Report on the First Homework (Convolutions, Fourier Transforms, and Image Pyramids)

Negar Nejatishahidin
Computer Science
George Mason University
Virginia, United states
nnejatis@gmu.edu

***Abstract*—This is a report on the first Homework project, mostly implementing different kinds of filters, up sampling images, and making image pyramids.**

I. P1.1.1 Some Simple Filters

In this part we are asked to implement 4 filters f_a, f_b, f_c, and f_d, name them and see if they are separable or not. To make these filters I only made a 2d numpy array and multiply the whole array with the coefficient to make magnitude one.

A. f_a

- This filter is Box blur filter, it makes the resulting image blurry since it replaced each pixel with the averaged of a 3*3 window. The result is figure 1(a).
- This filter is separable:
  $f_{a-sep} = [1, 1, 1]$
  $f_a = f_{a-sep}^T f_{a-sep}$

B. f_b

- This filter also blur the image but less than the box blur filter since it only uses the 1*3 window and average 3 pixels.The result is figure 1(b).
- This filter is separable:
  $f_{b-sep1} = [0, 1, 0]$
  $f_{b-sep2} = [1, 1, 1]$
  $f_b = f_{b-sep1}^T f_{b-sep2}$

C. f_c

- This filter is a negative of horizontal derivative(edge), with the average over the 3*1 window. The result is figure 1(c).
- This filter is separable:
  $f_{c-sep1} = [1, 1, 1]$
  $f_{c-sep2} = [-1, 0, 1]$
  $f_c = f_{c-sep1}^T f_{c-sep2}$

D. f_d

- This is a 3*3 Identity Matrix. This filter is average over the diagonal of the image.The result is figure 1(d).
- This filter is not separable.

Identify applicable funding agency here. If none, delete this.

II. P1.1.2 Image Derivatives

A. Sobel filters

In this section we asked for image derivatives I_x and I_y using the vertical and horizontal Sobel filters.

- To get the horizontal and vertical derivatives of an image I used I_xSobel and I_ySobel kernel convolved to the image. See the results in figure 2 .

$$I_xSobel = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} I_{yS}obel = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$I_x = I_xSobel * Image$$

$$I_y = I_ySobel * Image$$

- To compute the image gradient magnitude and angle I used following equations and then plot them as an image.

$$magnitude = \sqrt{I_x^2 + I_y^2}$$

$$angle = \arctan I_y / I_x$$

The results are shown in figure 2.

- The filter for image laplacian is as follow :

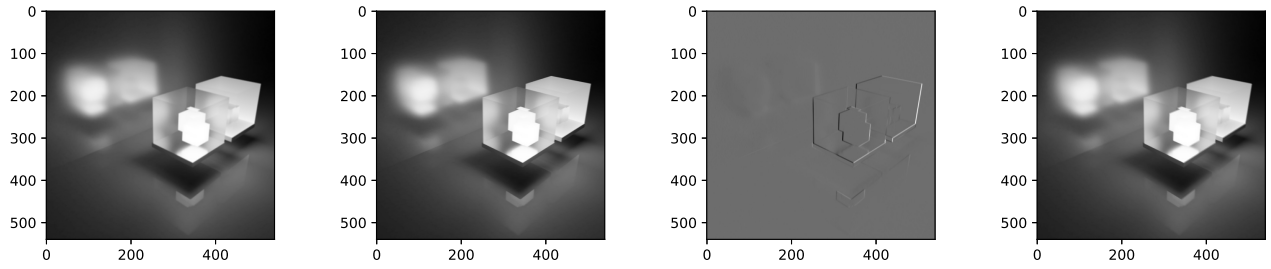
$$laplacian_filter = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$laplacian = laplacian_{filter} * Image$$

The image laplacian is the edge detector filter which uses the second derivatives of the image. The result is in figure 3.

III. P1.1.3 Gaussian Filtering

Gaussian filter is a blurring filter which makes the image smoother. One interesting property of Gaussian is that the Fourier transfer of a Gaussian is a Gaussian. I have used 3 different sigma 20, 10, and 2 with kernel width 60 figure 5. The results show that higher sigma makes the image more



(a) f_a * Image (b) f_b * Image (c) f_c * Image (d) f_d * Image

Fig. 1. Applying 4 different filters on image.

