DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING, UCLA
EC ENGR 188: THE FOUNDATIONS OF COMPUTER VISION

INSTRUCTOR: Prof. Achuta Kadambi
TA: Rishi Upadhyay

NAME: Krish Patel
UID: 605 796 227

HOMEWORK 2

PROBLEM	TYPE	TOPIC	MAX. POINTS
1	Analytical	Filter Design	10
2	Analytical + Coding	Blob Detection	10
3	Coding	Corner Detection	10
4	Analytical	2D Transformation	10
5	Analytical	Interview Question	5

Motivation

In the previous homework, we have seen how to process images using convolutions and mine images for features such as edges using image gradients. In this homework, we will detect other types of useful features in images such as blobs and corners that can be useful for a variety of computer vision tasks. We then transition from detecting useful features in images to relating different images via 2D transformations.

The problem set consists of:

- analytical questions to solidify the concepts covered in the class; and
- coding questions to implement some of the algorithms described in class using Python.

Homework Layout

The homework consists of 5 problems in total, with subparts for each problem. There are 2 types of problems in this homework - analytical and coding. We encourage you to answer all the problems using the Overleaf document; however, handwritten solutions will also be accepted. The overleaf link is here: `https://www.overleaf.com/read/ggqwqzsgmcjn#d2fb52`. The coding jupyter notebook is: `https://colab.research.google.com/drive/1Tp_8YKNiZ7HlvFE4OyT9Z84t3DzwT1ac?usp=sharing`

Submission

Submission will be done via Gradescope. You will need to submit three things: (1) this PDF with answers filled out, (2) Your .ipynb file, (3) a PDF printout of your .ipynb file with all cells executed. You do not need to create a folder, you will be able to upload all three files directly to gradescope.

1 Filter Design (10.0 points)

In the class you were taught the Central Difference/Laplacian filter for detecting the edges in an image (by computing the gradients along the horizontal and vertical directions). In the discussion we derived those filters by approximating the first (for gradient) and second derivative (for Laplacian) of a univariate function $f(x)$ using the Taylor series expansion of $f(x+h)$ and $f(x-h)$, where h is a small perturbation around x with $h=1$ for the discrete case. These filters only consider the adjacent neighbours of the current pixel. In this question you will derive a high-order approximation for the two filters, such that the filters will use 2 adjacent neighbours. To summarize, you will be deriving a higher order approximation to the first/second derivative of $f(x)$. We will use $f^{(1)}(x), f^{(2)}(x)$ to denote the first and second derivative respectively.

1.1 Compute $f(x+h)$ (1.0 points)

Write the Taylor series approximation for $f(x+h)$ around $f(x)$, such that the approximation error tends to 0 as h^5, i.e. consider only the first 5 terms in the Taylor series approximation. Use $f^{(1)}(x)$ for the first derivative, $f^{(2)}(x)$ for the second derivative and so on.

The Taylor series approximation for $f(x+h)$ around $f(x)$ up to the fifth term is:

$$f(x+h) \approx f(x) + f^{(1)}(x)h + \frac{f^{(2)}(x)}{2!}h^2 + \frac{f^{(3)}(x)}{3!}h^3 + \frac{f^{(4)}(x)}{4!}h^4 + \frac{f^{(5)}(x)}{5!}h^5$$

1.2 Compute $f(x-h)$ (1.0 points)

Similarly, write the Taylor series approximation for $f(x-h)$ around $f(x)$, such that the approximation error tends to 0 as h^5.

The Taylor series approximation for $f(x-h)$ around $f(x)$ up to the fifth term is similar to the above formula, but with h replaced with negative h

$$f(x-h) \approx f(x) - f^{(1)}(x)h + \frac{f^{(2)}(x)}{2!}h^2 - \frac{f^{(3)}(x)}{3!}h^3 + \frac{f^{(4)}(x)}{4!}h^4 - \frac{f^{(5)}(x)}{5!}h^5$$

1.3 Compute $f(x+h) - f(x-h)$ (1.0 points)

Using the expressions you obtained in the previous two parts, compute the expression for $f(x+h) - f(x-h)$. You may assume that $\mathcal{O}(h^5)$ tends to 0, so that you can neglect the term. You will be using this result to compute a high-order approximation to $f^1(x)$.

The opposite sign terms would cancel each other out, giving the following formula

$$f(x+h) - f(x-h) \approx 2f^{(1)}(x)h + \frac{f^{(3)}(x)}{3}h^3$$

1.4 Unknowns at each pixel (1.0 points)

Let's assume that you have access to $f(x)$, which is a 1D signal (or equivalently one row in an image). This means that you can easily obtain the values for $f(x)$, $f(x \pm h)$, $f(x \pm 2h)$ and so on (assuming appropriate zero padding for start and end values). However, for each pixel x, if you only consider using its adjacent neighbours ($\pm h$), then the equation in the previous part has 2 unknowns. What are the two unknowns? *Note:* $h = 1$ for the discrete case, so h is not an unknown. But don't substitute $h = 1$ as of now; we will substitute it later.

