel q and x and adds it to the value q. As a shown in figure 2, I already put the difference between AL and x and BR and x in 2 lists. Therefore

to calculate the distance map between each q and x , the abs value of $q1$ and $q2$ was simply added to the value q.

4. Finally, the new value of x is equal to the minimum value in the list. Respectively, the function calculates:

$$\text{distance map}(x) = \min_{q \in BR \cup AL} [L_1(x, q) + \text{distance map}(q)]$$

5. This function is applied 2 times: first with AL and then with BR.

$$\text{distance map}(x) = \min_{q \in AL} [L_1(x, q) + \text{distance map}(q)]$$

$$\text{distance map}(x) = \min_{q \in BR} [L_1(x, q) + \text{distance map}(q)]$$

6. The function produced the following images:

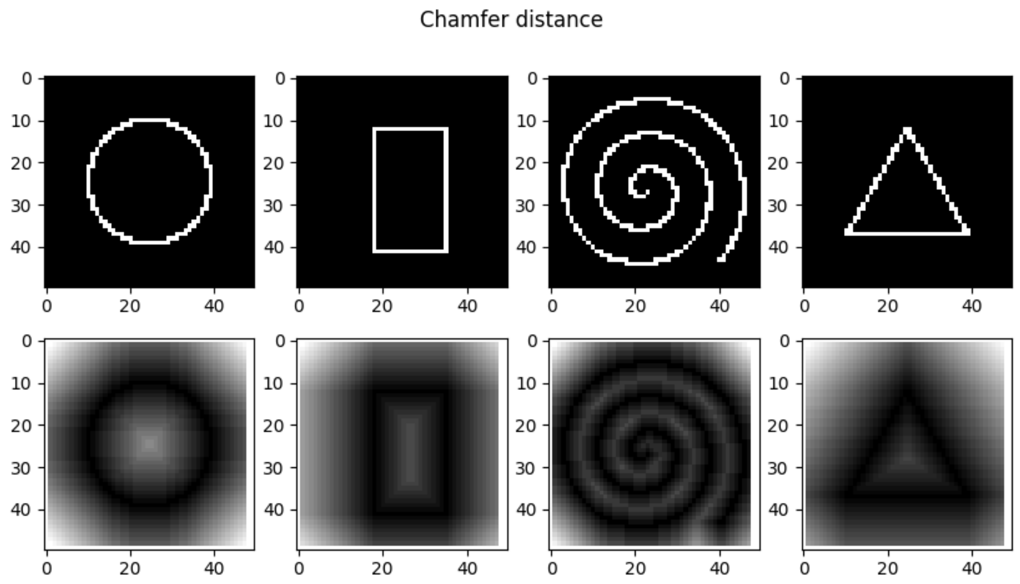

Figure 1: The first row correspond to the edge maps and the bottom row the corresponding distance map.

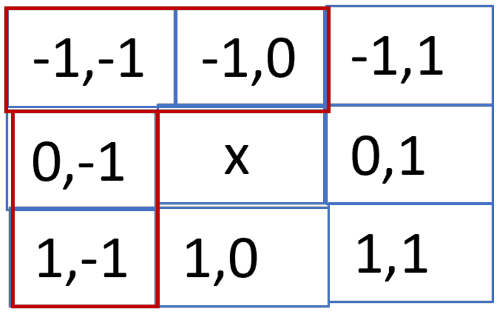

Figure 2: In red are the AL neighbor and in blue the BR neighbor of x. the number correspond to the position of the neighbor in function of the pixel center, the position of x being 0,0.

3 Bilinear interpolation

3.1 To implement a bi linear interpolation of images it was first created a function named interp. This function take 3 argument: 3 numpy arrays,y_{val}, x_{val} and x_{new}. Based on the 2 first argument it determine the y value of x_{new}. To do so, the function loop through allx_{new}. Then for each value it first check if x_{new} is smaller or bigger than the smallest and biggest value of x_val. if so y_{new} is set to the the smallest y_{val} or biggest y_{val} respectively. If not the function determine between which value is x. To do so it substrate all x_{val} and x_{new} and order it, but taking their original index (from x_{new} list) . the 2 smallest value obtain are the values between which is the current x_{new}, respectively $x0$ and $x1$. Finally the y[i] value is calculated with the following formula:

