CS-GY 6643 Fall 2024 Computer Vision Project1

Remo Shen – `ys6345`

October 4, 2024

1 Histogram Equalisation - 25

(a) Brief introduction of histogram of an image and what it does. (3)
Answer: Histogram of an image is a plot that reflect the distribution of the pixels. The x-axis of this plot represents the type of pixel (e.g. grayscale level or color intensity). While the y-axis represents the total number or proportion of pixels corresponding to each value in the image.

(b) Implement the following functions: `def create_pdf(im_in)` which returns the pdf, `def create_cdf(pdf)` which returns the cdf and `def histogram_equalization(im_in)` which returns the equalized image utilizing your previously implemented functions of pdf and cdf. (7)

Answer: Probability Density Function (pdf) represents the probability of the occurrence of each intensity value in the image. To calculate it, we first calculate the histogram which gives the number of pixels at each intensity level. Then normalize the histogram by dividing the counts by the total number of pixels to get the probabilities.

Equation (1) shows calculation of pdf.

$$PDF(i) = \frac{h(i)}{N} \tag{1}$$

Where $h(i)$ represents the histogram, and N represents total number of pixels.

```python
#Defice a function that returns the pdf of the image
def create_pdf(im_in):
    pdf = np.zeros(256)
    # calculate the histogram of the image
    for i in range(im_in.shape[0]):
        for j in range(im_in.shape[1]):
            pdf[im_in[i,j]] += 1
    # normalize
    pdf = pdf/(im_in.shape[0]*im_in.shape[1])
    return pdf
```

Cumulative Distribution Function (cdf) is the cumulative sum of pdf values, which means the probability that a pixel intensity is less than or equal to a certain value. Specifically, to calculate the cdf of the image. For each intensity value, we sum up the probabilities of all intensity values that are less than or equal to the current value.

Equation (2) shows how to calculate the cdf.

$$CDF(i) = \sum_{j=0}^{i} PDF(j) \tag{2}$$

```
# Define a function that returns the cdf of the image
def create_cdf(pdf):
    #cdf
    total_pix = 0
    cdf = np.zeros(256)
    for i in range(256):
        total_pix += pdf[i]
        cdf[i] = total_pix
    return cdf
```

Histogram equalization is used to enhance the contrast of an image by redistributing its intensity values. So, what we need to do is to get the histogram of the output image uniform approximately.

To implement this (see Equation (3)): first, we can calculate the PDF and the CDF of the image's intensity values. Each pixel's intensity is then mapped to a new value according to the formula above. This mapping stretches the intensity range and enhances contrast, making the image more visually appealing and easier to analyze.

$$Equalized(k) = 255 \cdot \frac{CDF(k) - CDF_{\min}}{CDF_{\max} - CDF_{\min}} \tag{3}$$

Where, k represents the intensity value of each pixel in the image, which can be a grayscale level here.

```
# Histogram equalization function
def histogram_equalization(im_in):
    pdf = create_pdf(im_in)
    cdf = create_cdf(pdf)
    eq_im = np.zeros(im_in.shape)
    cdf_min = cdf.min()
    cdf_max = cdf.max()
    for i in range(im_in.shape[0]):
        for j in range(im_in.shape[1]):
            eq_im[i,j] = 255*(cdf[im_in[i,j]]-cdf_min)/(cdf_max-
                                                    cdf_min)
    return eq_im
```

And finally, we report the image after histogram equalization, see Figure 1.

