Homework 1

Carla Andrade
Introduction to Signal and Image Processing

May 9, 2023

1 Linear Filtering

1.1 Write a function that returns a box filter of size [n $\times$ n].

the function boxfilter(n) simply create an array of size [nxn], which values sum up to 1.

1.2 Implement a 2D convolution function between an image [m $\times$ n] and a filter [k $\times$ l]

myconv2 function takes 2 arguments: a matrix of dim 1,2,or 3 (if it's an RGB image) and an 1D or 2D filter. The convolution necessities an inversion of the filter through axis x and, if it's a 2D filter,through axis y. Then the function add a padding to the images to have an image of size [(m+k-1)x(n+l-1)]. Then for each pixel (i,j,d)- depending of the number of dimension of the image-, it calculate the sum of filter* matrix[i:i+m, j:j+n,dim], where m,n are the size of the filter.

1.3 Create a box filter of size 11 and convolve this filter with your image. Show your result.

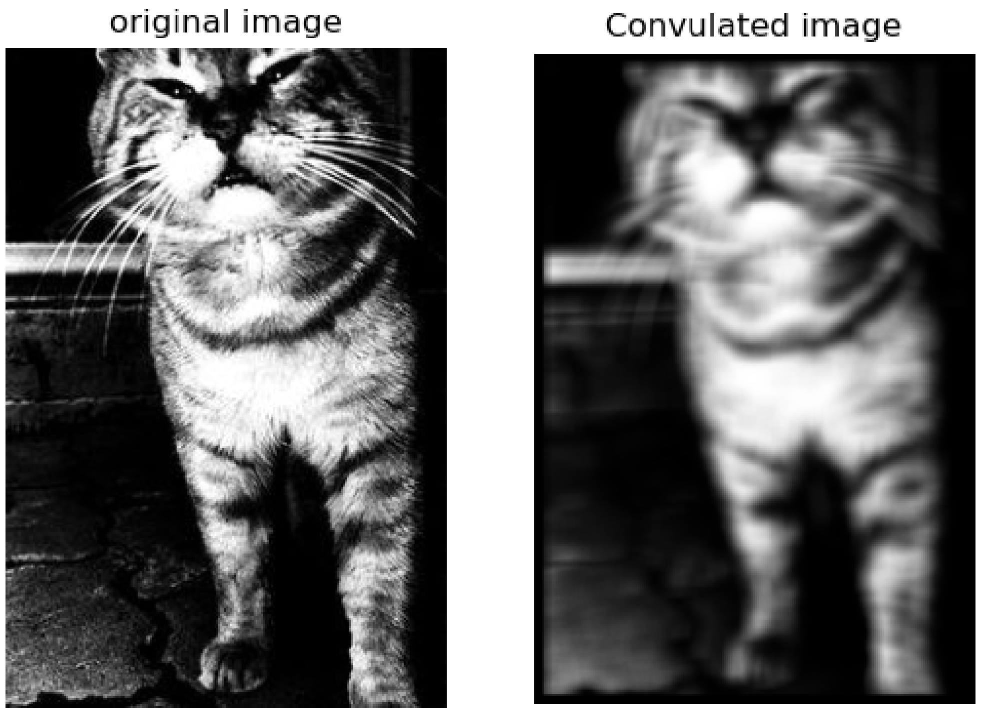

Figure 1: The first image represent the original images. On the right is the image after convolution it with an boxfilter of size 11.

1.4 Write a function that returns a 1D Gaussian filter for a given value of sigma.

gauss1d takes 2 arguments, sigma an numeric values and the $filter_{\text{length}}$. If the $filter_{\text{length}}$ is odd the function return a 1d Gaussian filter of the same size than $filter_{\text{length}}$, otherwise Gaussian filter size is of $filter_{\text{length}} + 1$. The function simply return:

$$gauss_{\text{length}} = exp((-x**2)/(2*(sigma**2)))$$

where x is a distance value from the center of the array. Ex: $x = [-1, 0, 1]$ for $filter_{\text{length}} = 3$

2 Finding edges

2.1 Begin by defining a derivative operator of your choosing.

the derivate chosen for dx is a Sobel operator and dy is it's transpose. These operators are then simply convoluted with myconv2 function to produce 2 edge filters.

2.2 Using dx and dy, construct an edge magnitude image.

I made 2 version of this function. In the first version, i used the Gaussian filter of scipy to reduce the noise of the image, then I calculated the deviate Ix, Iy by convoluting the image with dx and dy using myconv2. the gradient magnitude of the image is then calculated using:

$$np.sqrt(Ix**2 + Iy**2)$$

In the second version I used the edge filter of 2.1 to calculate Ix and Iy (and therefore my own Gaussian). The image obtained is blurrier than the one with scipy gaussian.

