Computer Vision Project 1 Sample Documentation

Malhar and Rishabh

2024/10/18

Contents

1 Note

This documentation serves as a sample of the level of detail and structure we expect from students, while intentionally leaving programming solutions open-ended to encourage creativity and unique approaches. The document aims to provide conceptual explanations and illustrative examples but does not include the actual solutions for the questions, as ample space was provided for those in the Google Colab notebook. The intention behind submitting a PDF explanation of the project is not to replicate solutions meant for the Colab environment but to produce a standalone document that can be understood by someone without a background in Computer Vision, which in turn lets us grade your conceptual knowledge.

Furthermore, while the documentation outlines the concepts, it does not delve into the mathematical derivations of the functions; students are encouraged to explore these details in their own submissions for a more comprehensive discussion if they want. We created a sample overleaf document for easy editing: Clicky-Linky. Additionally, figures for the bonus section (5) have been omitted, as it was an open-ended question, and our primary goal here is to illustrate the expected documentation style and structure rather than provide complete solutions.

2 Histogram Equalisation

2.1 Introduction

Histogram equalisation is an image processing technique that helps in improving the contrast of a given image. It does by trying to spread the concentrated histogram (low contrast) whose spread is limited to a certain range,to the full range of possible value of the intensity(usually 8-bit). In the process of this transformation the bins are observed to be spread apart along with some areas to be squeezed as the number of pixels remain the same. Steps to perform the the transformation(equalisation) are as below:

- Create the pdf of the image
- Using the pdf created above compute the cdf
- Round off the cdf for each intensity value(as images are in discrete domain)
- Use this as a mapping to change the value an intensity in the original image to the transformed image

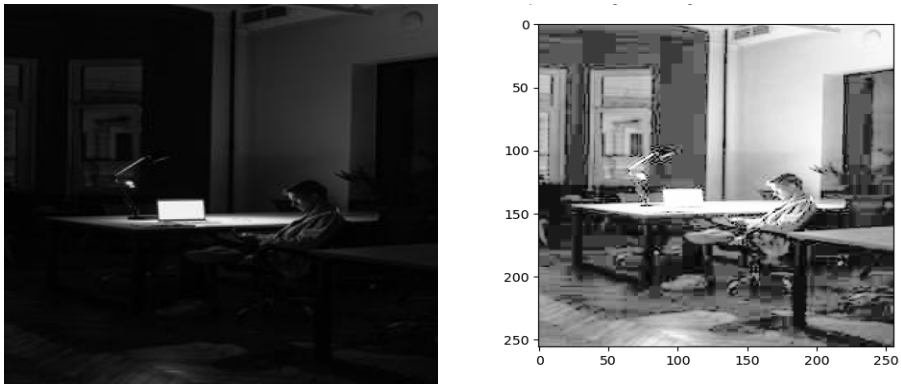
This will transform the image to one with the pixels intensities ranging in the full range and hence the final image have a better contrast(in principle).

2.2 Sample Image used for Histogram Equalisation and Code

We will use Figure 1image to demonstrate the effects of histogram equalisation on an image. As evident here the image has a very low contrast and hence the details of the image are not evident.

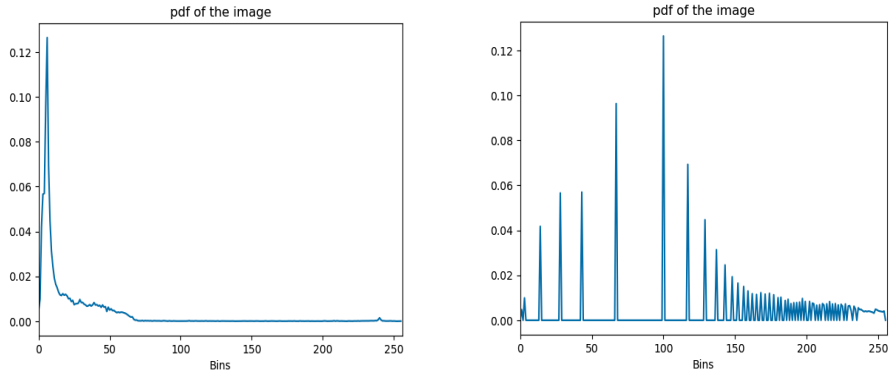
Figure 1

2.3 Visual Comparisons



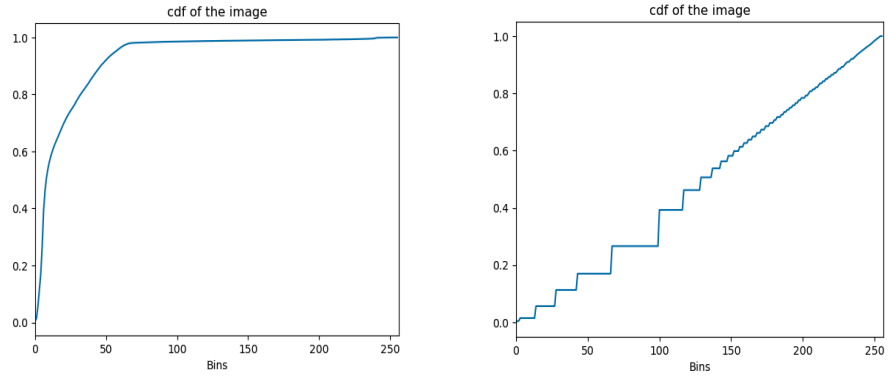
(a) Original Image (b) Transformed Image