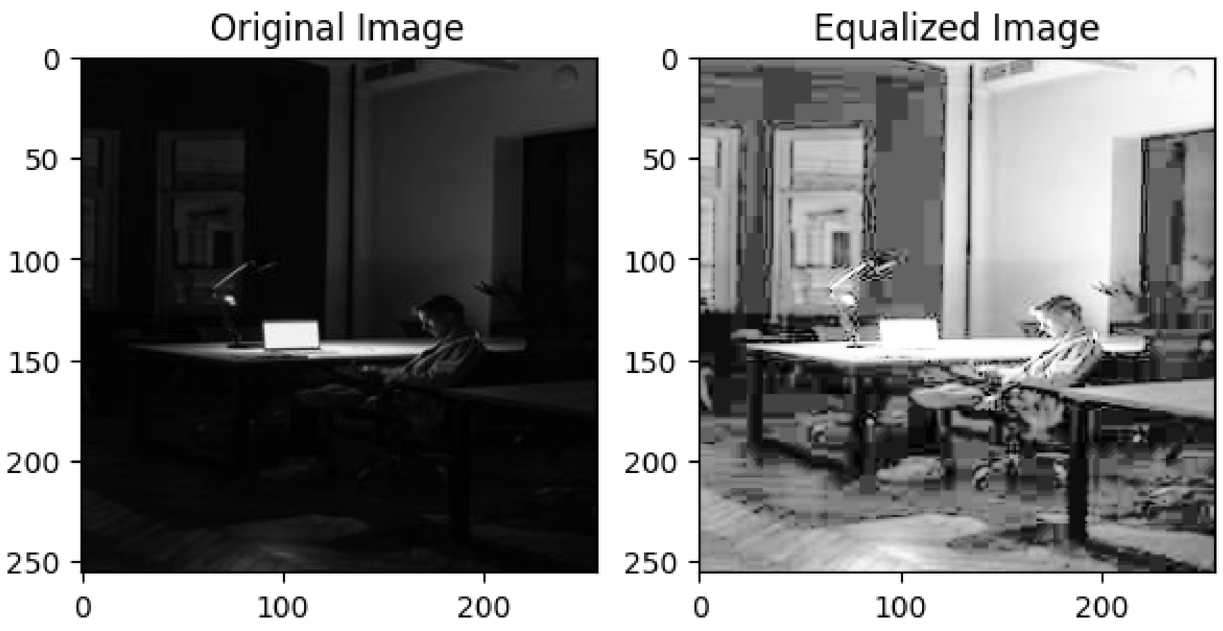

Figure 1: image before and after histogram equalization

(c) Plot pdf and cdf before and after histogram equalization. Plot them side-by-side for easy comparison. (10)

Answer: Figure 2 plot the pdf before and after histogram equalization. And Figure 3 shows the Plot cdf before and after histogram equalization.

