Homework 2: Image Enhancement & Image Restoration

Part I. Implementation:

Image enhancement part:

```python
def gamma_correction(img, gamma):
    img_float = img.astype(np.float32) / 255.0
    corrected = np.power(img_float, 1 / gamma)
    brightness = 1.5
    corrected = corrected * brightness
    corrected = (corrected * 255).clip(0, 255).astype(np.uint8)
    return corrected
```

First I've apply gamma correction using power transformation. Cause the output picture are too dark, I adjust the brightness of the picture. After that, i scale it back to 8BIT range and clip the value.

```python
def histogram_equalization(img):
    hist = np.zeros(256)
    rows, cols = img.shape[:2]

    for i in range(rows):
        for j in range(cols):
            hist[img[i, j]] += 1

    clip_limit = (rows * cols) * 0.01
    hist = np.minimum(hist, clip_limit)

    cdf = np.cumsum(hist)
    cdf_normalized = (cdf - cdf.min()) * 255 / (cdf.max() - cdf.min())
    return cdf_normalized[img].astype(np.uint8)
```

This part I implement the histogram equalization from scratch. First needs to calculate the histogram of input image. After that, I've apply clipping to prevent over-enhancement. Then calculate the cumulative distribution function (T(r) in handouts). Last, map original pixels to new values by using it.

```python
def other_enhancement_algorithm(img):
    """using Unsharp Masking"""
    # Apply Gaussian blur
    blurred = cv2.GaussianBlur(img, ksize: (5, 5), sigmaX: 0)

    # Calculate the unsharp mask
    unsharp_mask = cv2.addWeighted(img, alpha: 1.5, blurred, -0.5, gamma: 0)

    return unsharp_mask
```

In other method part, I using unsharp Masking to implement image enhancement. First, Gaussian blurring the input image. Then mix the original image with blur image.

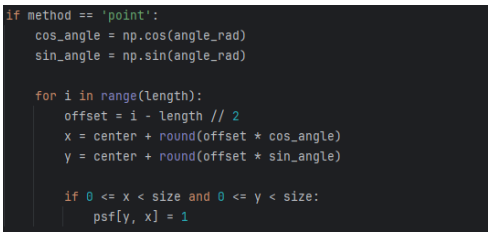
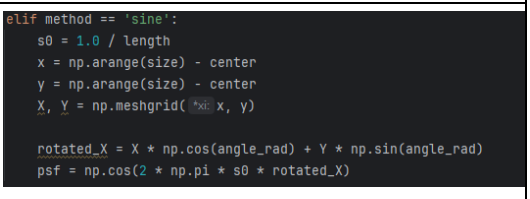
$$Unsharp\ mask = ori.img * \alpha - bur.img * \beta$$

Where alpha and beta are the weight of the pictures. In this method, the edge of the picture would be enhanced.

Image Restoration part:

```python
def generate_motion_blur_psf(size=15, angle=30, length=9, method='point'):
    """ 產生motion blur PSF, 依據testcase的不同可以調整想要的生成方法"""
    angle_rad = np.deg2rad(angle)
    center = size // 2
    psf = np.zeros((size, size))
```

I’ve implement four different kinds of method which are demonstrate on hand out. Based on different case, you can choose the best method to generate the motion blur psf.

Point	Sine
	
Line	Edge

```python
elif method == 'line':
    cos_angle = np.cos(angle_rad)
    sin_angle = np.sin(angle_rad)

    x = np.arange(size) - center
    y = np.arange(size) - center
    X, Y = np.meshgrid( *xi: x, y)

    line_distance = X * sin_angle - Y * cos_angle
    psf = np.exp(-(line_distance ** 2) / (2 * (length / 6) ** 2)
```

```python
elif method == 'edge':
    cos_angle = np.cos(angle_rad)
    sin_angle = np.sin(angle_rad)

    x = np.arange(size) - center
    y = np.arange(size) - center
    X, Y = np.meshgrid( *xi: x, y)

    edge_distance = X * cos_angle + Y * sin_angle
    psf = np.gradient(psf)[0]
```

Point: The easiest way, directly put a series of point on specific angle.

Sine: Using sine wave to generate periodic blurring mode.

Line: Using Gaussian function to generate continuous linear blur. It is more natural and smoother.