Figure 2: The image after histogram equlaization



(a) pdf of Original Image (b) pdf of Transformed Image

Figure 3: pdf after histogram equalization



(a) cdf of Original Image (b) cdf of Transformed Image

Figure 4: cdf after histogram equalization

2.4 Observations

The comparison above allows for a visual assessment of the improvements made through the equalization process. To expand further, the Probability Density Function (PDF) represents the likelihood of a random variable (in this case, pixel intensity) taking on a particular value. For an image, the PDF provides a statistical distribution of pixel intensities, indicating how often each intensity occurs in the image, with the image divided in 256 different types of intensities, with 0 standing for black values and 255 standing for white. In the context of Histogram Equalization, the PDF is computed by generating a histogram of pixel intensities in the image. The histogram divides the intensity range into bins, and each bin corresponds to a certain range of pixel intensities. The frequency of occurrences in each bin is normalized to obtain the probability density, giving us the PDF. For a grayscale image, the PDF provides insights into the distribution of dark and bright pixels. In Histogram Equalization, modifying the PDF involves redistributing the probabilities of different intensity levels, ultimately enhancing the overall contrast of the image. The initial examination of the original image revealed areas of extreme darkness, where finer details were obscured.

The Histogram Equalization process works by reshaping the pixel intensity distribution, mitigating the prevalence of extreme darkness and enhancing overall contrast. Inspecting Probability Density Function (PDF) of Figure 3a, we observe the uneven pixel distribution centered on the initial bins, reflecting the darkness of the image. Post-Histogram Equalization, the transformation is apparent in Figure 3b, where the equalized image and its PDF showcases a more uniformly distributed intensity range, lightening the initial darker parts. This adaptive adjustment of pixel intensities corresponds to the methodology described in the introduction, where Histogram Equalization redistributes intensity values to cover the entire available spectrum.

The Figure 4 compares the Cumulative Distribution Functions (CDFs) of the original and equalized images. The blue curve of 4a represents the CDF of the original image, and the blue curve of 4b corresponds to the CDF of the equalized image. Analyzing this plot provides insights into the cumulative probability changes induced by Histogram Equalization. Shifts and variations in the curves help in understanding how the transformation has influenced the overall contrast and brightness of the image. The Cumulative Distribution Function (CDF) is derived from the PDF and provides information on the probability that a random variable (pixel intensity) takes on a value less than or equal to a given intensity. In the context of image processing, the CDF is computed for a point by accumulating the probabilities from the PDF. Specifically, each point in the CDF represents the sum of probabilities up to that certain point. The CDF is a monotonically increasing function, ranging from 0 to 1, and is particularly useful in Histogram Equalization. In Histogram Equalization, the objective is to transform the original CDF into a linear distribution, effectively spreading out the intensities across the entire available range. This redistribution of intensities helps in enhancing the contrast of the image.

We can see that after the histogram equalisation of the image we can clearly see a lot of details which were hidden(not clearly visible) in the original image. The transformed image has a much better contrast than the original image evident clearly from the pdf of the transformed image. There is also few drawbacks of this transformation as boxy artifacts have emerged in the image(though solvable by appropriate filtering) because some of the loss in information when pixels are squished onto each other as this is a discrete world and rounding off the numbers may introduce such scenarios.

3 Otsu's Image Thresholding

3.1 Introduction

Image thresholding is a technique in Image processing where we convert an image in to a binary image about a fixed pixel intensity. Any intensity below this threshold intensity is brought down to 0 and anything equal or above is updated to be 1. This is particularly beneficial in image segmentation or identification of areas of interest. We can choose this threshold manually or through some algorithm which tries to find the best threshold. One such algorithm which we will explore is Otsu thresholding, it is an interesting approach where the algorithm exhaustively searches for an optimal threshold where is tries to maximize inter class variance (separate the intensities to two classes) or minimize the intra class

variance (group all pixel intensities of one class), class here is either back ground and foreground etc.