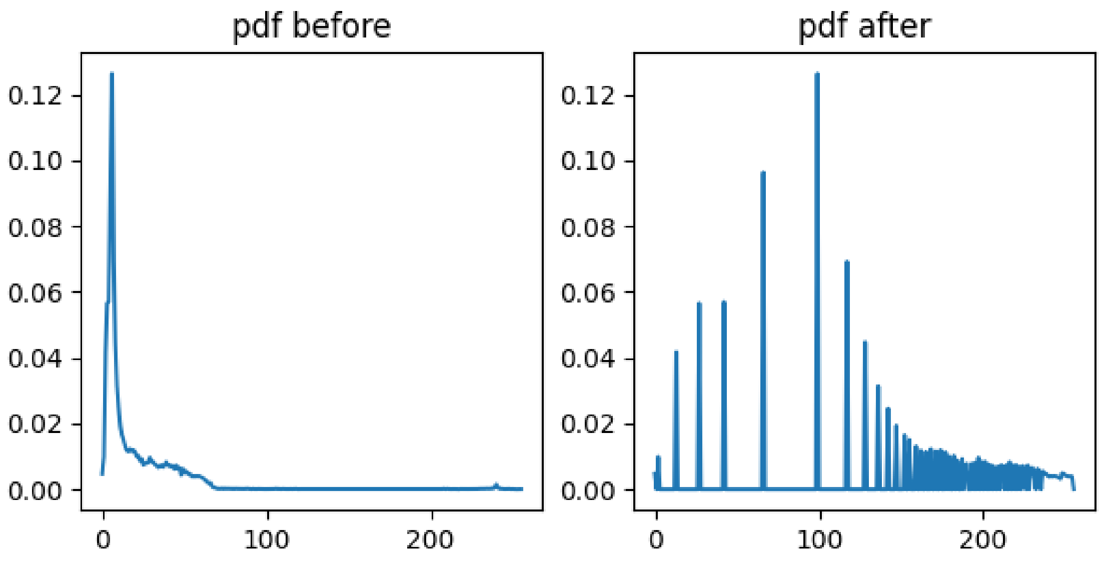

Figure 2: Plot pdf before and after histogram equalization

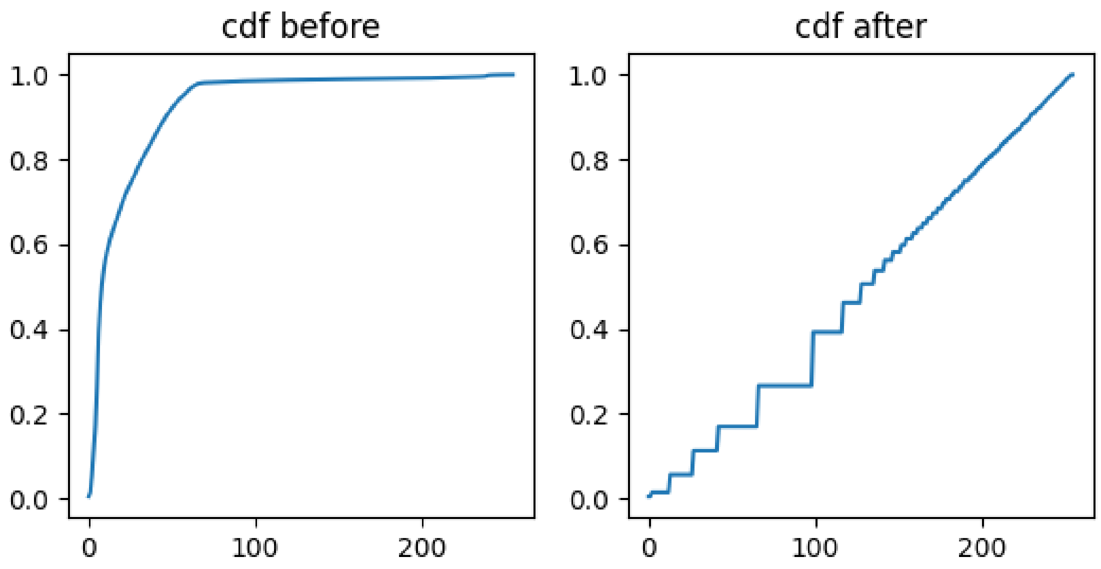

Figure 3: Plot cdf before and after histogram equalization

(d) Provide a comparative discussion of the PDF and CDF plots, and relate them to the images contrast and intensity. (5)

Answer: The pdf plot gives the distribution of pixel intensity, which shows the number of pixels corresponding to each intensity value. Pdf indicates how pixel intensities are distributed, with peaks representing frequent intensities and the valleys meaning less frequent intensity levels. A pdf with multiple peaks and valleys generally signifies high contrast,abd pixel intensities are more uniformly distributed, making it difficult to distinguish features.

The cdf represents the cumulative probability of pixel intensities being less than or equal to a given value. It starts from zero and then one, giving overall intensity distribution and its impact on image. The cdf is crucial in histogram equalization, to remap the pixel intensities and enhance contrast. Hence, the process stretches the range of intensities, improving feature visibility. The steepness of the cdf curve indicates intensity frequency.

By analyzing the pdf, we can identify regions of high and low intensity distribution. The corresponding cdf helps adjust pixel values to spread intensities more evenly across the range (0-255 for grayscale images), enhancing contrast and visual quality.

The characteristics of the pdf and cdf influence how pixel intensities are perceived. A well-distributed pdf results in a cdf that better maps intensities, allowing clearer distinctions between image features. After histogram equalization, the pdf ideally shows a uniform distribution, and the cdf covers the entire intensity range. This transformation enhances details, resulting in a more informative and visually appealing image.

2 Image Thresholding - 25

(a) Show histogram of the image. (3)

Answer: Figure 4 (a) shows the histogram of the cell TEM image.