Based on the equation we got above, the two unknowns in this equation are the first oder derivative with respect to x, and the third order derivative. We would need to calculate this(f'(x) can be calculated using the gradient in X filter)

1.5 Compute $f^{(1)}(x)$ (1.0 points)

From the previous part, you know that for each pixel x, if we only consider $x \pm h$, then we have 1 equation and 2 variables (underdetermined system). To mitigate this issue, we consider two adjacent neighbours for each pixel ($x \pm 2h$) in addition to $x \pm h$. Replace h with $2h$ in the previous equation and you will get another equation for that pixel. So now, for each pixel, you have 2 equations and 2 variables. One equation should have $f(x \pm h)$ and the other should have $f(x \pm 2h)$. Using these two equations, solve for $f^{(1)}(x)$.

1.6 Convolution Kernel (1.0 points)

What is the convolution kernel corresponding to the new filter for $f^{(1)}(x)$? Substitute $h = 1$ for the discrete case. This filter is now a higher order central-difference filter which can be used to compute the gradients/edges.

Substituting $h = 1$ in the equation above gives us the following kernel:

$$\begin{bmatrix} 1 & -8 & 0 & 8 & -1 \end{bmatrix}$$

This kernel is not normalized, as seen above, and would result in convolutions greater than the allowed pixel value.

1.7 Laplacian Filter (1.0 points)

We will repeat the same exercise as the previous parts; however, now we compute a higher-order approximation to the Laplacian filter. Similar to 1.3, compute the expression for $f(x+h)+f(x-h)$.

Using a Laplacian filter:

$$f(x+h) \approx f(x)+f^{(1)}(x)h+\frac{f^{(2)}(x)}{2!}h^2+\frac{f^{(3)}(x)}{3!}h^3+\frac{f^{(4)}(x)}{4!}h^4+\frac{f^{(5)}(x)}{5!}h^5$$

$$f(x-h) \approx f(x)-f^{(1)}(x)h+\frac{f^{(2)}(x)}{2!}h^2-\frac{f^{(3)}(x)}{3!}h^3+\frac{f^{(4)}(x)}{4!}h^4-\frac{f^{(5)}(x)}{5!}h^5$$

thus $f(x+h)+f(x-h)=2f(x)+h^2f^2(x)+\frac{h^4}{12}f^4(x)$

1.8 Unknowns for the Laplacian (1.0 points)

Similar to 1.4, what are the two unknowns for each pixel in the expression from the previous part?

The unknowns in this scenario are f”(x) and f””(x)

1.9 Compute $f^{(2)}(x)$ (1.0 points)

Similar to 1.5, use $f(x\pm 2h)$ to solve for $f^{(2)}(x)$. Write the expression for $f^{(2)}(x)$.

$$f(x+2h)+f(x-2h)=2f(x)+4h^2f^2(x)+\frac{16h^4}{12}f^4(x)$$

$$f(x+h)+f(x-h)=2f(x)+h^2f^2(x)+\frac{h^4}{12}f^4(x)$$

$$\Rightarrow \frac{h^4}{12}f^4(x)=f(x+h)+f(x-h)-2f(x)-h^2f^2(x)$$

$$\text{Therefore } f^2(x) \approx \frac{-f(x-2h)+16f(x-h)-30f(x)+16f(x+h)-f(x+2h)}{12h^2}$$

1.10 Convolution Kernel (1.0 points)

What is the corresponding convolution for the filter you derived in the previous part? Use $h=1$ for the discrete case.

Using h=1, these are the values we get for the convolution filter/matrix

$$\begin{bmatrix} 1 & -4 & 6 & -4 & 1 \end{bmatrix}$$

This is not normalized, and we have to divide it by the L1 norm to get normalized

2 Blob Detection (10.0 points)

In this question, you will be using the Laplacian of Gaussian (LoG) filter to detect blobs in an image. Let's consider a 2D Gaussian $G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-(\frac{x^2+y^2}{2\sigma^2})}$. Remember that we need to smooth the image using a Gaussian filter before computing the Laplacian to prevent noise amplification. However, instead of computing the Laplacian after filtering the image with a Gaussian kernel, we can directly filter the image with the Laplacian of Gaussian filter.

2.1 Compute $\frac{\partial^2 G_\sigma(x,y)}{\partial x^2}$ (1.0 points)

Write the expression for $\frac{\partial^2 G_\sigma(x,y)}{\partial x^2}$.

Computing partial second derivative with respect to x:

$$\frac{\partial^2 G_\sigma(x,y)}{\partial x^2} = \frac{1}{2\pi\sigma^2}\left(-\frac{1}{\sigma^2} + \frac{x^2}{\sigma^4}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$

2.2 Compute $\frac{\partial^2 G_\sigma(x,y)}{\partial y^2}$ (1.0 points)

Write the expression for $\frac{\partial^2 G_\sigma(x,y)}{\partial y^2}$.

Similar to above:

$$\frac{\partial^2 G_\sigma(x,y)}{\partial y^2} = \frac{1}{2\pi\sigma^2}\left(-\frac{1}{\sigma^2} + \frac{y^2}{\sigma^4}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$

2.3 Laplacian of a 2D Gaussian (1.0 points)

Using the results from the previous parts, write the expression for the Laplacian of a 2D Gaussian, $L(x,y)$.

Laplacian of a 2D Gaussian would be the sum of both the partial derivatives which would be equal to:

$$\frac{1}{2\pi\sigma^2}\left(-\frac{1}{\sigma^2} + \frac{x^2}{\sigma^4}\right) e^{-\frac{x^2+y^2}{2\sigma^2}} + \frac{1}{2\pi\sigma^2}\left(-\frac{1}{\sigma^2} + \frac{y^2}{\sigma^4}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$

2.4 Scale-Normalization (1.0 points)

In class, we studied that it is important to normalize the scale of $L(x,y)$ before using it for blob detection. What is the normalization factor? Provide justification.