3.1.1 Algorithm

- Compute histogram and probabilities of each intensity level
- Set up initial ω_i and $\mu_i(0)$
- Iterate over the possible thresholds t = 1,.....max intensity
  - Update ω_i and μ_i
  - Compute $\sigma_b^2(t)$
- Find the threshold which corresponds to the maximum $\sigma_b^2(t)$

We will now use the below image and try Otsu's Thresholding on it and compare our findings with a new thresholding algorithm - Niblack Thresholding

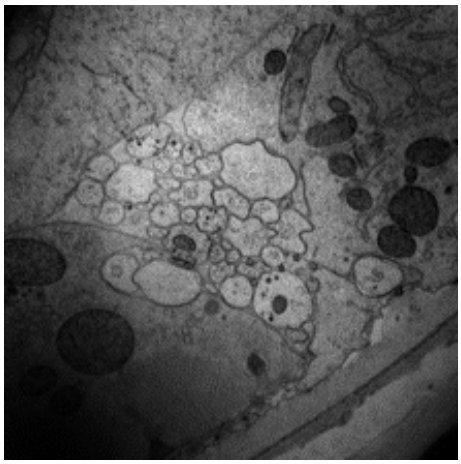


Figure 5: Image used for comparison

3.1.2 *pdf* of the image

pdf of the input image to visualise the histogram nature

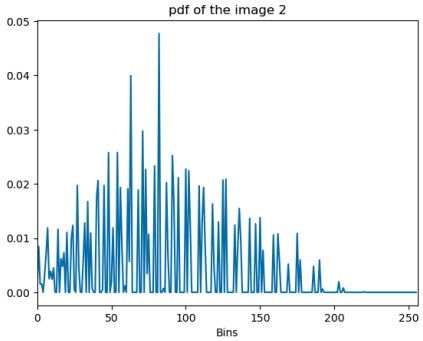


Figure 6: pdf of Image used for comparison

3.2 Inter-Class Variance to select the Threshold in Otsu

Inter-class variance is a crucial measure in image processing, especially when deciding how to separate different parts of an image. Imagine a grayscale image where you want to distinguish between the object

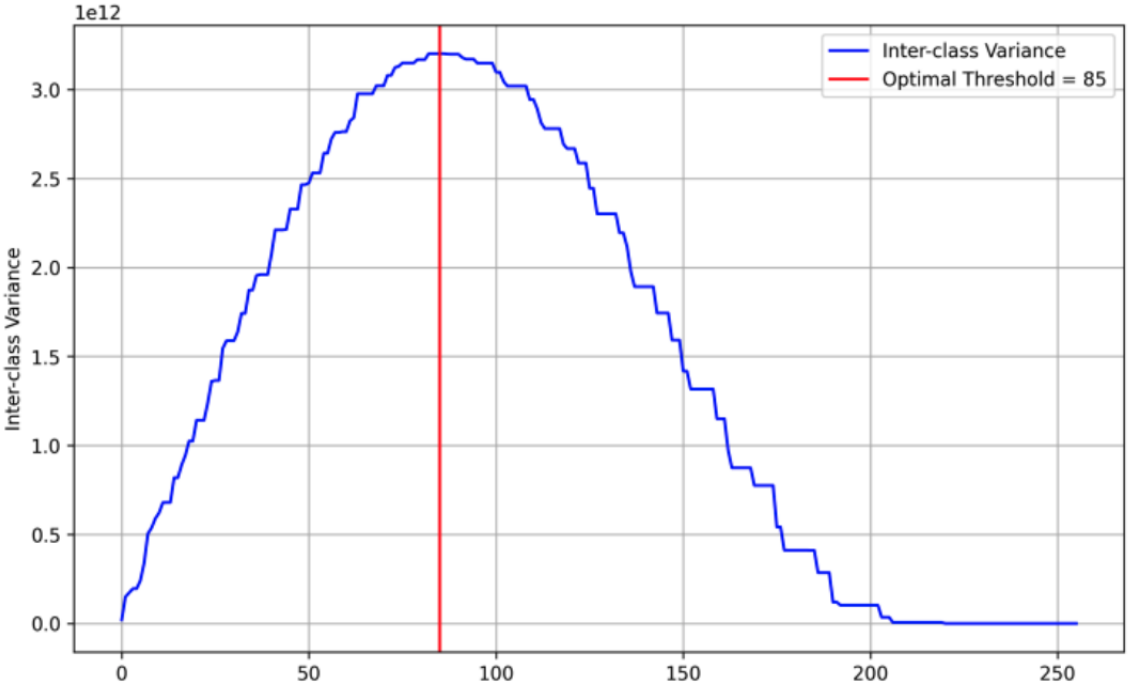


Figure 7: Inter class Variance vs threshold

you're interested in and the background.For each possible threshold, we calculate the probability of a pixel falling into one of the two classes (object or background). We also find the average intensity (mean) for each class. The key is to measure the difference between these means and how spread out the classes are from each other. This is represented by the between-class variance, a value that tells us how well a particular threshold distinguishes between the two classes. The goal is to find the threshold that maximizes this between-class variance. In simpler terms, we're looking for the threshold that best separates the object from the background by maximizing the difference in average intensities. The figures mentioned in this subsection display exactly that - the different values of variances resulting from the above calculation for each threshold from 0-255.