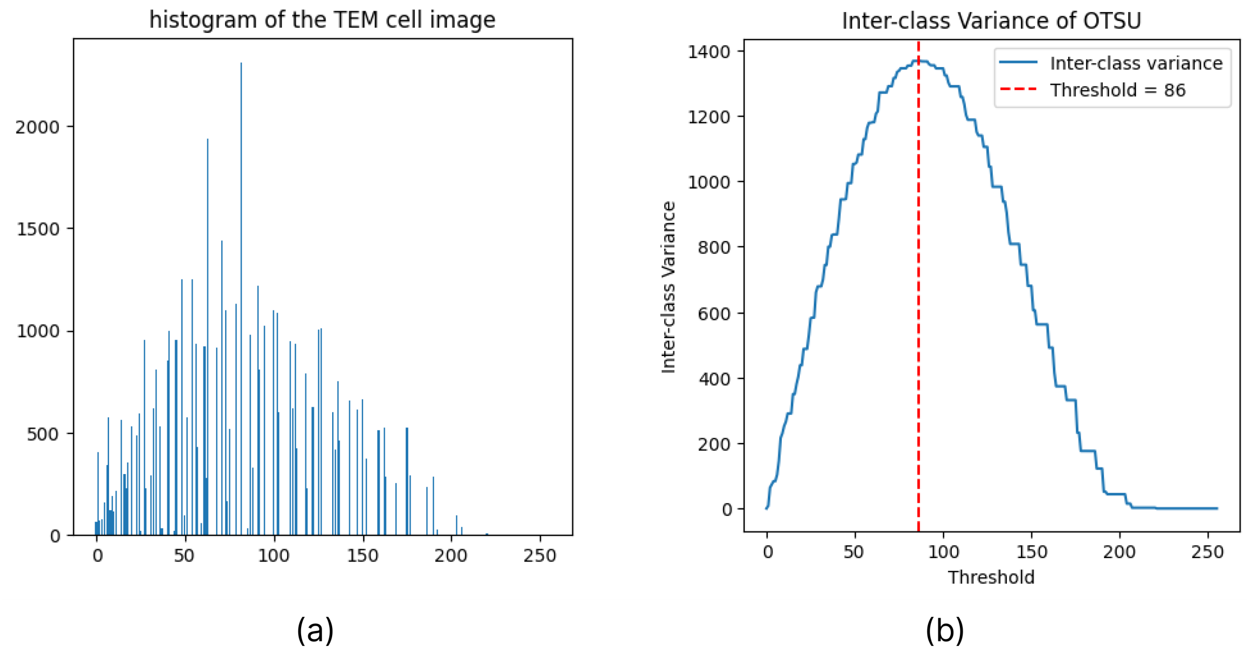

Figure 4: (a) Histogram of the cell TEM image; (b) the inter-class variance of OTSU

(b) Implement and apply Otsus thresholding and generate a plot of the inter-class variance as a function of the chosen threshold (i.e., x-axis with each possible threshold from 0-255, y-axis with the resulting variance) and highlight the threshold value where it is maximised. (6)

Answer: Figure 4 (b) shows the inter-class variance as a function of the chosen threshold. The vertical red line highlights the threshold value where it is maximised.

(c) Show the binary image after thresholding and discuss the results. Explain why do you think the results are they way they are. (3)

Answer: Figure 5 shows the binary image after OTSU's thresholding.

After applying the Otsu's thresholding, we get a binary image that converts the pixel values into black or white. The algorithm automatically selects the optimal threshold by maximizing the inter-class variance, setting pixels with intensity above the threshold as 1 (white) and pixels with intensity below it as 0 (black). This algorithm analysis the pixel distribution and try to find an optimal threshold that can distinguish between the foreground and background.