$$y_{[\text{i}]} = y_0 + \frac{(x - x_0) \times (y_1 - y_0)}{(x_1 - x_0)}$$

3.2 The last function permit to interpolate one dimensional signal by a given factor. This function will apply interp function on a 1D array. It takes 2 argument the signal and the scale factor. The signal is equivalent at the y_{val} in the function interp. The x_{new} correspond to a sequence of number with a max equal to the length of the signal and $min = 1$. The numbers of x_{new} are equal to the $lenghtofsingla * scale_factor$ and the numbers are evenly spaced. finally the x_{val} correspond to a range of y_{val} length.
Finally to do a bilinear interpolation on images it was first applied 1D interpolation on each rows of the images then 1D interpolation on each column of the images. To render the images after the transformation a scaling had to be done. The values were divided by 255 because the plot function can only render pixel values that are either between 0-1 or whole values between 0-255 for the RGB images. As shown in figure 3,4,5 the images are the sames but with more pixels. Figure 6 is a zoom of the original image of the monkey and its bilinear interpolation, on its eye. As we can see in the bilinear images the pupil has more pixel in it.

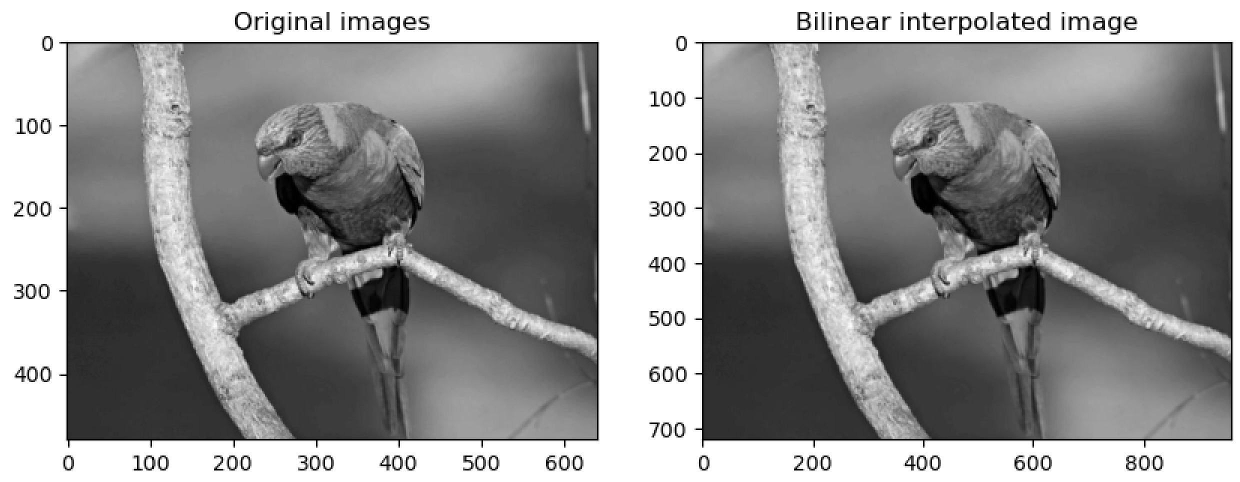

Figure 3: bird Image. On the left is the original image. On the right is the image after bilinear interpolation with a scaling factor of 1.5.