The normalization factor in this case ensures that the integral of the LoG would be equal to zero. WIth respect to the equations above, the normalization factor is $\frac{1}{\sigma^2}$.

2.5 LoG Filter (1.0 points)

(See the Jupyter notebook) Using the expression for $L(x,y)$ and the scale normalization, write a Python function which will compute the LoG Filter.

```python
def log_filter(size: int, sigma: float):
    if size % 2 == 0:
        size += 1
    if size is None:
        size = 6 * sigma
    if size % 2 == 0:
        size += 1
    x, y = np.meshgrid(np.arange(-size//2+1,size//2+1),
            np.arange(-size//2+1,size//2+1))
    kernel = -(1/(np.pi * sigma**4)) *
             (1-((x**2 + y**2) / (2 * sigma**2))) *
            np.exp(-(x**2 + y**2) / (2 * sigma**2))
    kernel = kernel/np.sum(np.abs(kernel))
    return kernel
```

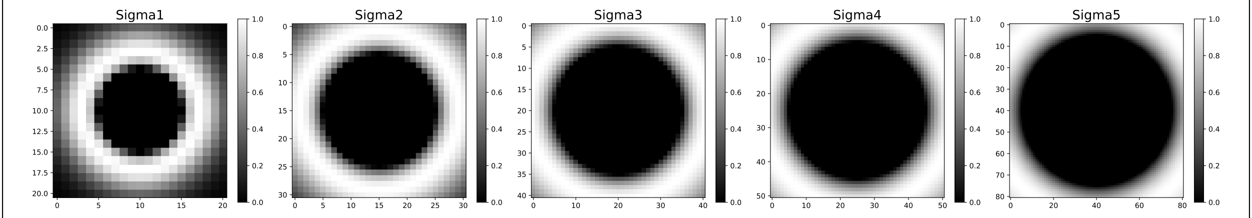
2.6 σ values (1.0 points)

(See the Jupyter notebook) What are the 5 sigma values which give the maximum response? To visualize the response, you can use the colormap in the Jupyter notebook.

The 5 sigma values that give the best response are 3.54, 7.07,10.61, 14.14, 24.75. Note: These are approximations for the values up to two decimal places

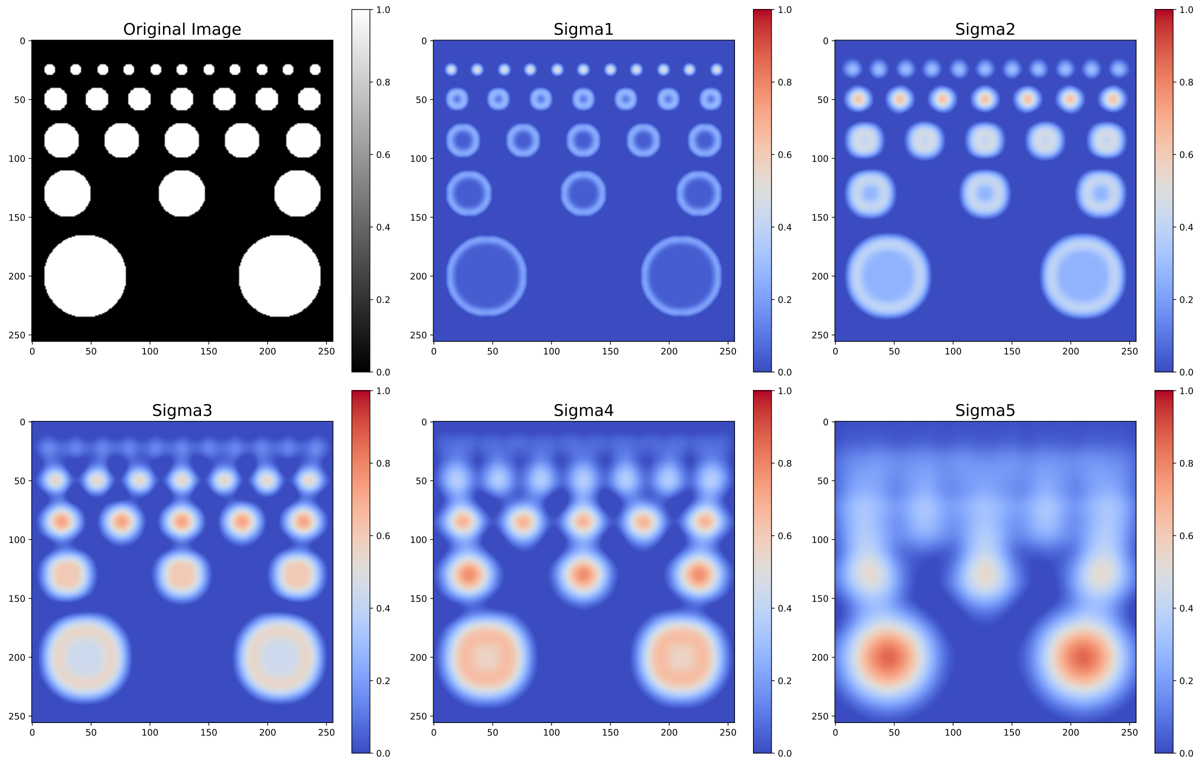
2.7 Visualize LoG Filter (1.0 points)

(See the Jupyter notebook) In this sub-part you will visualize the LoG filter. Copy the saved image from the Jupyter notebook here.