Edge: Focus on the edge detection.

```python
def wiener_filtering(image, psf, K=0.01):
    """實作wiener filter的部分，實現image restoration"""
    result = np.zeros_like(image)
    for channel in range(3):
        channel_float = image[:, :, channel].astype(float)
        channel_fft = np.fft.fft2(channel_float)
        psf_fft = np.fft.fft2(psf, s=channel_float.shape)
        psf_fft_conj = np.conj(psf_fft)
        H = psf_fft_conj / (np.abs(psf_fft) ** 2 + K)
        result[:, :, channel] = np.abs(np.fft.ifft2(channel_fft * H))

    return np.clip(result, a_min: 0, a_max: 255).astype(np.uint8)
```

Cause this homework's testcase are all color pictures. I've deal with each channel separately. First, turn it into frequency domain and match the PSF with image by the parameter s. Second, multiply image's frequency domain with wiener filter. Where

$$H(u,v) = \frac{H^*(u,v)}{|H(u,v)|^2 + K}, \begin{cases} H^*: PSF's\ conjugate\ complex\ number \\ |H|^2: Square\ of\ the\ PSF's\ amplitude \\ K: rate\ of\ noise\ and\ signal \end{cases}$$

Last, turn it back to spatial domain and clip it.

```python
def constrained_least_square_filtering(image, psf, gamma=0.01):
    """實作csl filter的部分，實現image restoration"""
    result = np.zeros_like(image)
    laplacian = np.array([[0, -1, 0], [-1, 4, -1], [0, -1, 0]])

    for channel in range(3):
        channel_float = image[:, :, channel].astype(float)
        channel_fft = np.fft.fft2(channel_float)
        psf_fft = np.fft.fft2(psf, s=channel_float.shape)
        psf_fft_conj = np.conj(psf_fft)
        p_fft = np.fft.fft2(laplacian, s=channel_float.shape)
        H = psf_fft_conj / (np.abs(psf_fft) ** 2 + gamma * np.abs(p_fft) ** 2)
        result[:, :, channel] = np.abs(np.fft.ifft2(channel_fft * H))

    return np.clip(result, a_min: 0, a_max: 255).astype(np.uint8)
```

The very difference between wiener filter and CSL filter is the function of the filter.

$$H(u,v) = \frac{H^*(u,v)}{|H(u,v)|^2 + \gamma|P(u,v)|^2}, \begin{cases} H^*, |H|^2 : same\ as\ wiener\ filte \\ |P|^2 : Square\ of\ th\ magnitude\ of\ lap.operator \\ \gamma : parameter\ of\ normalization \end{cases}$$

Part II. Results & Analysis:

Task 1 : Image Enhancement

- Gamma correction:

γ= 0.5	
γ= 1.0	

γ= 2.0	

We can see that γ= 0.5 can let the bright part in the picture getting more colorful, but can't well handle the dark part. γ= 1.0 just linear mapping. γ= 2.0 can brightness the dark part of the picture, and let us clearly see the detail. But it leads to the picture loss contrast and color.

● Histogram Equalization:

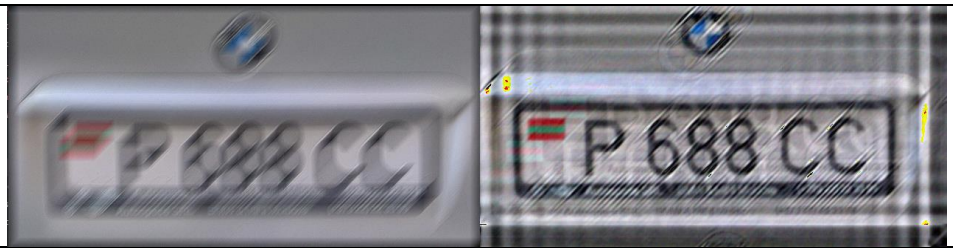
This method improves overall contrast, and do better on detail visible in both dark and bright regions. However, it still loss a little color in the bright regions, But it still did good job.

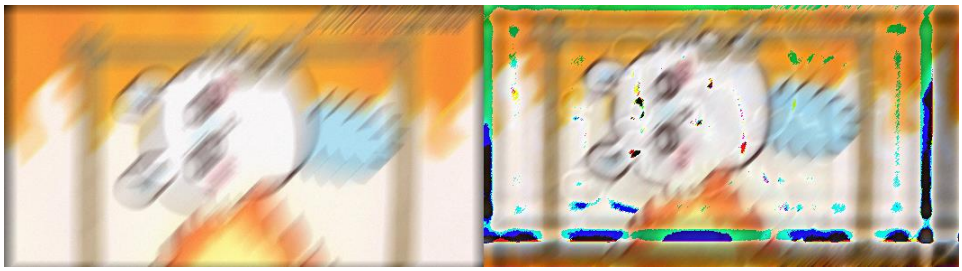
● Comparison of methods

Although histogram equalization offers better global contrast enhancement and is more natural-looking, but gamma correction provides more controlled adjustment, it is better for targeted correction. Gamma correction also maintains better color relationships.