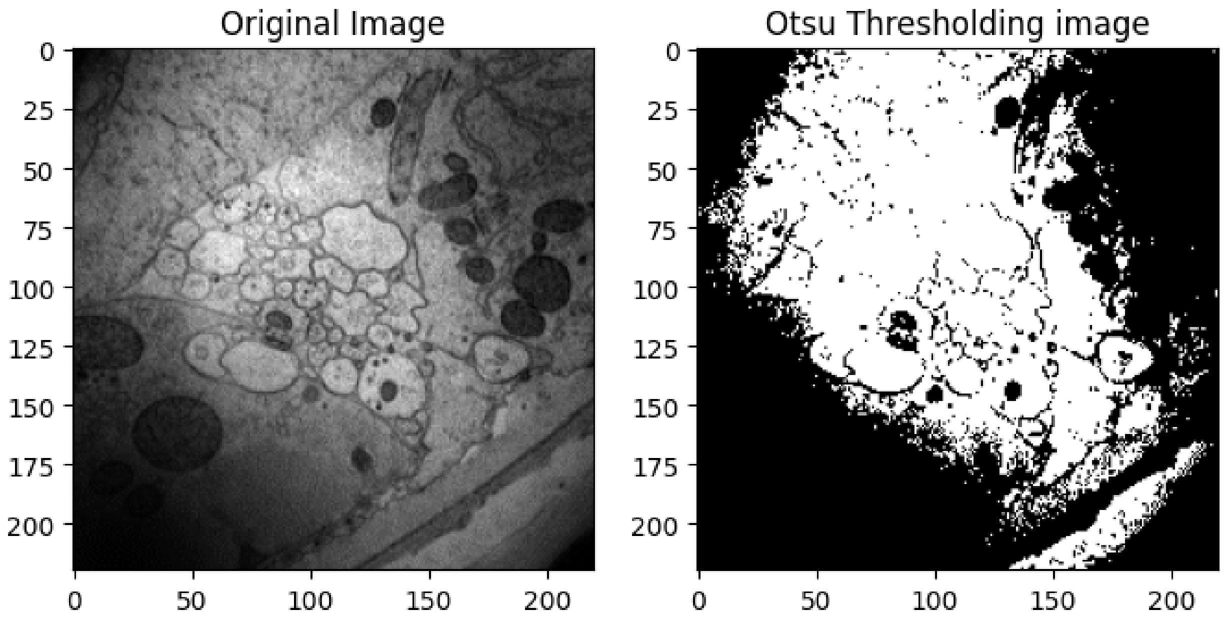

Figure 5: The binary image after OTSU's thresholding

(d) Implement and apply Niblack thresholding algorithm and show how changing the filter shape affect the results. Show the thresholded binary image with 3 examples of changing filter size. (10)

Answer: Figure 6 shows different results of Niblack's thresholding with different filter sizes

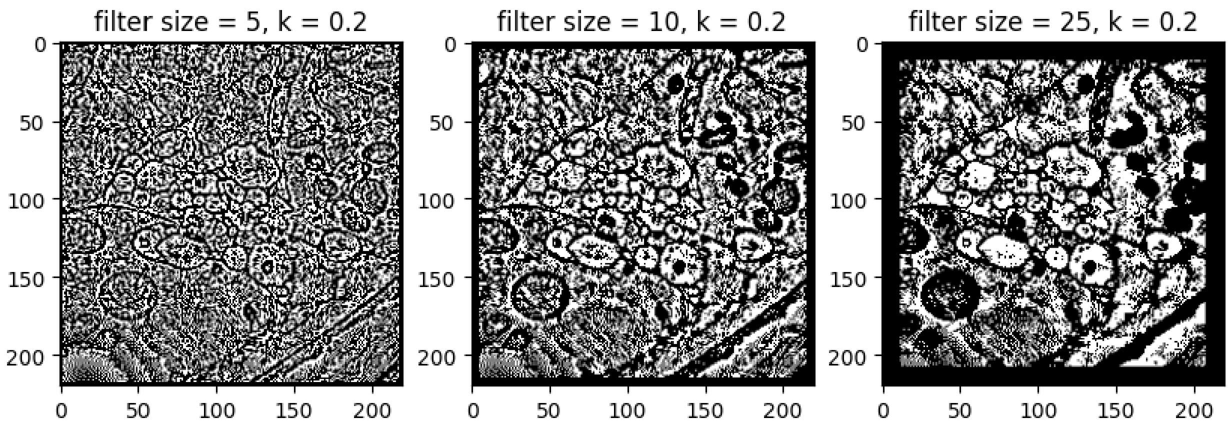

Figure 6: Niblack's thresholding with different filter sizes

(e) Comparatively discuss the two thresholding based on the results you got. (3)

Answer: For Otsu's algorithm, it can generate a binary image that separates foreground from background based on global pixel distribution effectively. The result is less sensitive to local changes, and also it can preserve important structures. But it seems difficult for the algorithm to process images uneven lighting. For Niblack algorithm, It adapts well to local conditions and is therefore effective for images with different lighting or contrast. Also, it is able to detect small structures very well, which is useful in images where detail needs to be captured. Filter size and The choice of k will significantly affect the results, wrong settings will lead to excessive noise.

