CENG 466 Introduction to Image Processing Programming Assignment 3

Image Processing Techniques: Image segmentation using gray-scale morphology and K-means clustering

# Cas Carree

*Student ID: 2704930*

***Abstract*—** **Image segmentation is a critical step in image analysis, enabling the partitioning of an image into distinct regions based on pixel characteristics. This study investigates the performance of three segmentation methods—grayscale morphology, K-means clustering with RGB features, and K-means clustering with Local Binary Pattern (LBP) features—on various image types requiring distinct segmentation objectives. The target features include primary patterns on rugs, visible fabric wrinkles, and zipper detection. Each method was evaluated on its ability to isolate the desired features across six images effectively, with 2 images for each target feature.**

***Keywords*— segmentation, grayscale morphology, K-means, clustering, Local Binary Pattern**

1. Grayscale morphology
2. *Background*

Morphological operations in image processing are used to analyze and manipulate the structure of objects in an image, enabling tasks like noise reduction, edge detection, feature extraction, and shape-based segmentation [1]. Grayscale morphology employs fundamental operations such as erosion, dilation, opening, and closing, applied to grayscale images. These operations use a structuring element to modify the local pixel intensities of the image. Erosion reduces the intensity of regions near object boundaries, while dilation increases the intensity by expanding brighter areas [2]. Opening, is a combination of erosion followed by dilation, which removes small noise and irregularities, whereas closing, the reverse sequence, fills gaps and smooths boundaries [2]. A significant advantage of grayscale morphology is its ability to preserve the geometric structure of image features while manipulating their intensity levels. For example, Morphological gradients, calculated as the difference between the dilated and eroded versions of an image, are used for edge detection by highlighting boundary details [2]. Similarly, top-hat and black-hat transformations highlight small-scale features and suppress background structures, improving feature isolation by comparing the input image to the opened image for top-hat and closed image for black-hat. Despite its strengths, the choice of structuring element size, shape, and intensity threshold remains critical, as these parameters significantly affect the outcome.

1. *Methodology*

The following methodology was implemented to preprocess and analyze the images using morphological operations, smoothing techniques, and histogram analysis.

**I**mage Loading and Preprocessing

1. Image Acquisition: the six images were loaded using the cv2.imread function, and each image was converted from BGR to RGB color space using cv2.cvtColor for compatibility with visualization libraries such as Matplotlib.
2. Grayscale Conversion: Each image was converted to grayscale using cv2.cvtColor, ensuring efficient processing for morphological operations and histogram analysis.

Histogram Analysis

To analyze pixel intensity distribution:

1. Grayscale histograms were computed for each grayscale image using the cv2.calcHist function with 256 bins, spanning the intensity range [0, 255].
2. These histograms were plotted to observe the distribution of pixel intensities, helping identify the characteristics of the image.
3. Based on the histograms and the results of the morphological operations, the following preprocessing steps were utilized to increase performance: image 2 and image 3 were blurred using cv2.Gaussian blur with the corresponding kernel sizes of 3x3 and 7x7. Image 3 was cropped from 225x255 to 199x199 to remove noise at the edges of the image.

Morphological Operations

The following Morphological operations were employed to emphasize specific features in the images: Erosion and Dilation: To shrink or expand object boundaries. Opening and Closing: To remove small noise (Opening) and close gaps (Closing) within the object. Gradient: To highlight the edges of objects. Top-Hat and Black-Hat: To extract bright and dark regions relative to the surrounding background. As parameters the structuring element of shapes cv2. MORPH\_ELLIPSE, cv2. MORPH\_CROSS, or cv2. MORPH RECT was used with varying kernel sizes (e.g., 3x3, 5x5).

