Computer Vision Assignment 1

Q1 Histogram Equalization

Histogram equalization is a technique used in image processing to enhance the contrast of an image. It works by redistributing the intensity values of pixels in an image so that the histogram of pixel values becomes more uniform. This makes dark regions lighter and bright regions darker, resulting in better contrast and visibility of details in the image.

Step-by-Step Explanation of the Code:

1. **Importing Libraries:**
    - Utilized libraries like numpy for array manipulation, matplotlib for plotting, and skimage to read the input image.
2. **PDF Function (Probability Density Function):**
    - The function computes the histogram of the image (how frequently each pixel intensity appears) and normalizes it to form a PDF, which represents the likelihood of each pixel intensity value.
3. **CDF Function (Cumulative Distribution Function):**
    - The function computes the cumulative sum of the PDF. The CDF tells us how the pixel values accumulate from the darkest to the brightest in the image.
4. **Histogram Equalization:**
    - In this function the original pixel values are remapped according to the CDF. This ensures that pixel intensities are spread out more evenly across the range (0-255 for grayscale images), resulting in an image with improved contrast. We call both the functions, PDF and CDF and utilize the values.
    - The np.interp function is used to interpolate and assign new intensity values to each pixel based on the CDF.
5. **Plotting PDF and CDF:**
    - The code plots the original and equalized PDFs and CDFs to visually compare the distributions before and after equalization. The original image may have pixel values concentrated in certain ranges, whereas the equalized image has a more balanced distribution.
6. **Displaying Images:**
    - The original grayscale image and the equalized image are displayed side by side to show the effect of histogram equalization on contrast enhancement. We create and plot the graphs, and we use axs[x, y] to determine the position of the graph in output.
    - In the end, we display the original and the equalized image side by side for comparison.

Key Words and Concepts:

- **Histogram:** A graphical representation of the distribution of pixel intensities in an image.
- **PDF (Probability Density Function):** Shows the probability of each pixel intensity occurring in the image.

- **CDF (Cumulative Distribution Function):** Accumulates the probabilities from the PDF, helping to map original pixel values to new ones for contrast adjustment.
- **Equalization:** The process of adjusting pixel intensities so that the image has a more uniform distribution of gray levels, improving visibility of image details.

Purpose of Histogram Equalization:

The purpose of this technique is to enhance the contrast of images, especially when the original image has poor visibility in certain regions. By spreading out pixel intensity values, it makes features of the image easier to distinguish. This method is particularly useful in applications such as medical imaging or satellite images where clarity and detail are crucial.

Q2 Image Thresholding

Thresholding is a crucial technique in image processing, allowing us to convert grayscale images into binary images by separating pixels into foreground (object of interest) and background. We will go through 2 of the most popular methods: **Otsu's Thresholding** and **Ni-Black's Thresholding**.

Step-by-Step Explanation:

1. **Importing Libraries:**
    - The code uses libraries like numpy for array manipulation, matplotlib for plotting, and skimage to read the input image.
2. **Otsu Thresholding Function:**
    - Otsu's method calculates the variance between the two classes (foreground and background) for each possible threshold value (from 1 to 254). It computes the weights of the two classes and their respective means.
    - Weight (w0, w1): Represents how many pixels belong to the background and foreground at each threshold.
    - Mean (mu0, mu1): Represents the average intensity in the background and foreground.
    - Inter-class variance: It is the product of the weights of the two classes and the squared difference between their means. This variance measures how well-separated the two classes are. Otsu's method selects the threshold that maximizes this variance.
3. **Ni-Black Thresholding:**
    - The method requires defining a window size. The window sizes chosen here are **15, 25, 35.**
    - For each pixel in the image, the algorithm calculates the local mean and standard deviation of the pixel intensities within the specified window. The local mean indicates the average intensity of the pixels within that region, while the standard deviation measures the spread of those intensities.

    - Padding adds a border around the image to prevent issues when computing local statistics at the edges. The reflect mode ensures that the border pixels mirror the edge pixels.
    - The local threshold for each pixel is calculated by adding a constant k (commonly suggested to set to -0.2) multiplied by the standard deviation to the local mean. The constant k adjusts the influence of the standard deviation on the threshold.
    - The final step is to create a binary image by comparing each pixel's intensity to its local threshold. Each pixel's intensity is compared to its calculated local threshold. If the intensity exceeds the threshold, the pixel is classified as foreground; otherwise, it is classified as background.
4. **Final Steps**: We then print the histogram of the image TEM followed by the plotting of the inter-class variance vs threshold and marked the chosen Otsu threshold. Then we finally display the image after Otsu Thresholding. The results for different window sizes and shapes are visualized for Ni-Black, displaying how the choice of parameters affects the binary segmentation of the image.