3 Template Matching - 25

(a) Implement cross-correlation with the original image and the template as mask. (8)

Answer: See code in the jupyter note book.

(b) Parse the correlation image to detect the maximum peak value and its (x, y) location. You may mark this location with an overlay of a circle or just manually painting an arrow or else. Discuss if this location matches your expectations. Discuss the appearance and cause of other peaks, and how they compare to the global peak, you may just manually check other peaks and annotate them on the peak image. (10)

Answer: See Figure 7, the left part of it shows the final result after using the template matching algorithm. I successfully find the maximum peak value and I label it in the image by using blue circle. This location matches my expectations because it is the brightest point in the picture. And also, we try to label this location in the original image which shows in Figure 8, we can see that the final result fit the template.

I also labeled other peaks (90% of the maximum peak). See the right part. The presence of these peaks may be due to similar textures or patterns in the image. By observing the correlation values of these peaks, I find that their values are significantly lower than the global peak, indicating that their matching degree with the target template is low.

(c) Show the image which helps you identify the peaks (cross-correlation output) with a mark on the highest peak.(2)

Answer: See Figure 7, left part. I can identify the peak according to the brightness.

(d) Overlay the found peak on the original image. (5)

Answer: Figure 8 shows the result of the template matching. We overlay the found peak using blue circle.

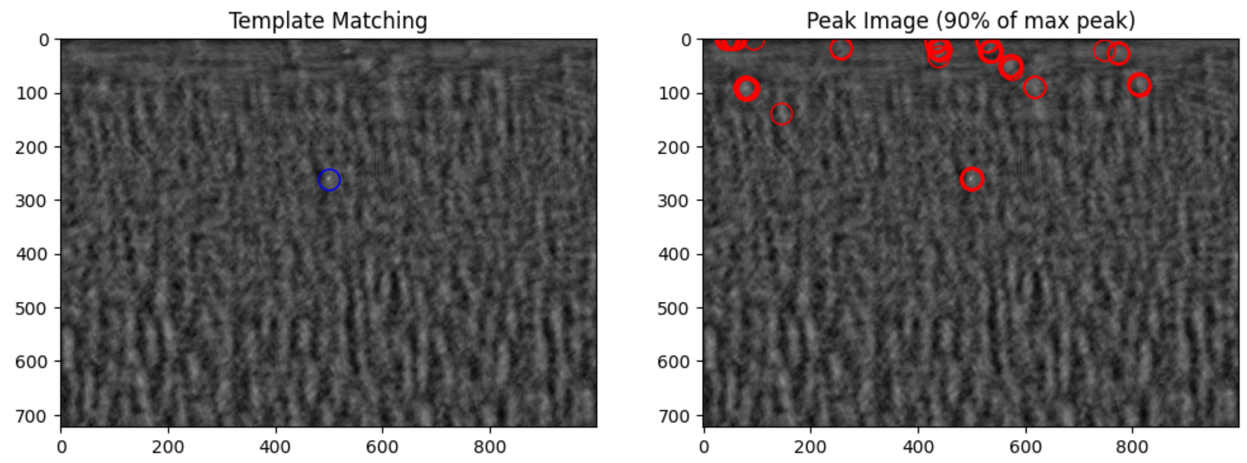

Figure 7: the left part shows the final result of template matching, the right part shows other peaks (90% of the max peak)