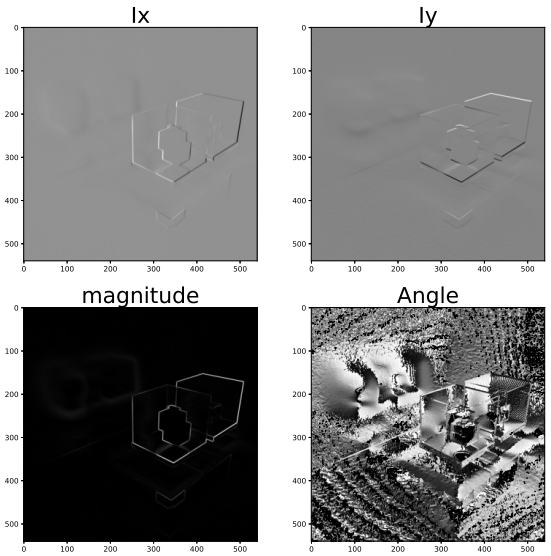


Fig. 2. The image derivatives I_x and I_y,and image gradient magnitude and angle.

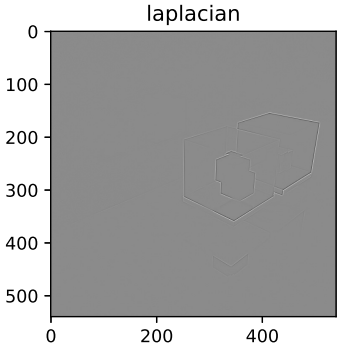


Fig. 3. The image laplacian.

blurry.

Theoretically it's we choose a sigma which is 1/3 of the kernel width, since we prefer the edge of the Gaussian close to zero. If we choose a bigger sigma, the edges are not close to zero and computing the convolution is computationally expensive. The sigma of the mentioned kernel is computed as follow :

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$G(0,0) = \frac{1}{2\pi\sigma^2} e^{-\frac{0}{2\sigma^2}} = 4/16$$

$$G(0,0) = \frac{1}{2\pi\sigma^2} = 1/4$$

$$\sigma = \sqrt{2/\pi}$$

Just to check if σ is correct :

$$G(1,1) = \frac{1}{2\pi 2/\pi} e^{-\frac{1^2+1^2}{2(2/\pi)}} = 1/16$$

$$G(1,1) = 1/4 e^{-\frac{\pi}{2}} = 1/16$$

Therefore :

$$e^{(-\pi/2)} = 1/4$$

Which this equation is roughly true.

IV. P1.1.4 Derivative of Convolution

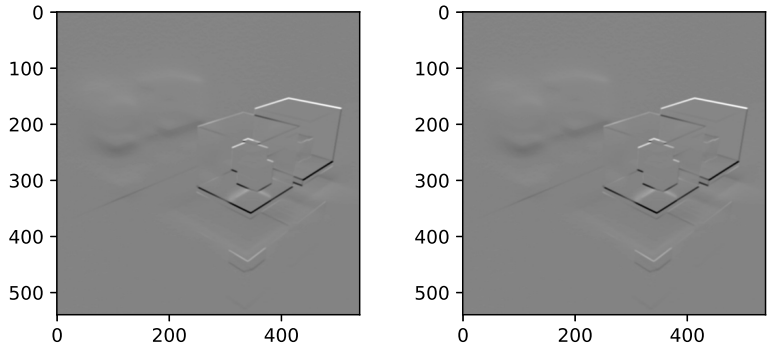
This part wants to show that the drivitive of convolution is equal to first do the drivitive and then apply the convolution. The results are shown in figure 4.

V. P1.2.1 Upsampling Kernels

In this section we are going to implement 3 different interpolation strategies, Nearest Neighbor, Biliniar, and Bicubic. I will discuss them in more details later.

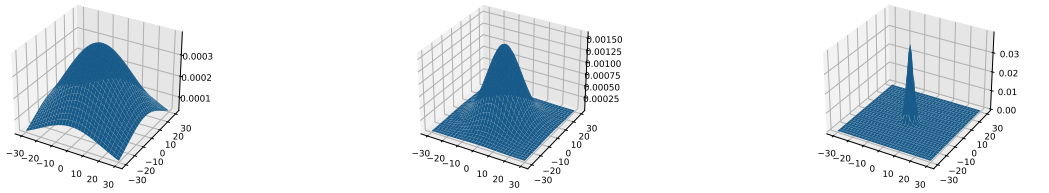
A. Nearest Neighbor interpolation

This interpolation replace the new pixel with the nearest neighbor. to get the nearest neighbor we can get the floor of the location of pixel in upsampled image divided by the upsampling ration which is exactly floor of (new y,new x). The result is shown in figure V-C.