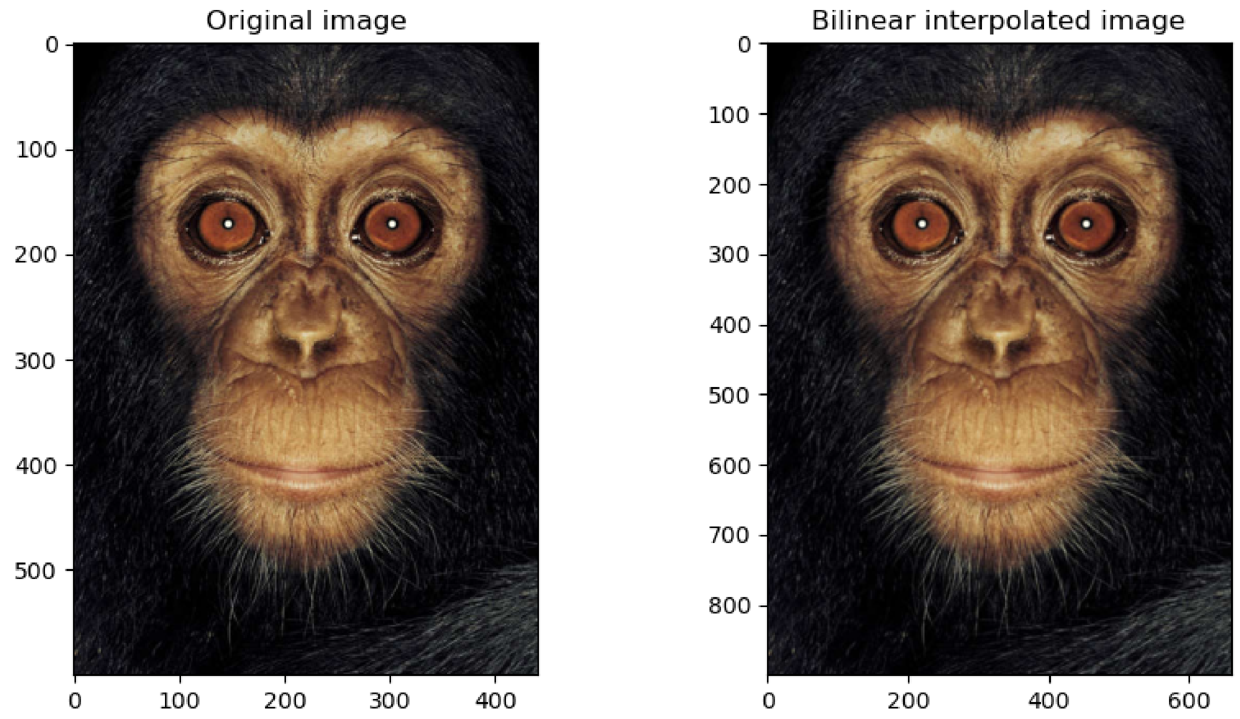

Figure 4: Monkey Images . On the left is the original image. On the right is the image after bilinear interpolation with a scaling factor of 1.5.

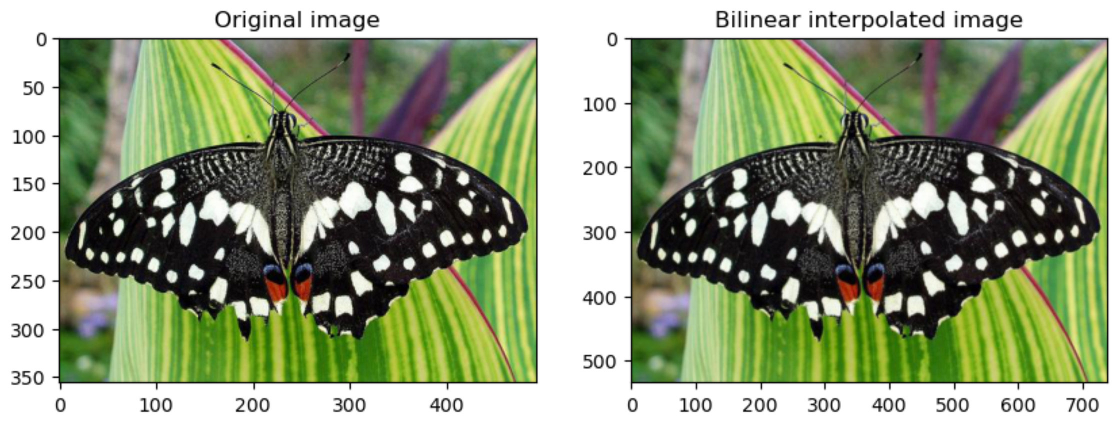

Figure 5: Butterfly Images . On the left is the original image. On the right is the image after bilinear interpolation with a scaling factor of 1.5.

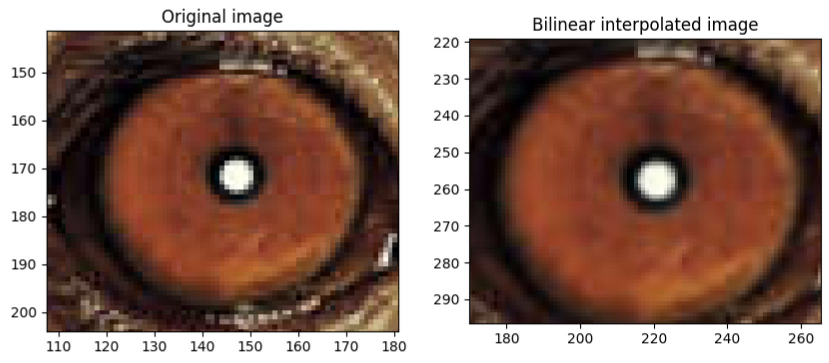

Figure 6: Eye of the Monkey Images . On the left is the original image. On the right is the image after bilinear interpolation with a scaling factor of 1.5. The pupil in the right images has more pixel.