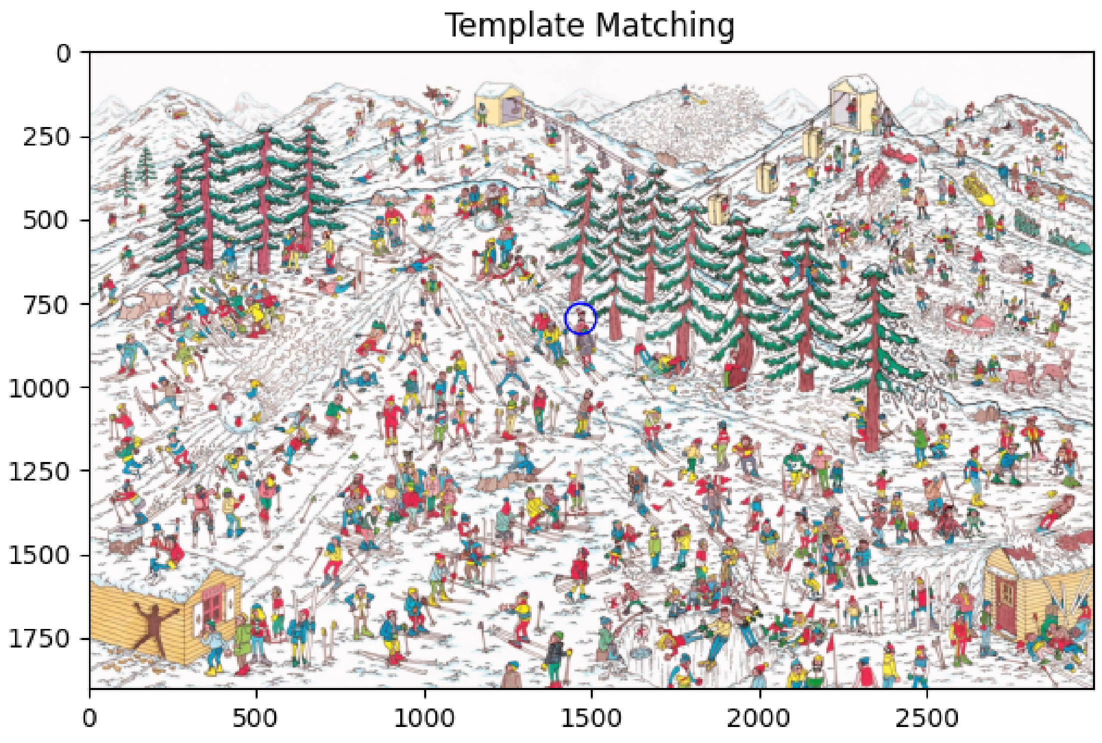

Figure 8: the result of the template matching, overlay the found peak using blue circle

4 Creative Section - 25 + 5

4.1 Introduction of the idea

Template matching is a commonly used image analysis technique.it is widely used in object detection and recognition. Traditional template matching algorithms may result in excessive computational load when processing large images, while image pyramid provides an efficient multi-scale method that can reduce the computational load and increase the matching speed. This report describes a template matching algorithm based on image pyramid, including its basic idea, steps and running time analysis.

4.2 Algorithm Overview

The algorithm contains the following steps:

1. **Calculate Integral Image:** Integral Image is used to quickly calculate the sum of pixel values in any rectangular area. Through the cumulative summation method, the image is converted into an integral image, so that the subsequent area summation operation can be completed in constant time.

2. **Create Pyramid of the Image:** Generating image pyramids using Gaussian downsampling. Each layer of the pyramid is a downsampled version of the image from the previous layer. Through the pyramid structure, matching can be performed on different scales, improving the robustness and efficiency of the algorithm.

3. **Template Matching:** Start at the top level of the pyramid and work your way down to the next level. In each layer, the integral map is used to calculate the squared difference of the current template in the sliding window in the image to find the best matching position. The template size is adjusted accordingly as the number of layers changes.

4. **Update matching positions layer by layer:** When entering the next layer, the matching positions are updated to adapt to the new scale, ensuring that the matching process is performed on smaller images to reduce the amount of calculation.

4.3 runtime complexity

1. **Build Pyramid:** The time complexity for constructing the pyramid is $O(nlogn)$, where n is the number of pixels in the image.