3.2.1 Observations

We can see clearly from the plot that at θ value 85 the interclass variance is maximized and hence minimizes the intra-class variance thus statistically(based on pdf) seperating the two classes into background and foreground.

3.3 Otsu's Thresholding

Thresholding based on the θ value where inter class variance is maximized

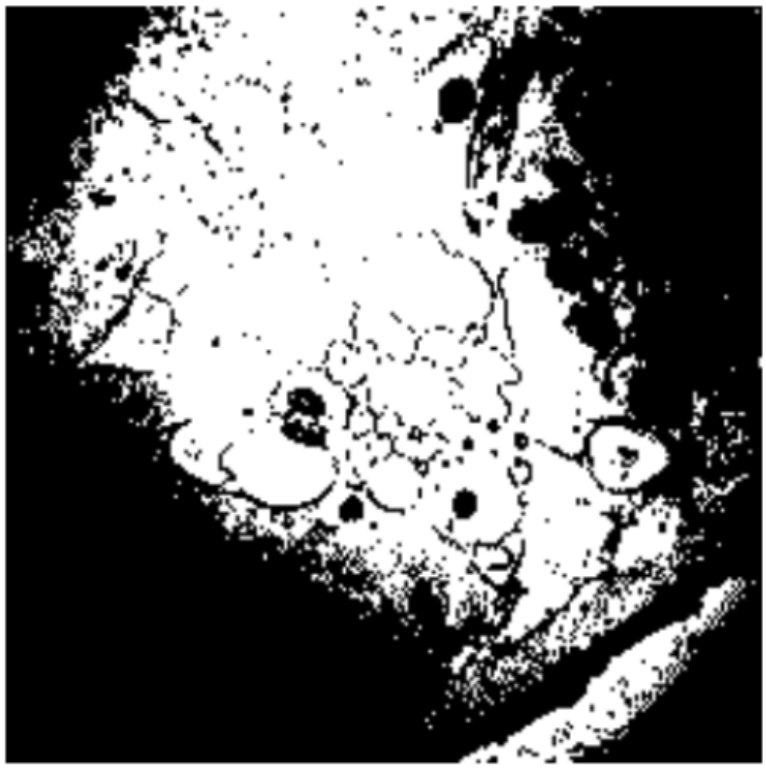


Figure 8: Binary image after Otsu Thresholding thresholded at 85

3.4 Niblack Thresholding

The niblack thresholding function is another method for converting a grayscale image into a binary image. Unlike Otsu's method, which uses a single global threshold, Niblack's method calculates a local threshold for each pixel based on the pixels around it. The function loops through all pixels in the image For Each Pixel: It takes a small window of pixels around the current pixel according the a window size we pass as an argument We compute the mean and standard deviation(std) of the intensities within the window. The threshold for the current pixel is calculated using the formula

$$mean + k \times std$$

The value of k adjusts how much the standard deviation influences the thresh- old.(We pass it as an argument but kept as the standard value of -2)

If the pixel's intensity is greater than the threshold, it's set to 255 (white) in the binary image, otherwise, it's set to 0 (black). Different window sizes are tested shown in Figure 9

3.5 Comparative study

Thresholding algorithms like Otsu's and Niblack's thresholding techniques show different strengths in the context of enhancing cell boundaries in TEM images. While Otsu's method has a global thresholding