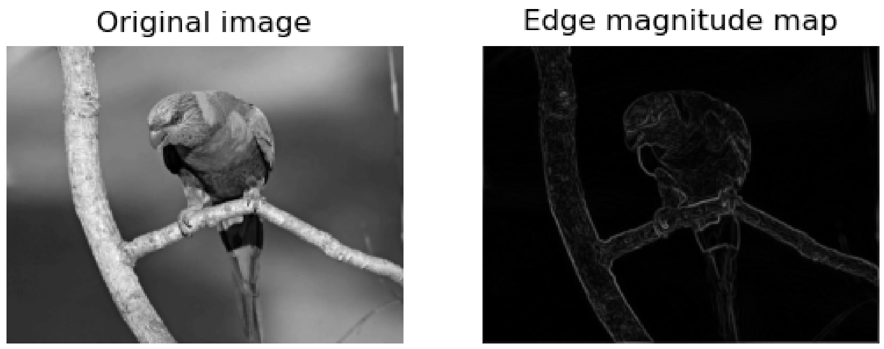

Figure 2: Bird Images . On the left is the original image. On the right is the image after applying edge magnitude function with the scipy Gaussian.

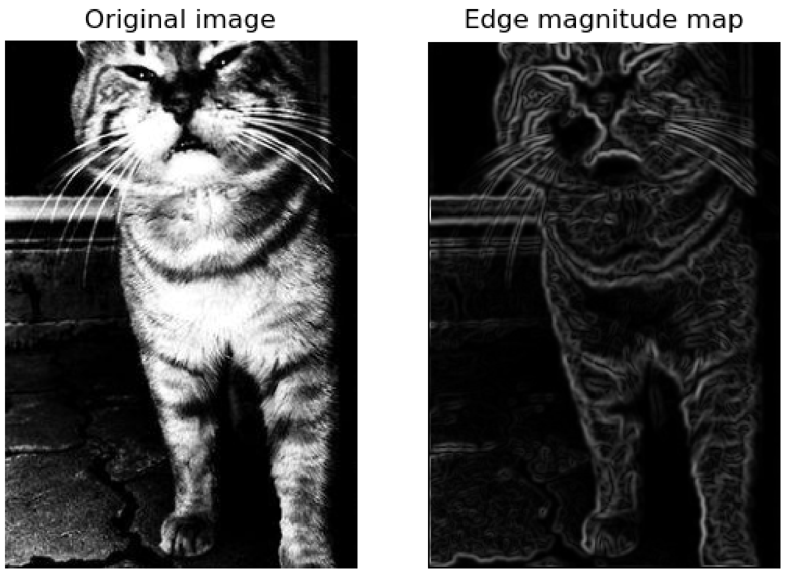

Figure 3: Cat Images . On the left is the original image. On the right is the image after applying edge magnitude function with the scipy Gaussian.

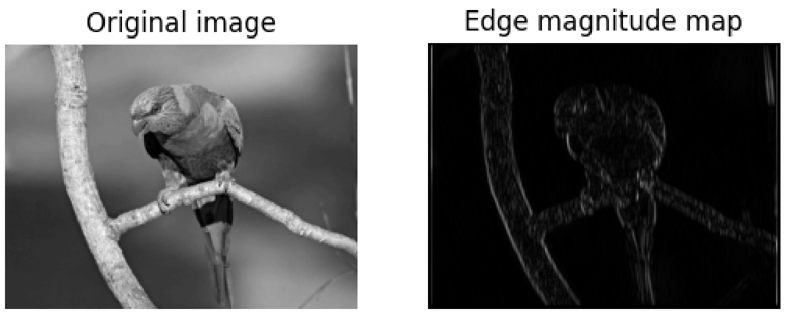

Figure 4: Bird Images . On the left is the original image. On the right is the image after applying edge magnitude function with my gaussian filter.

3 Corner detection

3.1 Write a function that computes the harris corner for each pixel in the image.

my Harris function takes 5 argument: image, window size, sigma, k and filter length. first it calculate the derivates Ix by convolating the image with a Sobel filter (and Sobel transpose for Iy). The product of derivates are computed by convovling the square of the derivates (Ixx, Iyy, Ixy) with a Gaussian $filter = 10$ and $sigma = 0.2$. the sum of product for each pixel of the image is then calculating using for loops. first a padding is added in the derivates product. then for each pixel i the sum of each pixel in the range of window size is calculated for each Ixx, Iyy, Ixy. these 3 matrix form the harris matrix:

$$H = np.array([[sxx, sxy], [sxy, syy]])$$

These harris matrix permit to calculate R with:

$$R = R[row, col] = np.linalg.det(H) - k * (np.trace(H) * *2)$$

3.2 Evaluate myharris on the image.
The image produced by myharris function is seen in figure 4, first image on the right.