2. **Integral Image:** the run time complexity for integral image is $O(n)$.

3. **Template Matching:** the time complexity for template matching is $O(m \cdot k^2)$, where m is the number of pixels in original image, and k is the template size (pixel).

4.4 Result

Figure 9 shows the result of the template matching by using image pyramid. I overlay the found target by using the blue circle.

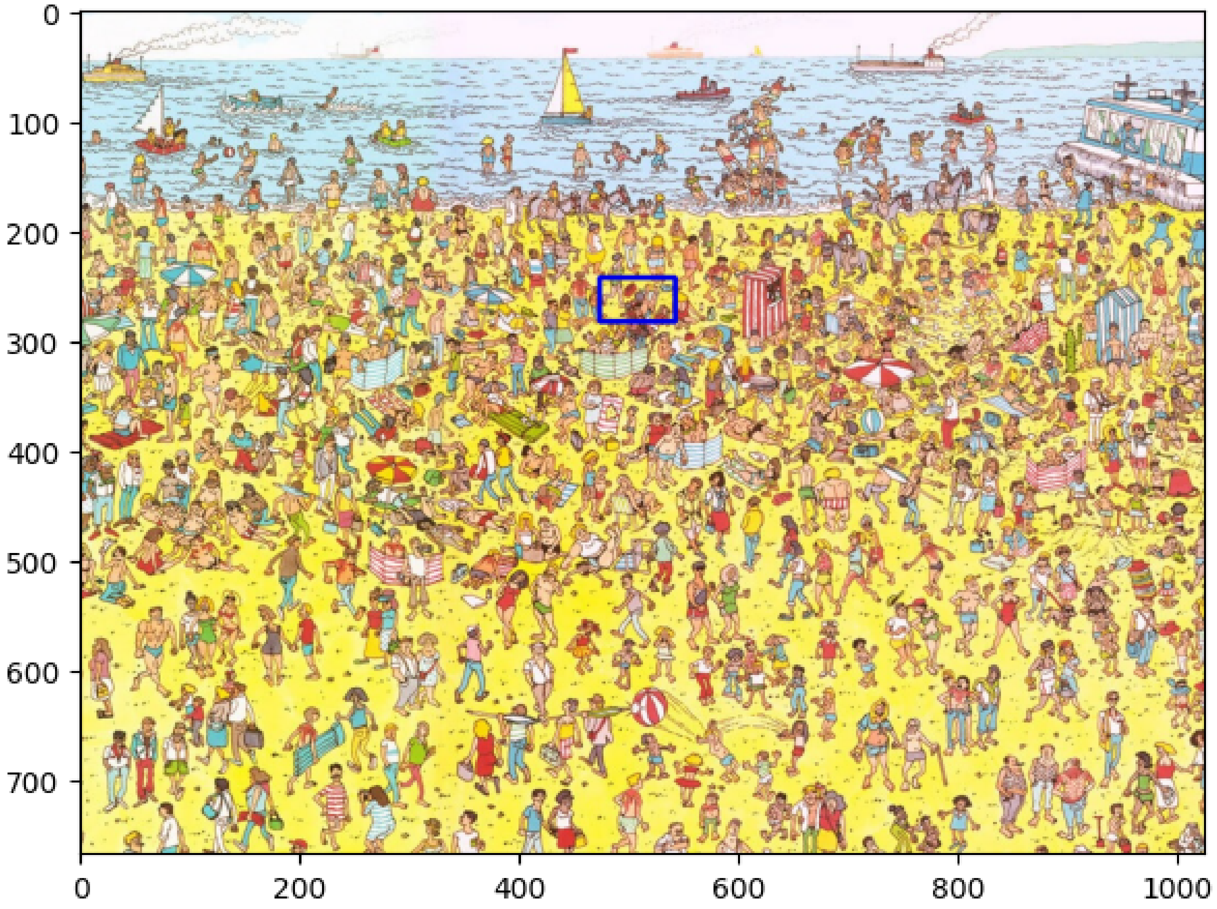

Figure 9: the result of the template matching, overlay the found target using blue circle