2.8 Visualize the blob detection results (3.0 points)

(See the Jupyter notebook) In this sub-part you will visualize the blob detection results. Copy the saved image from the Jupyter notebook here.



3 Corner Detection (10.0 points)

In this question, you will be implementing the Harris corner detector. As discussed in class, corners generally serve as useful features.

3.1 Computing Image Gradients Using Sobel Filter (1.0 points)

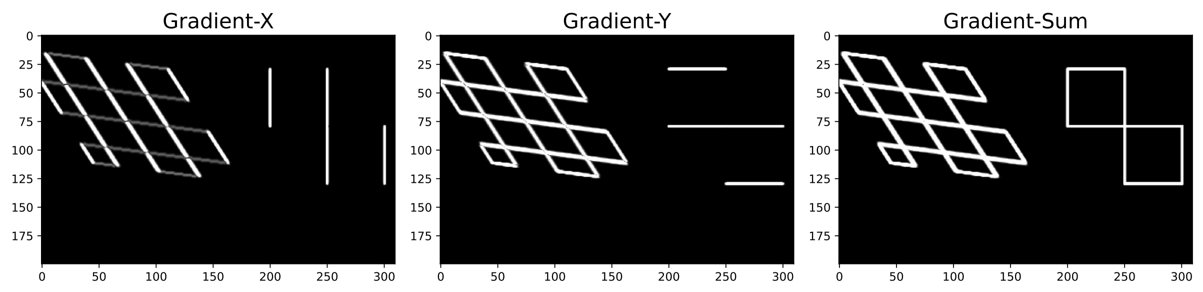
(See the Jupyter notebook). In this sub-part, you will write a function that computes image gradients using the Sobel filter. Make sure that your code is within the bounding box.

```
def compute_image_gradient(image: np.array):
    sobel_x = np.array([[1, 0, -1], [2, 0, -2], [1, 0, -1]])
    sobel_y = np.array([[1, 2, 1], [0, 0, 0], [-1, -2, -1]])
    grad_x = conv2D(image, sobel_x)
    grad_y = conv2D(image, sobel_y)
    return grad_x, grad_y
```

This uses the filters given above

3.2 Visualizing the Image Gradients (1.0 points)

(See the Jupyter notebook). In this sub-part, you will visualize the image gradients. Copy the saved image from the Jupyter notebook here.



3.3 Computing the Covariance Matrix (1.0 points)

(See the Jupyter notebook). In this sub-part, you will write a function that computes the covariance matrix of the image gradients. Make sure that your code is within the bounding box.

```
def grad_covariance(image: np.array, size: int):
    avg_filter = average_filter(size)
    grad_x, grad_y = compute_image_gradient(image)
    grad_x_sq = grad_x**2
    grad_y_sq = grad_y**2
    grad_xy = grad_x*grad_y
    grad_x_sq_avg = conv2D(grad_x_sq,avg_filter) # convolve gradient
     ↪   sqrd with average filter
    grad_y_sq_avg = conv2D(grad_y_sq,avg_filter)
    grad_xy_avg = conv2D(grad_xy,avg_filter)
    return grad_x_sq_avg, grad_y_sq_avg, grad_xy_avg
```

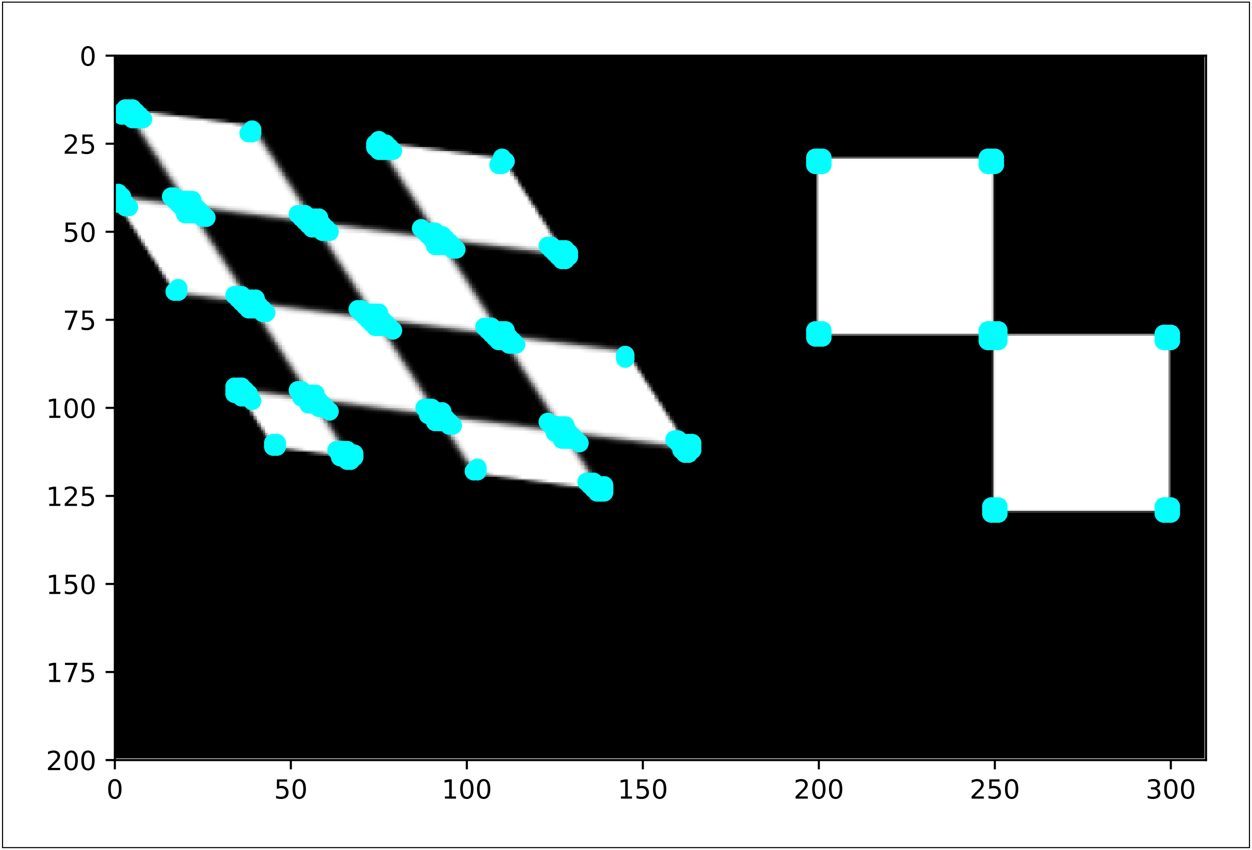
3.4 Harris Corner Response (1.0 points)

