Harris Detector

We saw in Module 6.4 "Identifying corners" that we can perform convolutions to implement the Harris corner detector. Let's do it!

What to do

Write a Python program to do the following:

1. Given an input image, your program should first convert it to grayscale.
2. Calculate the partial derivative of the image in the x and y directions. You can do this by convolving the grayscale image with a filter that looks like

   $$\begin{bmatrix} -1 & 1 \end{bmatrix}$$

   for the x direction and

   $$\begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

   for the y direction. Let us call the results of these convolutions I_x and I_y, respectively.
3. Let $I_x^2 = I_x \times I_x$
4. Let $I_x I_y = I_x \times I_y$
5. Let $I_y^2 = I_y \times I_y$
6. Calculate A_W, B_W, C_W by convolving I_x^2, $I_x I_y$, I_y^2 with a box filter, such as

   $$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

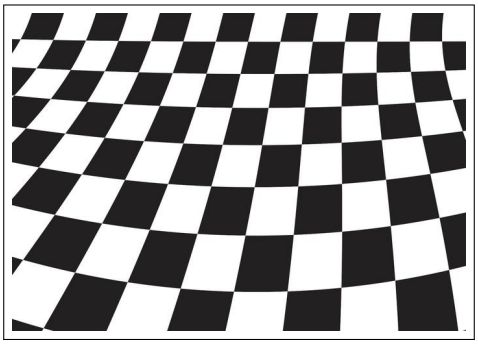
7. Compute the eigenvalues of

   $$\begin{bmatrix} A_W & B_W \\ B_W & C_W \end{bmatrix}$$

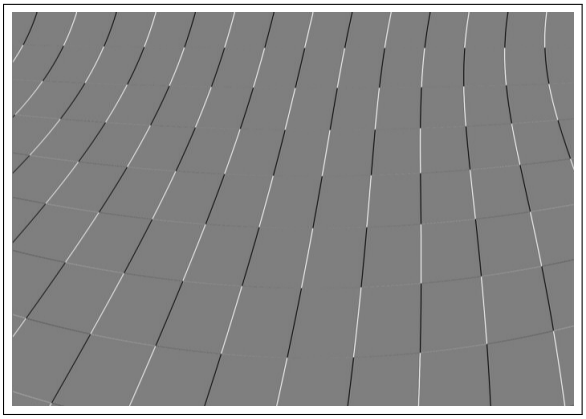
   at each pixel and only keep those where $min(\lambda_1, \lambda_2) > T$ where T is some threshold.
8. Visually mark the detected corners from the previous step on top of the original image. Save the output to a file.
9. **Optional** Apply non-maximal suppression on the results from the step 7 to avoid multiple detections.

Visual Walkthrough

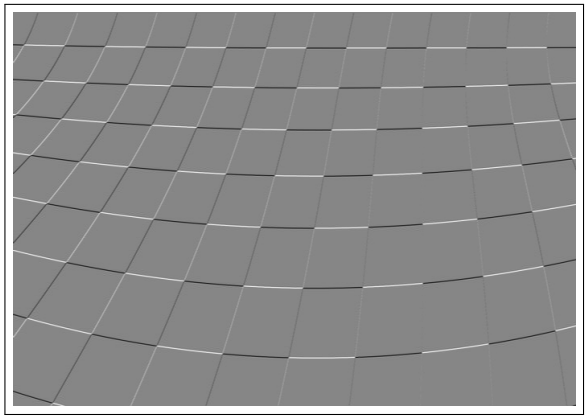
For example, we will use this image[1] as our input.



After step 2, we have:

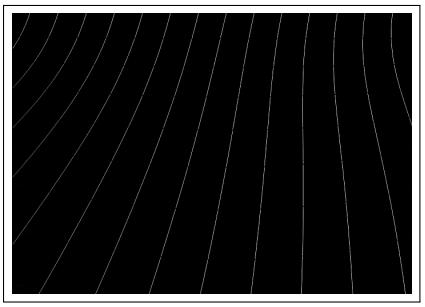


(a) I_x

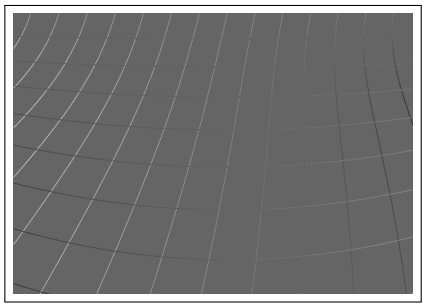


(b) I_y

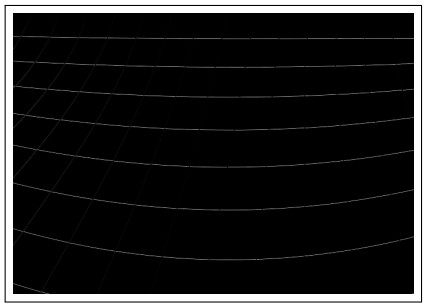
Steps 3, 4, and 5 will create:



