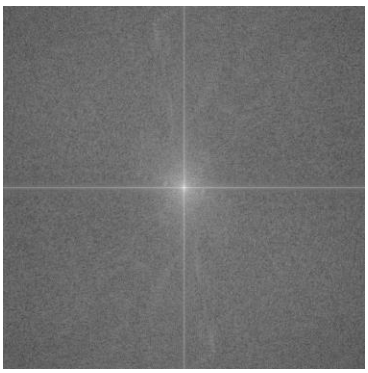
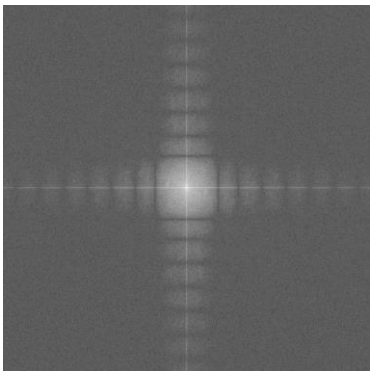
Part 2

We don’t come out a perfect morphing image. The main reason is that the resize of two images is different, so we have to resize one of them. Also It is hard to match their faces on the same position.

Part 3

3)





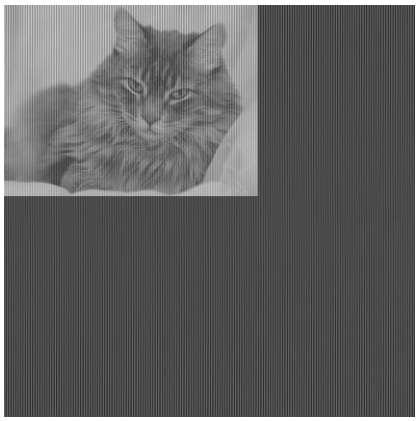
After applying Gaussian filter to the spectrum, the spectrum has a decreasing frequency from center along x and y axes. Also the frequency patterns along the axes are no longer continuous.

4)

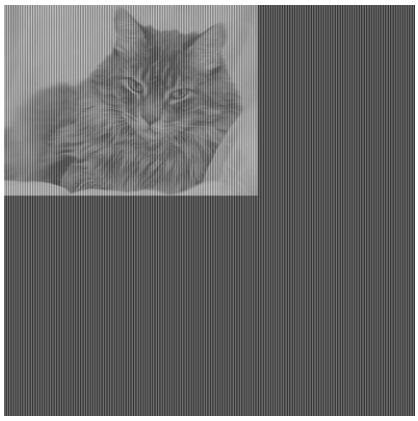
We failed to recover the image from its blur version. It may be caused by the normalization of the blur image. So when we read it as the input, its pixel value may be different. Another issue we may have is the rounding of image part and real part of image and filter. So when we are doing the complex division we can’t get the pixel value of original image.

Part 4

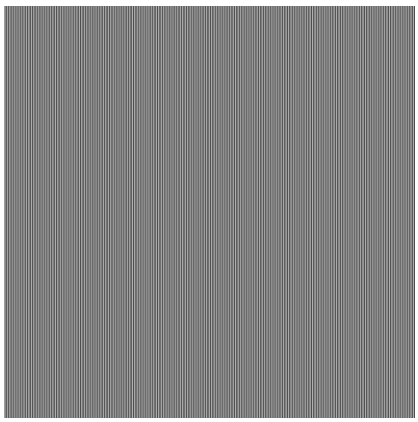
1) By observing the example in the instruction, we notice that the circle pattern on the spectrum has a radius of ¼ image width. So we set the r equivalent to ¼ image width. And the length l of vector will depend on N. And we test multiple values for alpha, which affect the intensity of watermark.



alpha =10



alpha = 20



alpha = 50

2) We initialize vector C which has the same length as vector V from part 4.1. Also we use the same N for this part. So when we test the code we should always get the result that there is a watermarking in the image. We normalize the real part of image with watermarking from 0 to 1. And we restore the value. Then we go over the real part of image on the same spot of part 4.1 and restore the value in C. If we V and C have correlation, then V[i] (value of 1) should correspond to larger C[i] (value may be larger than 0.5) and V[i] (value of 0) should correspond to smaller C[i](value may be close to 0).

Then we calculate C[i]/V[i] when V[i] =1. Remember that if C and V have no correlation C[i] will be small when V[i] =1. So the value of C[i]/V[i] may be close to 1. Then we loop over the V and add up the value of C[i]/V[i].

We random N between 1 and image.width(). And when V and C have correlation the sum of C[i]/V[i] mostly are lager then 0.9. So we set threshold to 0.9 to detect watermarking.