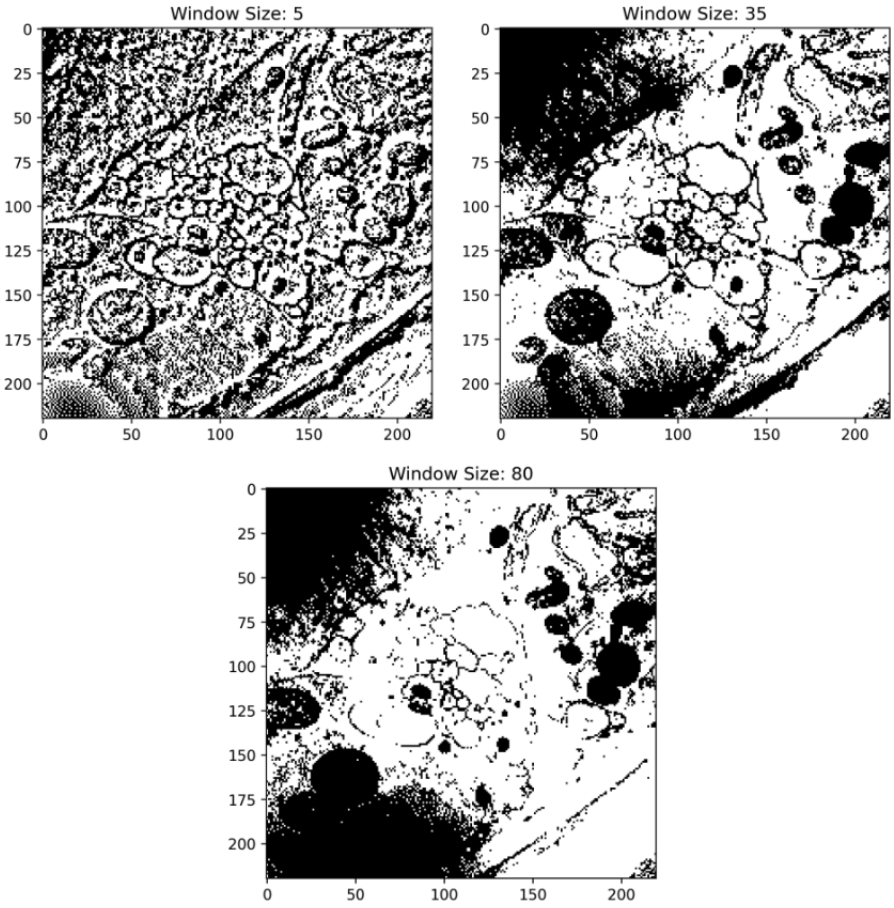


Figure 9: Note: I know the images look different, I just saw that I accidentally overwrote the expected image by something and I can't recover it

approach it provides an easy and strong way to segment the foreground out from the background, giving sharp segmentations with well defined cell structures even though it may miss fine details. On the contrary, thresholding by Niblack automatically adapts the threshold according to local variation by computing the mean and variance within a window of specified size. This technique captures more detailed information of the cells, although it introduces some noise. If there is a crucial need for the preservation of detail due to variations in brightness and staining, Niblack seems to be considerably more appropriate.

4 Cross-Correlation and Template Matching

In this section, we focus on cross-correlation and template matching, pivotal techniques in image analysis. The code employs a template image and the original image to demonstrate how cross-correlation helps identify the presence of the template within the larger image. By implementing normalization and convolution operations, the code showcases the correlation results and employs thresholding to highlight significant peaks, providing an illustrative representation of successful template matching. The code and pictures help us see how well the computer can find and match these patterns.

4.1 Correlation/Convolution

Let's first try to understand what these concepts mean. We will take an example of a Gaussian filter, one of the simplest filters in Computer Vision, which is used for blurring an image. The Gaussian filter is characterized by a bell-shaped curve, and its application involves convolving the filter with the input image. This process reduces noise, suppresses high-frequency details, and results in a more visually appealing and informative image. The sigma parameter in the Gaussian function determines the spread or standard deviation of the Gaussian distribution. A larger sigma value results in a broader, smoother filter, affecting the extent of blurring.

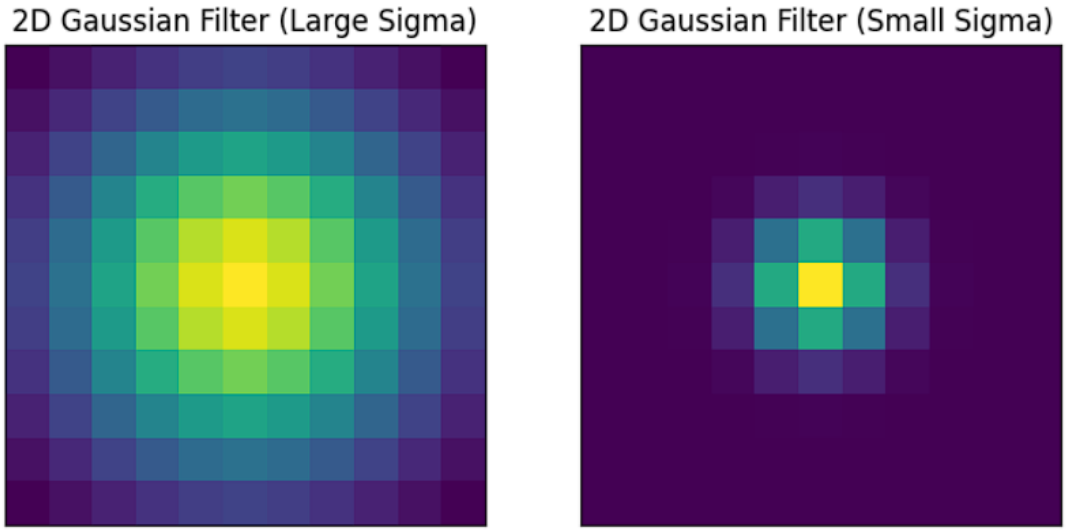


Figure 10: Comparison of two types of Gaussian filters with varying sigmas

The convolution/cross-correlation operation works in a way where each pixel in the output image is calculated by summing the element-wise products of the filter and the corresponding region of the input image. To facilitate this, the input image is padded to handle edge cases effectively. This operation, in its most basic form, is implemented using nested for-loops, iterating through each pixel position in the input image and computing the weighted sum according to the given filter. The resulting image reflects the smoothed representation of the original image. A major difference between convolution and cross-correlation is that convlution flips the template and cross-correlation works by sliding the un-flipped template over the original image and calculating how well they match at each position. The latter result, known as the correlation, indicates how similar the two patterns are at each location.