(a) I_x^2



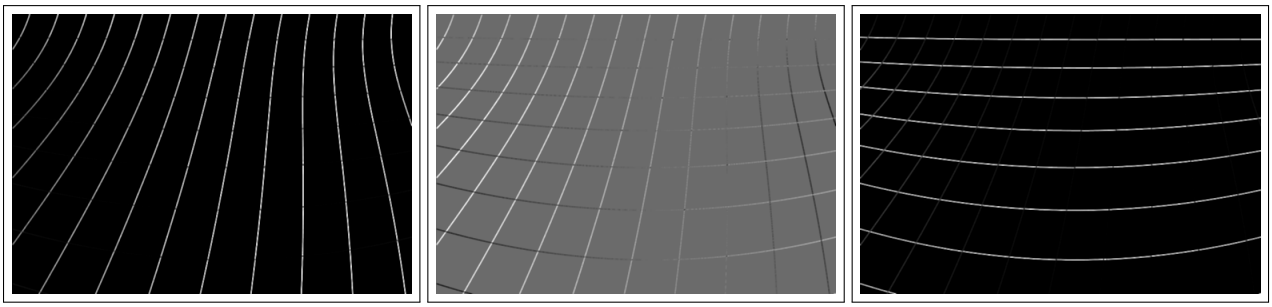
(b) $I_x I_y$



(c) I_y^2

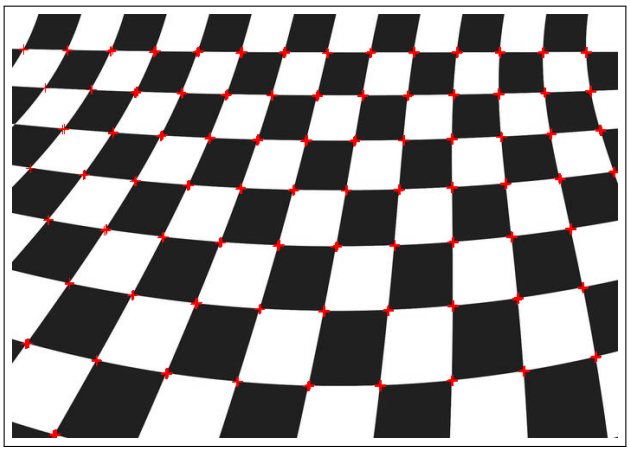
[1] https://www.vecteezy.com/vector-art/92734-checkerboard-pattern

Step 6 will create:



(a) A_W (b) B_W (c) C_W

And at step 8 we have:



Note that these visualizations look nice because the pixel values have been normalized. They do not represent the actual numerical values used during the calcuations.

Hints

Since the calculations involved in this algorithm will create numbers far beyond [0,255], you should only rely on Pillow's representation of images when loading and saving images. Other than that, you should treat the images as numpy arrays. Realistically speaking, in your convolution routine, you should work with numpy arrays of `dtype=numpy.int32` to save you from integer overflows.

Steps 2 and 6 are just convolutions.
Steps 3 through 5 are just matrix multiplications.
For Step 7, you can do for each pixel (x, y):

```
e_values, e_vectors = np.linalg.eig(np.array([[Aw[y][x],Bw[y][x]],
                                              [Bw[y][x],Cw[y][x]]]))
```

to obtain the eigenvalues of this special matrix, and then you can do the following to apply the threshold:

```
if min(e_values) > T:
...
```

To draw a red cross on top of the input image to indicating a detected corner, you can do something like:

```
im = Image.open("checker.jpg")
draw = ImageDraw.Draw(im)

# do the following for each coordinate (x,y) of a detected corner
draw.line(((x-5,y),(x+5,y)), fill=(255,0,0))
draw.line(((x,y-5),(x,y+5)), fill=(255,0,0))
```

More Hints

If you are using `numpy.int32` for your arrays, Pillow will probably complain when you try to save this to an image. One way to avoid this is to normalize the values:

```
def normalize(input_im):
    base = input_im.min()
    roof = input_im.max()
    diff = roof - base
    scale = diff/255

    input_im = input_im - base
    output = input_im/scale

    return np.uint8(output)
```

So, if you have an array `Ix`, you would use the above method in the following manner when saving it to a file:

```
Ix_image = Image.fromarray(normalize(Ix))
Ix_image.save("Ix.png")
```

What to turn in

Please submit a zip file containing your source code along with the input/output images. Please make sure that your code runs on the university's linux server prior to submission. If you used any external libraries other than Pillow, Numpy, or SciPy, please include a README file explaining why you did so.

When you submit your zip file, please put everything in a folder named `<username>_lab3` and zip that file.

Again, if you have any questions, please ask us on InScribe!