1. *Results and Limitations*

The analysis was conducted using a combination of parameter configurations, including kernel sizes of 3×33 \times 33×3 and 5×55 \times 55×5, and structuring element shapes of ellipse, cross, and rectangle. Performance evaluation was primarily based on visual inspection. Among the tested configurations, the kernel size 3×33 \times 33×3 demonstrated superior performance, with no significant variations observed across the different structuring element shapes. The gradient operator proved to be the most effective tool for image segmentation, although the black-hat and top-hat transformations also yielded satisfactory results. Notably, Image 6 presented the greatest segmentation challenge due to the high-texture background containing numerous edges. To address this, a smoothing operation was applied, which significantly enhanced segmentation performance

1. K-means on RGB features
2. *Background*

Among the many techniques available for image segmentation, K-means clustering has emerged as a versatile and widely used method for segmentation due to its simplicity. In particular, K-means segmentation using RGB features leverages the color information within an image to group pixels into meaningful clusters [3]. K-means clustering operates by initializing k clusters and then minimizing the intra-cluster variance, by iteratively assigning pixels to the nearest cluster based on their RGB intensity values and recalculating cluster centroids until a stopping criteria is reached [3]. By treating each pixel's RGB intensity as a three-dimensional vector, K-means clustering effectively captures the color-based similarity among pixels. The primary advantage of using RGB features in K-means segmentation lies in its ability to process color images without requiring complex transformations or feature engineering. Moreover, the technique is robust against simple variations in lighting and texture, providing reliable results for segmenting objects with visually distinguishable colors. However, its performance can be affected by factors such as overlapping color distributions between regions and sensitivity to initial cluster centroids. Therefore parameter tuning, including selecting the optimal number of clusters and initialization strategies, is critical to achieving high-quality segmentation results.

1. *Methodology*

K-means Clustering for Image Segmentation

Image Reshaping and Initialization:

Images were reshaped into a two-dimensional array of pixel data of size (num\_pixels,3), where each row represented the RGB color values of a pixel.

Pixel values were converted to the 32-bit floating-point format to meet the input requirements of the K-means clustering algorithm.

K-means Clustering Algorithm:

The clustering process was implemented using scikit-learn’s KMeans class with nclusters= 2 to 5, depending on the application.

Algorithm parameters included n\_init​=10 (number of initializations) and a maximum of 300 iterations to ensure convergence to an optimal clustering solution.

Cluster centroids were computed to represent the dominant colors or features within each image.

Segmented Image Reconstruction:

Pixel labels generated by K-means were mapped to their corresponding cluster centroids or a predefined set of distinct colors.

Segmented pixel data was reshaped back to the original image dimensions, resulting in a visual output highlighting the segmented regions.

Visualization and evaluation:

The original and segmented images were displayed side-by-side using Matplotlib for qualitative assessment of the segmentation outcomes.Segmentation performance was analyzed based on visual clarity and the algorithm’s ability to isolate desired features.

1. *Results and Limitations*

The K-means clustering algorithm was applied to segment six images, with the number of clusters tested in the range of 1 to 10 to identify the optimal configuration for each image. For Images 1 and 2, nclusters= 2 provided the best segmentation results, while Images 3 and 4 performed optimally with nclusters=3. For Images 5 and 6, nclusters​=5 corresponded to the most effective segmentation. To address noise artifacts observed at the right and bottom edges of image 3, cropping was performed, reducing the dimensions from 255×255 to 199×199. This preprocessing step reduced noise but did not fully eliminate it. Image 6 remained particularly noisy due to its uniform color distribution and lack of clear distinction between object and background, resulting in suboptimal segmentation. In contrast, Image 3 exhibited the best segmentation performance, attributed to the distinct color separation between the object and background. For images 3 and 4, clusters 2 and 3 corresponded to the object, while in images 5 and 6, cluster 5 identified the object. These findings highlight the dependence of segmentation success on image characteristics such as color contrast and noise levels.

1. K-means using local binary patterns
2. *Background*

Local Binary Pattern (LBP) is an image processing tool representing textures. LBP effectively captures local patterns by comparing the intensity of a pixel with its surrounding neighbors, encoding the resulting binary comparisons into a single value. This simplicity and computational efficiency have made LBP a widely used feature extraction technique for tasks such as texture classification, face recognition, and object detection.LBP operates by defining a circular neighborhood around each pixel, comparing the intensity of neighboring pixels to the center pixel, and assigning a binary value of 1 or 0 based on whether the neighbor is greater than or equal to the center [4]. The resulting binary string is converted to a decimal value, creating a texture code that describes the local spatial structure[4]. One of the key advantages of LBP is its invariance to monotonic grayscale changes, such as lighting variations, which makes it robust for real-world applications [5]. Additionally, LBP can be extended to multi-scale analysis by varying the radius and number of neighbors, enabling it to capture both fine and coarse texture patterns [5]. Due to these advantages and its efficiency it remains one of the most widely used tool for representing textures.