(See the Jupyter notebook). In this sub-part, you will write a function that computes the Harris response function. Make sure that your code is within the bounding box.

```
def harris_response(image: np.array, k: float, size: int):
    grad_x_sq_avg,grad_y_sq_avg,grad_xy_avg = grad_covariance(image,
     ↪   size)
    det = grad_x_sq_avg * grad_y_sq_avg-grad_xy_avg**2 #detmerinant
    trace = grad_x_sq_avg+grad_y_sq_avg #trace operator
    return det-k*trace**2
```

3.5 Visualizing the Harris Corner Detector (1.0 points)

(See the Jupyter notebook). In this sub-part, you will visualize the Harris corner detections. Copy the saved image from the Jupyter notebook here.



3.6 Thresholding the Harris Response (1.0 points)

To remove duplicate detections, you will write a function that applies non-maximum suppression to the Harris corner detections. To make writing this function easier, you will implement it in various parts.

(See the Jupyter notebook). In this sub-part, you will implement the first step of non-maximum suppression: thresholding the Harris response to obtain candidate corner detections. Make sure that your code is within the bounding box.

```
def threshold_harris_response(harris_response: np.array, threshold:
 ↪  float):
    return harris_response > threshold
```

3.7 Sorting Candidate Detections (1.0 points)

(See the Jupyter notebook). In this sub-part, you will sort the candidate detections by maximum Harris response value. Make sure that your code is within the bounding box.

```
def sort_detections(candidate_detections: np.array, harris_response:
 ↪  np.array):
    sorted_idces = np.argsort(harris_response.flatten())[::-1]
    unravelled_idces = np.unravel_index(sorted_idces,
     ↪  harris_response.shape)
    sorted_detections = np.column_stack(unravelled_idces)
    index_to_rank = {}
    for rank, idx in enumerate(zip(*sorted_detections)):
        index_to_rank[idx] = rank
    sorted_candidate_detections = sorted(candidate_detections, key=lambda
     ↪  x: index_to_rank.get(tuple(x),len(candidate_detections)))
    return np.array(sorted_candidate_detections)
```

3.8 Suppressing Non-max Detections (1.0 points)

(See the Jupyter notebook). In this sub-part, you will implement the final step of non-maximum suppression: removing corner detections that are not local maxima. Make sure that your code is within the bounding box.

```
def local_max(sorted_detections: np.array, distance: float):
    maxima = []
    for i in range(len(sorted_detections)):
        for j in range(i+1, len(sorted_detections)):
            if l2_distance(sorted_detections[i],sorted_detections[j])<
             ↪  distance:
                break
        else:
            maxima.append(sorted_detections[i])
    return np.array(maxima)
```