Key Words and Concepts:

- **Otsu's Thresholding**: A global thresholding technique that determines an optimal threshold value to separate the pixel intensities of an image into two distinct classes: foreground and background. It aims to maximize the inter-class variance, which quantifies the separation between the two groups of pixels.
- **Ni-black's Thresholding:** A local thresholding technique that computes a threshold for each pixel based on the pixel's local neighborhood. This method is especially useful for images with varying lighting conditions, allowing for better adaptation to local variations in intensity.
- **Local Thresholding**: Each pixel has a different threshold based on its local neighborhood, allowing for adaptive segmentation.
- **Window Size and Shape**: The dimensions of the area around each pixel used for computing the local mean and standard deviation.
- **Foreground and Background Classification**: The process of labeling pixels based on their intensity relative to the calculated threshold.
- **Global Thresholding**: A single threshold value is applied to the entire image.
- **Inter-Class Variance**: A measure of how separated the two classes (foreground and background) are based on the threshold.
- **Binary Image**: An image where pixels are either black or white, representing two classes.

Conclusion

Both Otsu's and Ni-Black's thresholding methods serve the critical function of segmenting images into foreground and background, each with its strengths and use cases. Otsu's method is effective for images with uniform lighting, while Ni-Black's method excels in scenarios with varying illumination.

Q3 Template Matching

Template matching is a computer vision technique used to locate a smaller image (the template) within a larger image. In this code we use **Normalized Cross-Correlation** to achieve this.

Step-by-Step Explanation of the Code:

1. **Importing Libraries**: Several libraries are utilized, including NumPy for numerical operations, Matplotlib for visualization, skimage for image loading, and time for measuring execution speed.
2. **Normalized Cross-Correlation Function:**
    - First, image and template are converted to grayscale if they are in color. This simplifies processing by reducing the data to a single channel.
    - Next, we store the dimensions of both the image and template and create an empty **result** array to hold the correlation scores, which will be smaller than the original image since the template must fit within it.
    - Then mean and standard deviation of the template are computed, which are essential for normalizing the cross-correlation calculations.
    - The loop performs **Cross-Correlation Calculation**. The algorithm slides the template over the image and extracts overlapping windows. For each window, it calculates the mean and standard deviation, then computes the normalized cross-correlation score. If both the window and the template have a standard deviation greater than zero, we compute the normalized cross-correlation. Each pixel in the window is subtracted by the window mean and divided by its standard deviation. The result is the average of the product of these normalized values. If the standard deviation is zero, we assign a correlation score of zero for that position. This normalization accounts for variations in lighting and contrast, improving the robustness of the matching process.
3. **Execution Time Measurement**: The time taken for the template matching process is measured to assess the algorithm's efficiency.
4. **Global Maximum Peak Identification**: The position of the highest correlation score is determined, indicating the best match of the template within the image.
5. **Adjusting Coordinates for Display:** We adjust the peak coordinates so that they represent the center of the matched area in the original image.
6. **Thresholding for Additional Peaks**: We define thresholds to filter the correlation scores, allowing us to identify additional areas in the image where the template matches. For this code we use 0.3, 0.5 and 0.7.
7. **Result Visualization**: The original image is displayed alongside marked points indicating where the template was found. The global peak is marked with a blue dot, and additional peaks above the thresholds are marked with red dots. The label defines that the blue point is a global point.
8. **Overlaying Results**: A rectangle is drawn around the matched area in the original image to visually highlight the location of the template.

9. **Displaying Cross-Correlation Output with Peak Markers**: The cross-correlation output (resulting from the comparison of the template with different areas of the image) is visualized using a heatmap (cmap = hot), where colors represent the correlation intensity at each position. Brighter regions indicate stronger matches. The global peak is identified and marked with a circle on the heatmap. This provides a clear visual representation of where the template matches most accurately in the image.
10. **Thresholded Image Overlay on the Original Image**: A thresholded image is created by applying a threshold (taken 0.5 in this case) to the correlation output. This results in a binary image where pixels with correlation values greater than 0.5 are marked as True while others are False. The thresholded areas are then overlaid onto the original image. For each position that exceeds the threshold, a red dot is placed on the original image to indicate where the template matches the image above this threshold.