1. *Methodology*

Image Preprocessing:

Images are read in RGB format and converted to grayscale using cv2.cvtColor. Grayscale conversion is essential for computing LBP features, which rely on intensity differences.

Feature Extraction using LBP:

The local\_binary\_pattern function from skimage.feature computes LBP features, capturing texture information based on a specified radius and number of circularly symmetric neighbors. The var and uniform methods were used. In the var variant the LBP output is normalized using cv2.normalize to ensure feature values are within a consistent range, addressing issues with NaN or infinite values.

K-means Clustering:

The flattened LBP features are clustered into k=10k = 10k=10 clusters using KMeans from sklearn.cluster. The algorithm groups similar texture patterns, with random\_state, n\_init, and max\_iter ensuring consistency and stability.

Cluster Visualization:

Cluster labels are mapped to predefined colors (distinct\_colors array) for visual clarity.

The segmented image is reshaped to its original dimensions and displayed alongside the original image using matplotlib.pyplot.

1. *Results and limitation*

Various combinations of parameters were tested, including the number of clusters (k= 1 to 10), LBP radius (r= 2,4,8), and the number of points (p= 4,8,16). Due to time constraints, an exhaustive search was not feasible. Instead, the search was terminated once a satisfactory result was obtained.

* **Images 1 and 2**: The uniform LBP method with k=2, r=10, and p=8 yielded the best segmentation results, likely due to the distinct texture patterns present in these images.
* **Images 3 and 4**: Despite testing various configurations, no meaningful segmentation was achieved. The uniform method with k=2, r=4, and p=8 provided the least suboptimal outcome. This is attributed to the lack of clear texture distinction between the object and the background in these images.
* **Images 5 and 6**: The variance-based LBP method performed better, with r=2, p=4, and k=10 producing the most accurate segmentation. The uniform method was less effective for these images.

1. conclusion

This study evaluated three segmentation approaches—grayscale morphology, K-means clustering using RGB features, and K-means clustering with Local Binary Pattern (LBP) features—across six images representing distinct segmentation challenges. The results highlighted the strengths and limitations of each method in isolating desired features such as rug patterns, fabric wrinkles, and zippers.Grayscale morphology proved effective for edge-based and shape-oriented segmentation tasks, with the gradient operator demonstrating the highest utility. However, the technique struggled with high-texture images where object-background boundaries were unclear. K-means clustering using RGB features performed well in images with distinct color separation but was less effective when object and background colors overlapped. Cropping and smoothing improved results but did not fully resolve segmentation challenges in certain cases.

K-means clustering with LBP features leveraged texture information for segmentation. While the uniform LBP method excelled in images with pronounced texture patterns (e.g., Images 1 and 2), it underperformed in images lacking clear texture distinctions. The variance-based LBP method provided better results, particularly for images 5 and 6. Overall, the study demonstrated that the choice of segmentation method and parameter configuration significantly impacts performance and must be tailored to the specific characteristics of the target features and images**.**

# references:

|  |  |
| --- | --- |
| [1] | T. Walche, „roboflow.com,” 3 9 2024. [Online]. Available: https://blog.roboflow.com/morphological-operations/. |
| [2] | opencv, [Online]. Available: https://docs.opencv.org/4.x/d9/d61/tutorial\_py\_morphological\_ops.html. [Geopend 06 01 2025]. |
| [3] | geeksforgeeks, „geeksforgeeks.org,” 09 02 2023. [Online]. Available: https://www.geeksforgeeks.org/image-segmentation-using-k-means-clustering/. |
| [4] | A. Rosebrock, „pyimagesearch.com,” 07 12 2015. [Online]. Available: https://pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/. |
| [5] | T. Ojala, „ieeeexplore.ieee.org,” 07 06 2002. [Online]. Available: https://ieeexplore.ieee.org/document/1017623/. |