4.2 Normalization of the image

In the context of this code, normalization involves adjusting the brightness of the template image to ensure fair comparisons during the cross-correlation process. The mean value of the pixel intensities in the template image is calculated, representing the average brightness. Subtracting this mean from each pixel in the template effectively centers the image on zero, creating a zero-mean template. Normalization is crucial because it eliminates any bias caused by variations in overall brightness, allowing the cross-

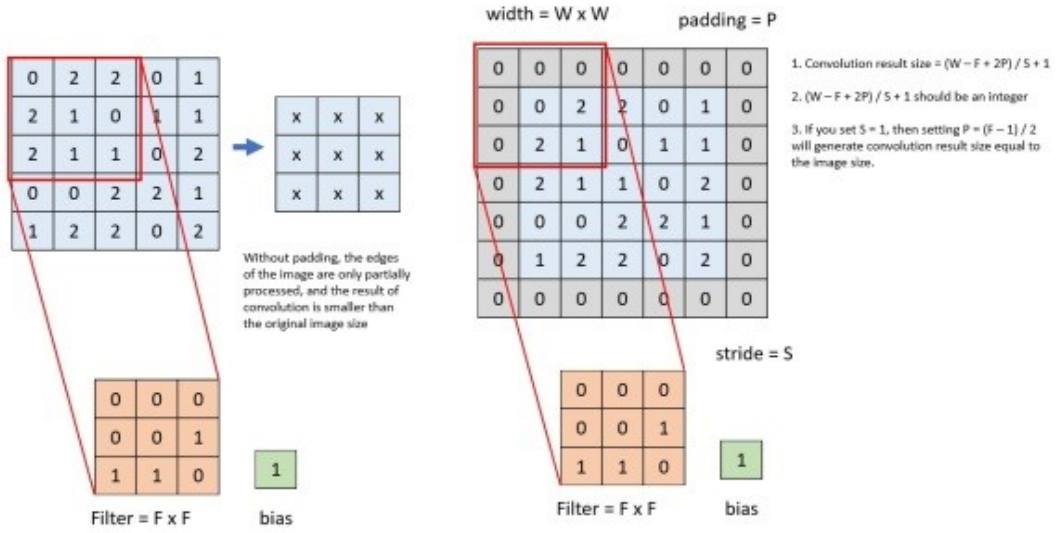


Figure 11: Working of the convolution/cross-correlation operation

correlation to focus on the specific pattern rather than being influenced by overall intensity changes. This step enhances the accuracy of the template matching process.

4.3 Thresholding Analysis

Thresholding is a crucial step in this process. It involves setting a certain level (threshold) and considering only those correlation values that exceed this threshold as significant. This step helps filter out less relevant information and focuses on the most meaningful matches. Choosing an appropriate threshold is important because it determines the sensitivity of the peak detection. A higher threshold may result in only the strongest and most reliable matches being considered, while a lower threshold may include more matches, including potential false positives. The thresholding process essentially separates the correlation values into two categories: those indicative of a match (peaks) and those that are not. This separation is crucial for effective template matching, allowing us to distinguish meaningful patterns from background noise and irrelevant matches.

After thresholding the correlation results to identify significant peaks, the next step is to overlay this thresholded image onto the original image to highlight regions of interest. This overlaying process involves superimposing the thresholded image, where peaks are represented as bright spots, onto the original image. By doing so, we effectively "mark" the regions in the original image where the template closely matches, providing a visual indication of where the pattern of interest is found. Bright spots in the thresholded image correspond to areas of high correlation, indicating strong matches between the template and the original image. When overlaid onto the original image, these bright spots serve as markers, highlighting the specific locations where the sought-after pattern is present. This overlaying process not only facilitates easy visualization of the regions of interest but also aids in the interpretation and analysis of the template matching results, allowing for quick and intuitive identification of important patterns within the image.

In order to enhance the clarity of our template matching results, we systematically applied thresholding to our correlation results, incrementally varying the threshold values in steps of 20. Subsequently, we overlaid each thresholded image onto the original image, resulting in conspicuous red spots at points of significant correlation/interest. These red spots denote areas where the template closely matches the corresponding regions in the original image, effectively highlighting regions of interest and facilitating the visual identification of potential matches. This could be attributed to the fact that a lot of the matches have similar clothing with Wilma and/or share similar characteristics with the template Wilma. An

important thing to note here is that the image was zero padded before template matching to make sure that the output had the same size as the input image, this led to false peaks around the image borders.