3.3 Repeat with rotated image by 45 degrees.

See figure 5, second image on the left.

3.4 Repeat with down-scaled image by a factor of half

See figure 4, second image on the right.

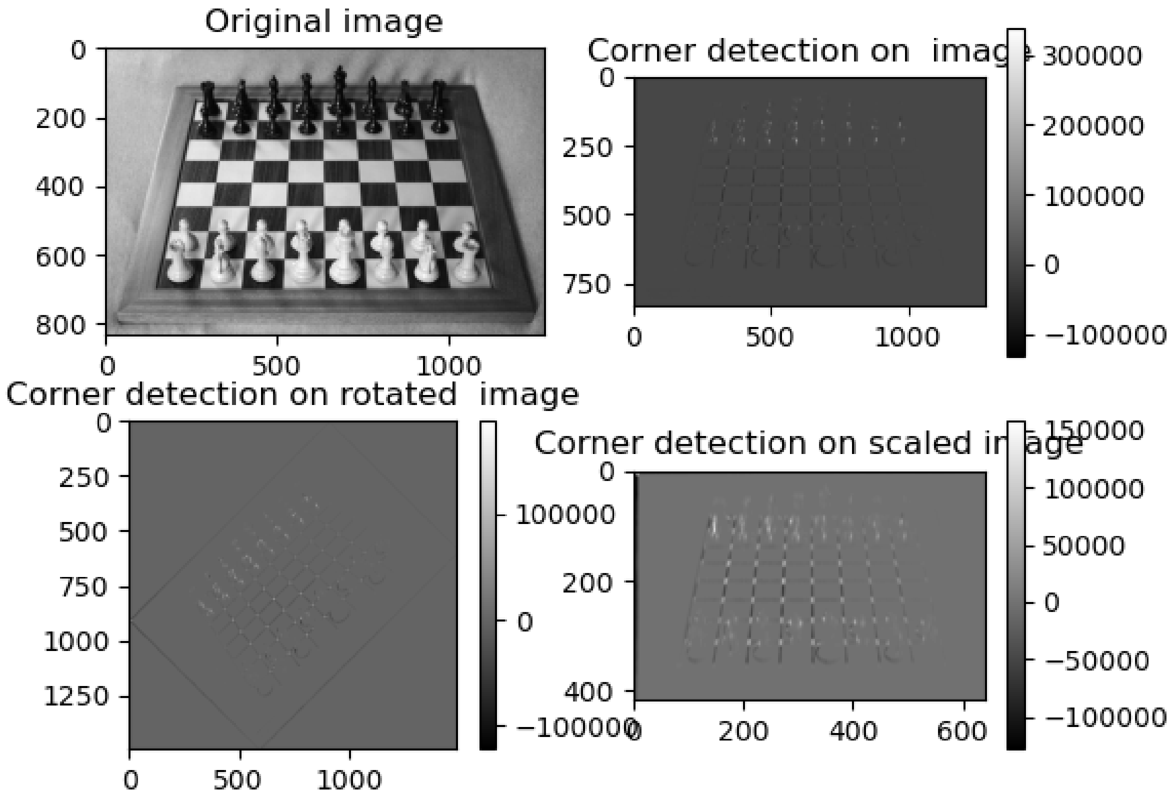

Figure 5: Chessboard Image. On the upper left is the original image. On the upper right is the image after applying myharris. The lower left is the image after 45 degree rotation. Finally, on the lower right is the down scaled image by half.

I also created a faster version. in this function the sum of product Ixx, Iyy, Ixy are calculated using the scipy gaussian filter. then the sum of products are calculated by convolving the sum of products with a kernel of 1s and of $windowSize = 5$. For this function $sigma = 2$