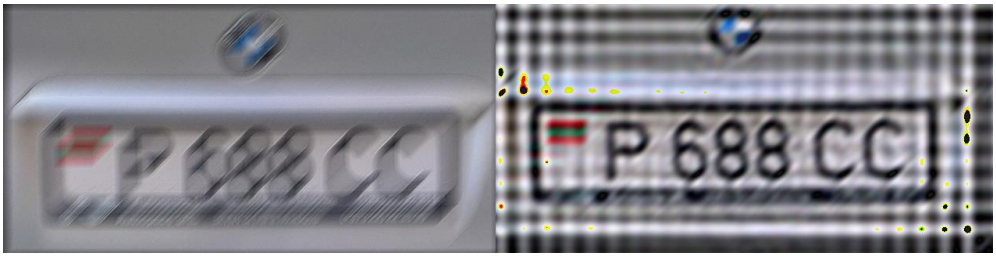
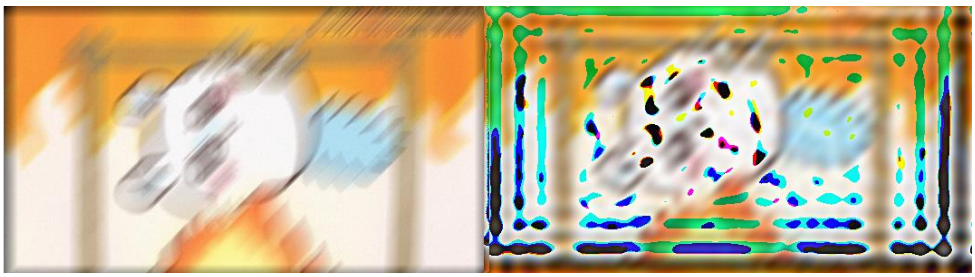
Task 2: Image Restoration

● Minimum Mean Square Error(wiener) Filtering

Testcase 1, size = 31, angle = -45, length = 13, method = line, k = 0.01


Testcase 2, size = 41, angle = -45, length = 41, method = point, k = 0.05


● Constrained Least Squares Restoration

Testcase 1, size = 31, angle = -45, length = 13, method = line, gamma = 1.4

Testcase 2, size=27, angle = -45, length = 13, method = line, gamma = 7


● Compare the above two methods on different test cases

Wiener filter's parameter K is much more sensitive than CSL filter's parameter γ. In both cases, we need to adjust gamma into higher value than k to implement greater normalizaiton. In a contrary, low value of k can do better on detail preservation.

Although there are four kinds of method in generating PSF, but in these case, cause there are both get blur by linear motion, line and point method can get much better results than others.

In a conclusion, Wiener filter performs well in both cases. While CLS filter shows less satisfactory results in case 2, its performance in case 1 is noteworthy. Despite slightly more pronounced artifacts and noise

compared to the Wiener filter, CLS filter achieves better overall contrast and text clarity. Based on my observations, CLS filter would be the preferred choice when detail preservation is a priority.

Part III. Answer the questions:

1. Please describe a problem you encountered and how you solved it.

   In Task 2 (Image Restoration), I encountered significant challenges in achieving satisfactory restoration results. The main problem was that I couldn't get good result in both Wiener filtering and Constrained Least Squares methods. If the size and length are too big, then the output would be full of noise and artifact. If the size and length are small, it would seem to the same as the original blur image. At the end, I choose to compromise to the artifact and noise. I adjust the length to a higher value, although there are significant artifact. but simultaneously keeps the output picture's outline are visible.

2. What potential limitations might arise when using **Minimum Mean Square Error (Wiener) Filtering** for image restorations? Please suggest possible solutions to address them.
   Because Wiener filter will applies same filtering parameters across whole image, so may over smooth in some region. In addition, Wiener filter are very sensitive to k, so may not got well result in the picture with different kind of blur. In my opinion, if want to deal with this kind of situation, we can divide the picture into several part first, and give different parameters to different areas.
3. What potential limitations might arise when using **Constrained Least Squares Restoration** for image restorations? Please suggest possible solutions to address them.
   In CSL restoration, it is very difficult to decide the gamma. When gamma is too low, the output picture would be full of artifact and noise. If the gamma are too high, the output picture will be over-smoothing and couldn't get good results. In this homework I've choose to compromise to the artifact, and try to find the balance between quality and noise. But

if we can apply boundary extensions or other techniques, we can no longer compromise to the quality. Another issue is the outputs picture may have severe ringing effects at the boundary of image. Because I have no idea with this problem, I've search some method to cope with this issue. The most common way is using boundary extension or weighted processing to create smooth transition area, this can reduce the discontinuous of the boundary area.