Key Words and Concepts:

- **Cross-Correlation**: A mathematical operation used to measure how well two signals or images match by sliding one signal (template) over the other (image) and calculating a similarity score at each position.
- **Normalized Cross-Correlation**: This is a more robust version of cross-correlation where both the template and the image region are normalized (subtract the mean and divide by the standard deviation) before matching, which makes the process invariant to changes in brightness and contrast.
- **Sliding Window**: A method where the template is systematically moved across the original image, and at each position, the cross-correlation value is computed to determine the level of similarity.
- **Global Peak**: The highest value in the correlation matrix, representing the point where the template matches the image most closely. This is identified by finding the maximum value in the correlation output.
- **Thresholding**: Applying a threshold to the cross-correlation matrix to highlight areas where the correlation score is above a certain value. This helps in identifying multiple regions of potential matches.
- **Non-Maximum Peaks**: Peaks in the cross-correlation matrix that are lower than the global peak but still represent areas with significant similarity to the template.
- **Heatmap**: A color-coded representation of the cross-correlation matrix that visually displays areas of higher or lower correlation, making it easier to spot peaks.
- **Mask**: A binary image or matrix where certain areas are marked (typically using thresholding) to highlight regions of interest. In this context, the mask highlights regions of the original image where the cross-correlation score exceeds a certain threshold.

Conclusion

When used the non-normalized cross-correlation method, the results were not as expected. This could be because cross-correlation doesn't account for variations in intensity or contrast between the template and the image. It does not factor in local variations within the image or template. Cross-correlation simply sums up the product of pixel values over the template and image region without adjusting for local differences in brightness or contrast.

The code when employed with normalized cross-correlation has worked as expected by detecting the template accurately. By normalizing both the template and the image window, it compensates for variations in lighting and contrast, which enhances the robustness of the matching process. This technique is foundational in computer vision and can be applied in various domains, such as object detection and image recognition, by effectively identifying and localizing templates within larger images.

Q4 Creative Section

The algorithm used is the Sum of Absolute Differences (SAD) method. This method works by comparing pixel values between the template and regions of the target image to find the best match. SAD typically employs a sliding window approach, where the template is moved across the image, and the score is computed for each position.

Step-by-Step Explanation of the Code:

1. **Importing Libraries**: Several libraries are used like NumPy is used for efficient numerical operations, Matplotlib is for visualizing results, and skimage is used for loading and converting images. The time library is also imported to measure the execution speed of the template matching process.
2. **Computing the Integral Image**: The function takes a grayscale image as input and returns its integral image. It uses the np.cumsum function to calculate the cumulative sum along both dimensions of the image.
3. **Calculating the Sum of a Region**: The function computes the sum of pixel values in a rectangular region of the integral image defined. It uses the top left and bottom right coordinates to compute the sum, leveraging the properties of the integral image for efficient calculations.
4. **Sum of Absolute Differences (SAD) Calculation**: The function calculates the SAD score to assess how closely the template matches different regions of the image. The function slides the template over the image, computes the sum of the region corresponding to the template, and calculates the SAD score by comparing it with the sum of the template. The coordinates of the best match are updated when a lower score is found.
5. **Template Matching Function:** The function accepts files for both the original image and the template, loads the images, and converts them to grayscale if they are colored. A

timer is initiated to measure execution time for reference and the sad function is called to find the best match. The function returns the coordinates of the bounding box around the matched template.

6. **Display Function:** The function visualizes the original image with a bounding box around the detected template. It draws a green rectangle around the matched area, sets a title, and hides the axes for improved visual clarity.

Key Words and Concepts

1. **Integral Image**: It allows for quick computation of the sum of pixel values in any rectangular region of an image. It is constructed such that each pixel value in the integral image at position (x, y) contains the sum of all pixels above and to the left of (x, y) in the original image. This enables efficient calculation of pixel sums within any rectangular area, which is crucial for speeding up the SAD computation.
2. **Sum of Absolute Differences (SAD)**: A method used to measure the similarity between two images or image regions. It computes the absolute differences between corresponding pixel values of the template and a region of the image and sums them up. The lower the SAD score, the better the match.

Conclusion

The implementation of template matching using the Sum of Absolute Differences method leverages the efficiency of integral images for rapid region sum calculations. By utilizing the sliding window technique, the algorithm effectively scans the original image to locate the template. It is a straightforward and efficient technique for template matching in image processing. By calculating the absolute differences between pixel values in a template and corresponding regions of an image, SAD quantifies their similarity effectively.

The SAD method would be significantly faster than the normalized cross correlation method that was used in question 3 (in this case). Even though SAD was faster, the Normalized cross correlation method is more robust as SAD could perform inferior due to variations in lighting and contrast, which can affect matching accuracy.