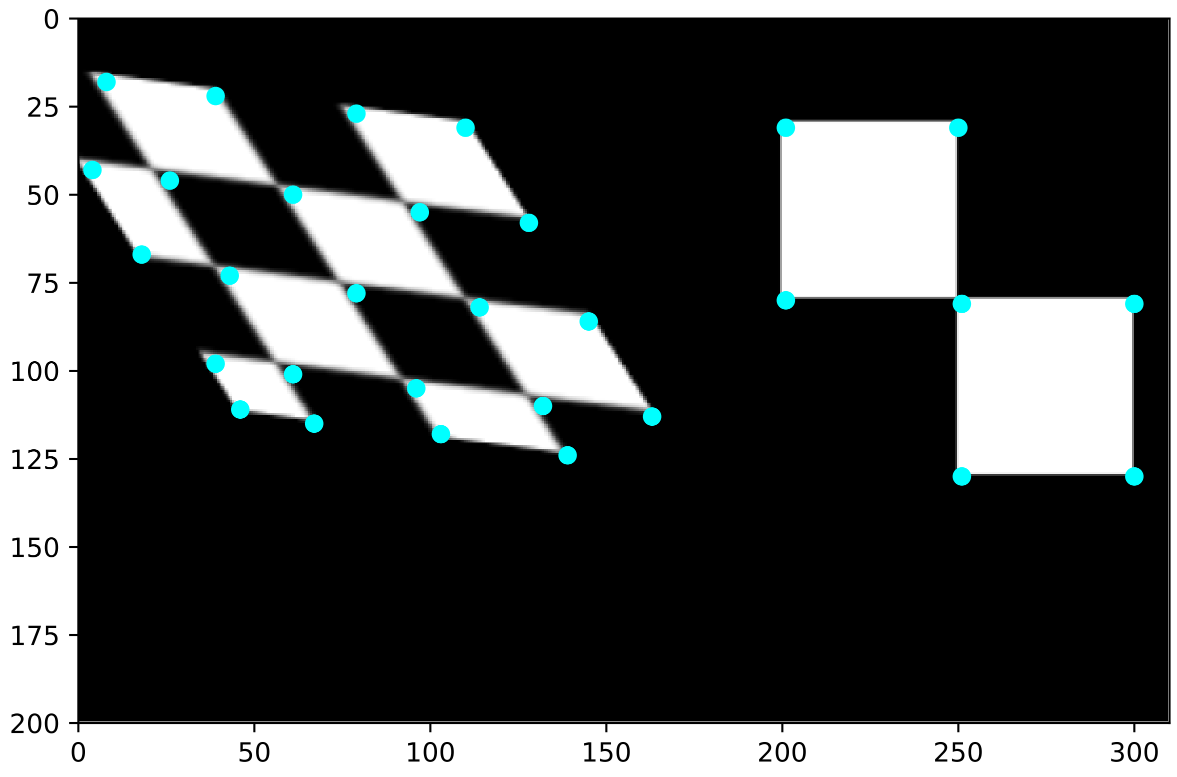
3.9 Non-Maximum Suppression: Putting it all together (1.0 points)

(See the Jupyter notebook). In this sub-part, you will write a function that performs non-maximum suppression on the Harris corner response. Make sure that your code is within the bounding box.

```
def non_max_suppression(harris_response: np.array, distance: float,
 ↪  threshold: float):
    candidate_detections = np.argwhere(harris_response>threshold)
    sorted_detections = sort_detections(candidate_detections,
            harris_response)
    maximaa =local_max(sorted_detections,distance)
    return maximaa
```

3.10 Visualizing Harris Corner Detections + Non-maximum Suppression (1.0 points)

(See the Jupyter notebook). In this sub-part, you will visualize the Harris corner detections after non-maximum suppression has been applied. Copy the saved image from the Jupyter notebook here. Duplicate corner detections should now be removed.



4 2D Transformation (10.0 points)

In this question, you will be identifying different 2D transformations. You will be given a set of feature points x and the corresponding transformed points x'. Given these two set of points, you have to identify the 2D transformation. For the first 5 sub-parts, there is only one transformation (translation, scaling, rotation, shearing). For the next parts, there may be more than one transformation. While justifying your answer for each part you should also write the 3×3 transformation matrix M.

4.1 Example 1 (1.0 points)

$x = \{(1,1),(2,1),(2,2),(1,2)\}$ and $x' = \{(2,2),(3,2),(3,3),(2,3)\}$. Identify the transformation and justify:

For these data points, it can be inferred by looking at the transformation that this is a translation. For each point in x, it is translated as one point in the x direction and 1 unit in the y direction. Thus to carry out this translation, we need to translate each point by (1,1).

4.2 Example 2 (1.0 points)

$x = \{(1,1),(2,1),(2,2),(1,2)\}$ and $x' = \{(0,\sqrt{2}),(\sqrt{2}-\frac{1}{\sqrt{2}},\sqrt{2}+\frac{1}{\sqrt{2}}),(0,2\sqrt{2}),(\frac{1}{\sqrt{2}}-\sqrt{2},\sqrt{2}+\frac{1}{\sqrt{2}})\}$. Identify the transformation and justify:

For this question, I used a transformation matrix with the form

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

I concluded that there was no translation when I looked at the points (1,1) and (2,2), which were essentially scales of each other after the translations(and also double confirmed using homeogenous coordinates methodology used below, and from the other two points. Plugging in the values for (1,1), I came up with the formula: a+b = 0, c+d = $\sqrt{2}$. Using the other data point (2,1), I came up with the equations a+2b = $\sqrt{2}-\frac{1}{\sqrt{2}}$ and c+2d =$\sqrt{2}+\frac{1}{\sqrt{2}}$ Thus, from these equations(derived from the data points) this I concluded that a is $\sqrt{2}-\frac{1}{\sqrt{2}}$, b = $-\sqrt{2}+\frac{1}{\sqrt{2}}$, c = $\frac{1}{\sqrt{2}}$, and d = $\sqrt{2}-\frac{1}{\sqrt{2}}$

$$A = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4.3 Example 3 (1.0 points)

$x = \{(1,1),(2,1),(2,2),(1,2)\}$ and $x' = \{(-1,1),(-1,2),(-2,2),(-2,1)\}$. Identify the transformation and justify:

Mx is order of operations matrix used:

$$\begin{bmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

Equations from the first point:

$$a+b+e=-1$$
$$c+d+f=1$$

Equations from the second point:

$$2a+b+e=-1$$
$$2c+d+f=2$$

Equations from the third point:

$$2a+2b+e=-2$$
$$2c+2d+f=2$$

Equations from the fourth point:

$$a+2b+e=-2$$
$$c+2d+f=1$$

using the equations above, a = 0, c = 1, b = -1, d = 0, e = 0 and f = 0 Thus, matrix looks like this

matrix used:

$$\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This is a rotation matrix, and it performs a rotation by 90 degrees in the CC direction