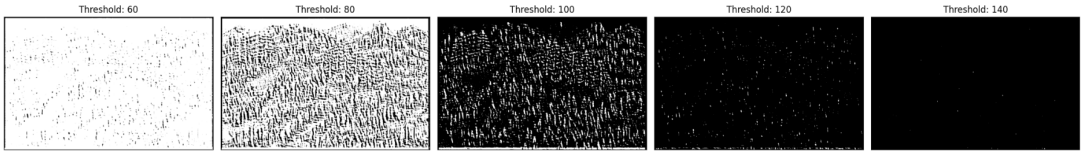


Figure 12: The cross-correlated image at different thresholds

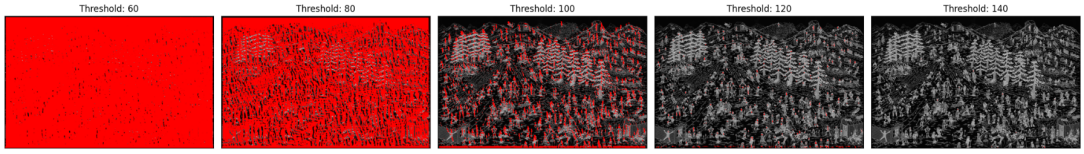


Figure 13: Overlaid thresholded image (in red) on the original image

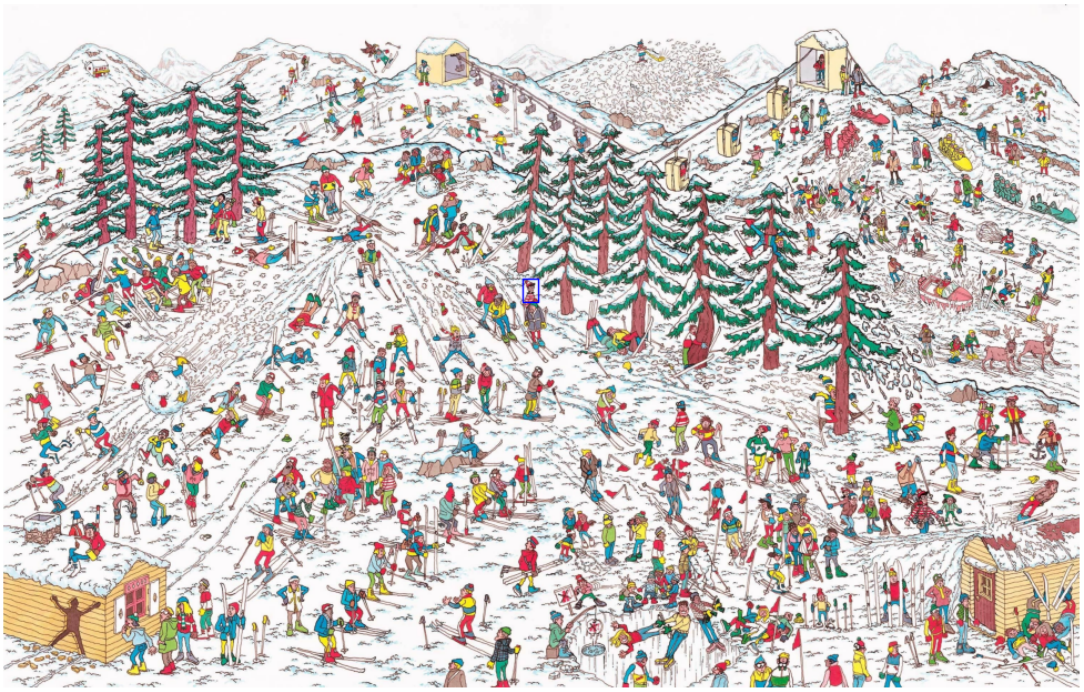


Figure 14: Maximum peak match (blue rectangle, near the center of image) for Wilma

5 Fast Normalized Cross Correlation

Template matching using Normalized Cross-Correlation (NCC) is a widely used technique in image processing, especially for pattern recognition and object detection. The traditional approach for calculating NCC, however, can be computationally intensive and slow, especially for large images and templates. To address this limitation, **J.P. Lewis introduced Fast NCC** ([1]), an optimized algorithm that significantly reduces the computational complexity of the vanilla method.

5.1 Challenges with Vanilla NCC

The standard approach for computing NCC involves sliding the template over the image and computing the cross-correlation for every possible position. For each position, the calculation requires computing sums over the image region, as well as the variance and mean for both the template and the image patch. This involves repetitive computations that make the process inefficient, especially when working with large images or templates. The main computational bottlenecks in vanilla NCC are:

- Repeated computation of sums and means for overlapping regions.
- Expensive variance calculations for every region.
- High time complexity due to the large number of operations required.