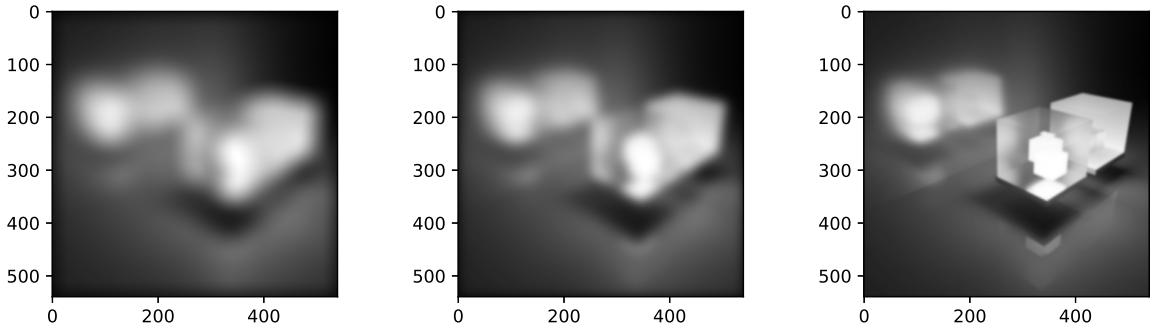


(a) Derivative of $f * g$. (b) Derivative of f then convolve with g.

Fig. 4. Derivative of convolution theorem.



(a) Gaussian filter with sigma 20 (b) Gaussian filter with sigma 10 (c) Gaussian filter with sigma 2



(d) Gaussian with sigma 20 applied on Image (e) Gaussian with sigma 10 applied on Image (f) Gaussian with sigma 2 applied on Image

Fig. 5. Applying 3 different Gaussian filters to image.

B. Bilinear interpolation

The logic behind this interpolation is find a linear function in two directions from the 4 pixels around the pixel that must be estimated. In more details look at figure 7. First we fined a linear function between Q_{12} and Q_{22} and fined the amount of R_2. Then we do the same for Q_{11} and Q_{21} to get the value R_1. Finally a linear function will be define between R_2 and R_1 to get the P value. instead of this linear function we can use a weighted average as well. The result is shown on figure 8.

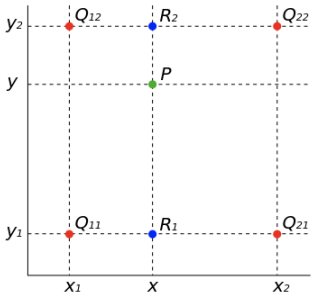


Fig. 6. Bilinear interpolation.

C. Bicubic interpolation

Bicubic interpolation outperforms other interpolation methods. However, if we have sharp changes in our image, it might not be the best solution.

Interpolation kernel :

$$W(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1 & \text{for } |x| \leq 1, \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a & \text{for } 1 < |x| < 2,\} \\ 0 & \text{otherwise,} \end{cases}$$

Interpolation uses the values of 16 pixels around the new pixel dst(x,y) [1][2]. (x,y) shows the location of pixels.

$$dst(x,y) = \frac{1}{2}\begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix} \begin{bmatrix} f_1 1 & f_1 2 & f_1 3 & f_1 4 \\ f_2 1 & f_2 2 & f_2 3 & f_2 4 \\ f_3 1 & f_3 2 & f_3 3 & f_3 4 \\ f_4 1 & f_4 2 & f_4 3 & f_4 4 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

Here, f means the values of pixels. x1, x2, x3, x4 are the distance of x direction from new pixel to near 16 pixels. y1, y2, y3, y4 are the distance of y direction.

The problem of Bicubic is that it cannot linearly interpolate straight lines, to improve this problem instead of using $a = -1$ we can set $a = -0.5$. The results are shown in figure 10 and 9.

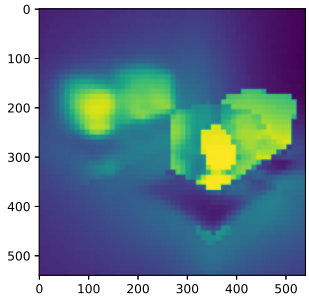


Fig. 7. Upsampling using Nearest Neighbor.