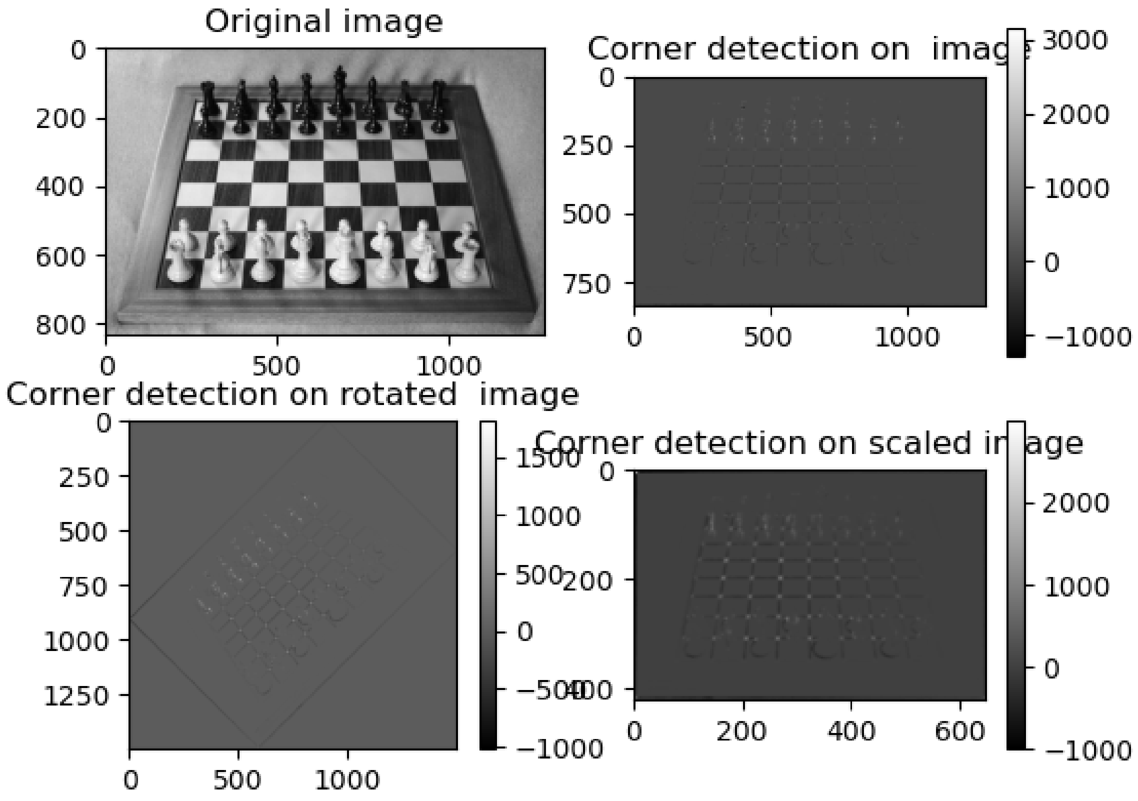

Figure 6: Chessboard Image. On the upper left is the original image. On the upper right is the image after applying the second myharris function. The lower left is the image after 45 degree rotation. Finally, on the lower right is the down scaled image by half.

To a better visualisation, in figure 7, The R value of myHarris (second version) are put to 0 if it's is bigger than 100 or 1 if lower than 100, to a better visualization of the corners.

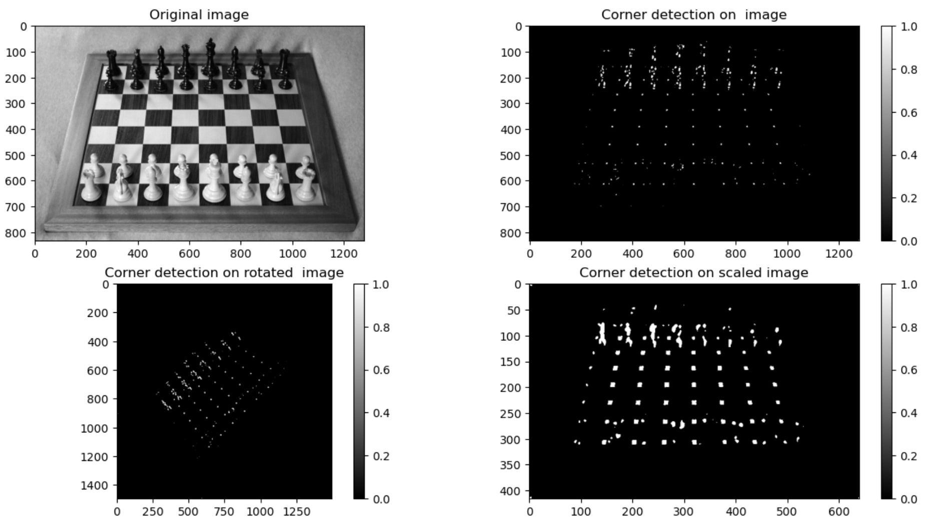

Figure 7: Chessboard Image. On the upper left is the original image. On the upper right is the image after applying the second myharris function. The lower left is the image after 45 degree rotation. Finally, on the lower right is the down scaled image by half. The R value of myHarris are put to 0 if it's is bigger than 100 or 1 if lower than 100, to a better visualization of the corners.

3.5 Looking at the results from 3.2, 3.3, and 3.4, what can we say about the properties of Harris corners? What is maintained? What is it invariant to? Why is that the case?

by comparing the images, we can observe that Harris corner is maintained when the image is rotated but when the image is scaled the corner appear bigger. Therefore Harris corner is rotation invariant. It's seems bigger on a scaled image simply because the image itself is smaller but is displayed on same size "screen" (the image is take the same space on the page than the original Harris corner) and therfore it's like the image has been zoomed in.