These challenges make vanilla NCC unsuitable for real-time applications or for processing high-resolution images where fast response times are necessary.

5.2 Optimizations in Fast NCC

Fast NCC by J.P. Lewis introduces optimizations by leveraging integral images (also known as summed-area tables) and precomputed sums, significantly reducing the number of operations needed:

- Use of Integral Images:
  - An integral image is a representation in which each pixel contains the sum of all pixel intensities above and to the left of it. This allows the computation of the sum of any rectangular region in constant time.
  - Similarly, an integral image for squared intensities can be used to compute the sum of squared pixel intensities within any region in constant time.
- Efficient Calculation of Mean and Variance:
  - With integral images, the mean and variance of any image patch can be calculated using just a few arithmetic operations, avoiding repeated summations over the same pixels.
  - The mean of a region is obtained by calculating the difference between the sums from the integral image.
  - The variance is computed using the sum of squares and the mean, derived from the integral image for squared intensities.
- Precomputation of Template Statistics:
  - The template mean and variance are computed once and subtracted from the corresponding region of the image. This avoids recalculating the template statistics repeatedly.
  - The algorithm performs zero-mean normalization for the template beforehand to simplify the correlation calculations.

5.3 Algorithm Steps

The Fast NCC algorithm follows these steps:

- Precompute the template statistics, including the mean and sum of squared deviations.

- Compute the integral images for both the input image and the squared pixel intensities.
- For each possible position of the template within the image:
  - Use the integral images to quickly compute the mean and variance of the corresponding image patch.
  - Calculate the zero-mean normalized cross-correlation between the template and the image patch.
- Store the correlation results in an output map, which represents how well the template matches different regions in the image.

5.4 Advantages of Fast NCC

The Fast NCC algorithm achieves a significant speedup compared to vanilla NCC due to:

- Constant time calculations for sums and variances using integral images, instead of linear time operations.
- Reduced number of arithmetic operations by avoiding redundant calculations for overlapping image regions.
- Efficient handling of large templates and images, making it suitable for real-time applications.

The optimized approach allows for quick identification of template matches within large images, making it an effective solution for various tasks in computer vision, including object tracking, face detection, and medical imaging.

5.5 Complexities

The computational complexity of template matching using the traditional Normalized Cross-Correlation (NCC) and the optimized Fast NCC differ significantly, primarily due to the use of integral images in Fast NCC.

search window(s)	length	direct NCC	fast NCC
168×86	896 frames	15 hours	1.7 hours
$115 \times 200, 150 \times 150$	490 frames	14.3 hours	57 minutes

Figure 15: These times were calculated on a 100 Mhz R4000 processor. The window size represents the template and the length represents the number of images that had to be cross-correlated.

5.6 Code Implementation

The provided code demonstrates an implementation of Fast NCC using precomputed sums and integral images:

```
# Function to compute fast NCC using precomputed sums
def fast_ncc(img, template):
    h, w = template.shape
    img_h, img_w = img.shape

    # Precompute template values
    template_mean = np.mean(template)
    template_zero_mean = template - template_mean
    template_sum_square = np.sum(template_zero_mean ** 2)

    # Compute integral images (sum of pixel intensities and squared pixel intensities)
    integral_img = np.cumsum(np.cumsum(img, axis=0), axis=1)
    integral_img_square = np.cumsum(np.cumsum(img**2, axis=0), axis=1)

    ncc_map = np.zeros((img_h - h + 1, img_w - w + 1))

    for i in range(img_h - h + 1):
        for j in range(img_w - w + 1):
            # Define the region (i:i+h, j:j+w) using integral image
            sum_region = integral_img[i + h - 1, j + w - 1]
            sum_square_region = integral_img_square[i + h - 1, j + w - 1]

            if i > 0:
                sum_region -= integral_img[i - 1, j + w - 1]
                sum_square_region -= integral_img_square[i - 1, j + w - 1]
            if j > 0:
                sum_region -= integral_img[i + h - 1, j - 1]
                sum_square_region -= integral_img_square[i + h - 1, j - 1]
            if i > 0 and j > 0:
                sum_region += integral_img[i - 1, j - 1]
                sum_square_region += integral_img_square[i - 1, j - 1]

            # Compute mean and variance from the integral image
            img_mean = sum_region / (h * w)
            img_variance = sum_square_region / (h * w) - img_mean ** 2

            # Calculate the normalized cross-correlation
            img_patch_zero_mean = img[i:i + h, j:j + w] - img_mean
            numerator = np.sum(img_patch_zero_mean * template_zero_mean)
            denominator = np.sqrt(template_sum_square * img_variance * h * w)

            # Avoid division by zero
            if denominator == 0:
                ncc_map[i, j] = 0
            else:
                ncc_map[i, j] = numerator / denominator

    return ncc_map
```

References

[1] JP Lewis. “Fast normalized cross-correlation”. In: *Vision Interface*. 1995.