4.4 Example 4 (1.0 points)

$x = \{(1,1),(2,1),(2,2),(1,2)\}$ and $x' = \{(3,5),(6,5),(6,10),(3,10)\}$. Identify the transformation and justify:

Mx is the order of operations. Matrix used:

$$\begin{bmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

textbfEquations from the first point:

$$a+b+e=3$$
$$c+d+f=5$$

Equations from the second point:

$$2a+b+e=6$$
$$2c+d+f=5$$

Equations from the third point:

$$2a+2b+e=6$$
$$2c+2d+f=10$$

Equations from the fourth point:

$$a+2b+e=3$$
$$c+2d+f=10$$

From the equations above, a = 3, c = 0, b = 0, d = 5, e = 0 and f = 0 Thus this matrix is a scaling matrix(diagonal, scales 3 times in X and 5 times in Y matrix used:

$$\begin{bmatrix} 3 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4.5 Example 5 (1.0 points)

$x = \{(1,1),(2,1),(2,2),(1,2)\}$ and $x' = \{(4,6),(5,11),(8,12),(7,7)\}$. Identify the transformation and justify:

Mx is the order of operations. Matrix used:

$$\begin{bmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

textbfEquations from the first point:

$$a+b+e=4$$
$$c+d+f=6$$

Equations from the second point:

$$2a+b+e=5$$
$$2c+d+f=11$$

Equations from the third point:

$$2a+2b+e=8$$
$$2c+2d+f=12$$

Equations from the fourth point:

$$a+2b+e=7$$
$$c+2d+f=7$$

From the equations above, a = 1, c = 5, b = 3, d = 1, e = 0 and f = 0 Thus this matrix is a shearing matrix(shear 3 times in X based on Y and 5 times in Y based on the X value) matrix used:

$$\begin{bmatrix} 1 & 3 & 0 \\ 5 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4.6 Example 6 (1.0 points)

$x = \{(1,1),(2,1),(2,2),(1,2)\}$ and $x' = \{(0,2),(0,3),(-1,3),(-1,2)\}$. Identify the two transformations and their order and justify:

Mx is the order of operations. Matrix used:

$$\begin{bmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

Equations from the first point:

$$a+b+e=0$$
$$c+d+f=2$$

Equations from the second point:

$$2a+b+e=0$$
$$2c+d+f=3$$

Equations from the third point:

$$2a+2b+e=-1$$
$$2c+2d+f=3$$

Equations from the fourth point:

$$a+2b+e=-1$$
$$c+2d+f=2$$

From the equations above, a = 0, c = 1, b = -1, d = 0, e = 0 and f = 1 Thus this matrix is a combination of a rotation matrix (90 degrees CC) and then a translation by 1 unit in y. matrix used:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

4.7 Example 7 (1.0 points)

$x = \{(1,1),(2,1),(2,2),(1,2)\}$ and $x' = \{(-2,2),(-2,3),(-3,3),(-3,2)\}$. Identify the two transformations and their order and justify:

Mx is the order of operations. Matrix used:

$$\begin{bmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

Equations from the first point:

$$a+b+e=-2$$
$$c+d+f=2$$

Equations from the second point:

$$2a+b+e=-2$$
$$2c+d+f=3$$

Equations from the third point:

$$2a+2b+e=-3$$
$$2c+2d+f=3$$

Equations from the fourth point:

$$a+2b+e=-3$$
$$c+2d+f=2$$

From the equations above, a = 0, c = 1, b = -1, d = 0, e = -1 and f = 1 Thus, this matrix is a combination of a rotation (90 degrees CC) and a translation of -1 in X coord and 1 unit in Y coord matrix used:

$$\begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

4.8 Example 8 (1.0 points)

$x=\{(1,1),(2,1),(2,2),(1,2)\}$ and $x'=\{(4,6),(7,6),(7,11),(4,11)\}$. Identify the two transformations and their order and justify:

Mx is the order of operations. Matrix used:

$$\begin{bmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

Equations from the first point:

$$a+b+e=4$$
$$c+d+f=6$$

Equations from the second point:

$$2a+b+e=7$$
$$2c+d+f=6$$

Equations from the third point:

$$2a+2b+e=7$$
$$2c+2d+f=11$$

Equations from the fourth point:

$$a+2b+e=4$$
$$c+2d+f=11$$

From the equations above, a = 3, c = 0, b = 0, d = 5, e = -1 and f = 1 Thus this matrix is a shearing matrix(shear 3 times in X based on Y and 5 times in Y based on the X value) matrix. After this, there is a translation of -1 in X and +1 in Y. matrix used:

$$\begin{bmatrix} 3 & 0 & -1 \\ 0 & 5 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

4.9 Example 9 (1.0 points)

$x = \{(1,1),(2,1),(2,2),(1,2)\}$ and $x' = \{(-5,3),(-5,6),(-10,6),(-10,3)\}$. Identify the two transformations and their order and justify:

Mx is the order of operations. Matrix used:

$$\begin{bmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

Equations from the first point:

$$a+b+e=-5$$
$$c+d+f=3$$

Equations from the second point:

$$2a+b+e=-5$$
$$2c+d+f=6$$

Equations from the third point:

$$2a+2b+e=-10$$
$$2c+2d+f=6$$

Equations from the fourth point:

$$a+2b+e=-10$$
$$c+2d+f=3$$

From the equations above, a = 0, c = 3, b = -5, d = 0, e = 0 and f = 0 Thus, this is a shearing matrix, (Y coordinates only defined by X, and vice versa). Shears -5 times in X(based on the y value and shears 3 times in Y (based on the X value) matrix used:

$$\begin{bmatrix} 0 & -5 & 0 \\ 3 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4.10 Example 10 (1.0 points)

$x = \{(1,1),(2,1),(2,2),(1,2)\}$ and $x' = \{(-6,4),(-11,5),(-12,8),(-7,7)\}$. Identify the two transformations and their order and justify:

Mx is the order of operations. Matrix used:

$$\begin{bmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

Equations from the first point:

$$a+b+e=-6$$
$$c+d+f=4$$

Equations from the second point:

$$2a+b+e=-11$$
$$2c+d+f=5$$

Equations from the third point:

$$2a+2b+e=-12$$
$$2c+2d+f=8$$

Equations from the fourth point:

$$a+2b+e=-7$$
$$c+2d+f=7$$

From the equations above, a = -5, c = 1, b = -1, d = 3, e = 0 and f = 0 Thus, this is a combination of two transformations, first a rotation matrix (90 degrees CC) and then a scaling of -5 in the X direction, and 3 in the Y direction. matrix used:

$$\begin{bmatrix} -5 & -1 & 0 \\ 1 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5 Interview Question (5.0 points)

Object detection is a technique which allows computers to identify and localize objects in images/videos. Let us consider that we have an object detection system that finds cars in a given image. The object detection system will propose some candidates for the object. You can see multiple candidates for the car in Figure 1 (left), and the final bounding box for the car in Figure 1 (right). Each bounding box has a score which measures the likelihood of finding a car. Given the set of bounding boxes with their locations and their scores, propose a method for generating just one bounding box per object. Use one of the algorithms that has been covered in the class or even this homework.

Hint: One metric for measuring whether different bounding boxes are in the same local "neighborhood" is intersection over union (IoU), which is defined as follows:

$$\mathrm{IoU}(\mathrm{box1}, \mathrm{box2}) = \frac{\text{Area of intersection of box1 and box2}}{\text{Area of union of box1 and box2}}.$$

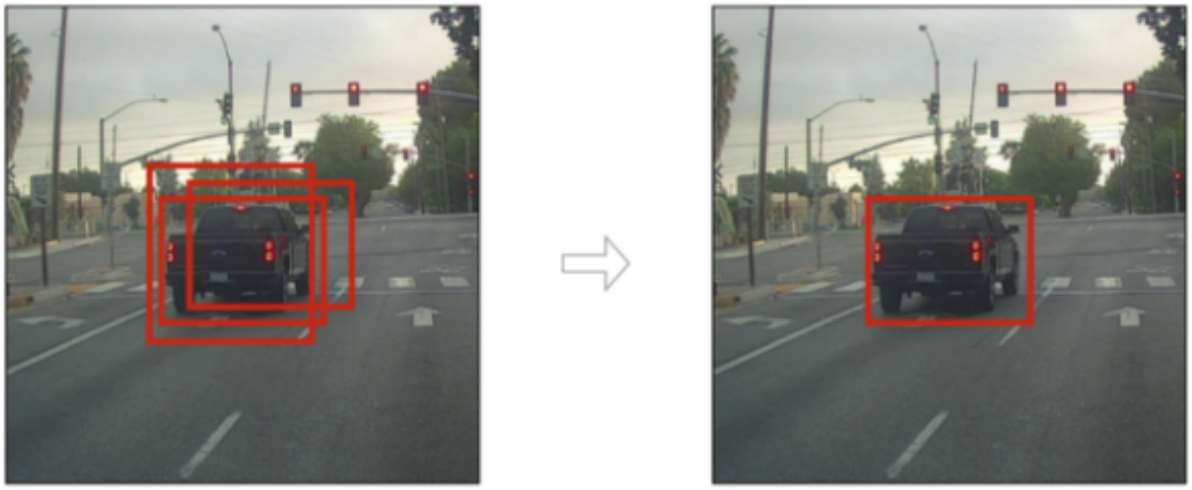


Figure 1: (Left) The object detection system identifies many candidates for the location of the car. For each candidate, there is a score which measures how likely it is to find a car in that bounding box. You can see that there are multiple red boxes, where each box will have a score between 0-1. (Right) The final bounding box for the car.

Considering there are multiple boxes we can use the following methodology to fix this issue: We use NMS(Non Maximal Suppression) : 1) We list the boxes in terms of their confidence scores(which represent the model's confidence/likelihood that it identifies the car. We sort them in descending order. 2) We then loop over each box starting with the one with the highest score. For each remaining box bj where j ¿ im we calculate the $\mathrm{IoU}(\mathrm{bi}, \mathrm{bj}) = \frac{\text{Area of intersection of bi and bj}}{\text{Area of union of bi and bj}}$. 3) If this is greater than a certain threshold(usually between 0.3 and 0.5 for practical uses), then we won't consider it for the final bounding box. We repeat this for all the boxes. 4) Finally when we have a list of all the boxes, we merge them together to create a large box (using unions and taking care of dimensions). We use appropriate adjustments for dimensions and unions.

Collaborated with Srinjana for a few problems on question 2