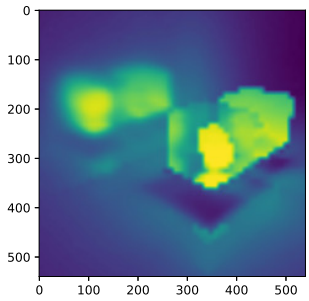


Fig. 8. Upsampling using Bilinear

VI. P1.2.2 Upsampling with Fourier Transforms

First we do Fourier transformation on the image. Then add the padding in the middle of the array, where the highest frequencies are. The easier way is to first use fftshift to move the zero frequency to the center of the array, then pad all around the array, then shift back and do invfft.See the result on figure 11(a) When we down sample image again it's exactly the image that we first down sampled, figure 11(b). The original down sampled image is figure 11(c). It seems that the result of up sampling using Fourier transformation is better than Nearest Neighbor. But it seems that Bicubic outperform all other methods. It make sense because

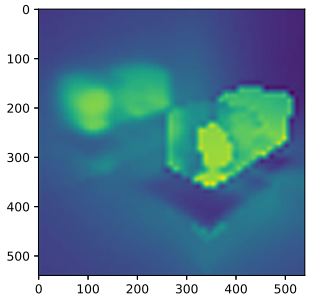


Fig. 9. Upsampling using Bicubic with $a = -1$.

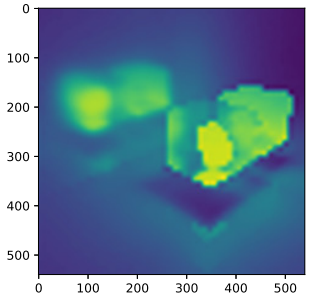
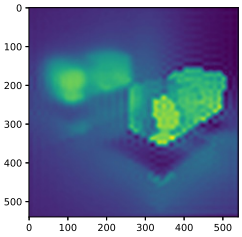


Fig. 10. Upsampling using Bicubic with $a = -0.5$.

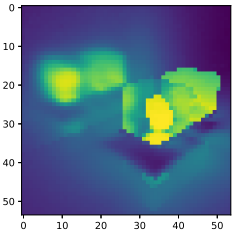
VII. P1.3 Hybrid Images (Contest)

Hybrid images combine the low spatial frequencies of one picture with the high spatial frequencies of another picture, producing an image with an interpretation that changes with viewing distance. This can achieved by using a Gaussian filter. Applying a Gaussian filter to the first image acts as a low-pass filter. Then, second image should convolve with a high-pass filter. This kernel can be achieved by $1 - G(x)$. The sigma value acts as a threshold between these two images. Figure 12(a).

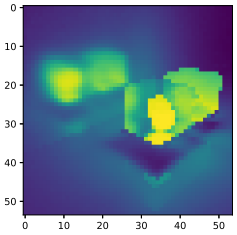
A low-pass pyramid is made by smoothing the image with an appropriate smoothing filter and then sub-sampling the smoothed image, usually by a factor of 2 along each coordinate direction. For this purpose, first a Gaussian filter used to make the image smoother. Then, the image size shrinks by a factor of two with removing the pixels alternately. Figure 12(b).



(a) Up sample using Fast Fourier Transform.

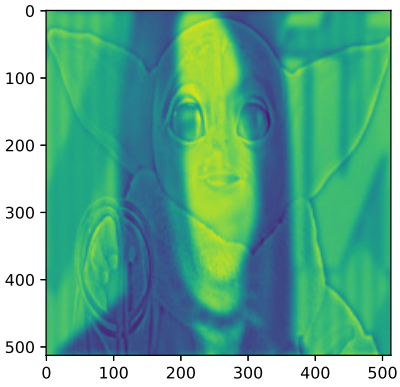


(b) Down sample of the up sampled image.

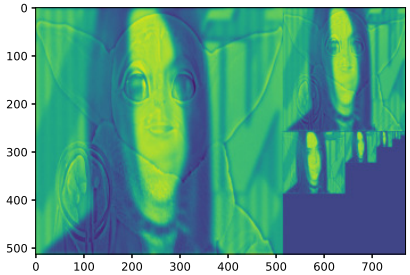


(c) Down sample of image.

Fig. 11. Fast Fourier Transformation results.



(a) Hybrid Image



(b) Gaussian image pyramid

Fig. 12. Hybrid Image with Image pyramid