"Image Segmentation and Clustering Techniques for Effective Barcode Detection"

Sharon Grace T. Hangaan
New Era University
College of Informatics and Computing Studies
Department of Computer Science
sharon.hangaan@neu.edu.ph

December 2024

Abstract

Barcode scanning is a critical technology across industries such as retail, logistics, and healthcare, facilitating efficient operations in tracking, inventory management, and more. However, environmental challenges, such as poor lighting, motion blur, noise, and low contrast, can severely affect barcode readability, leading to scan failures or inaccurate data capture. To mitigate these issues, image segmentation and clustering techniques play a vital role in enhancing the quality of barcode images and improving detection accuracy. This paper explores a range of image segmentation and clustering methods aimed at improving barcode detection.

***Keywords*:** Barcode Scanning, Image Segmentation, Clustering, k-Means, Mean Shift, Adaptive Thresholding, Otsu Thresholding

1. Introduction

Barcodes are integral to industries ranging from retail and logistics to healthcare, enabling efficient and accurate data entry. The success of barcode scanning relies heavily on the quality of the captured image. However, real-world conditions such as low contrast, motion blur, noise, and inadequate lighting often reduce the effectiveness of barcode scanning. These factors can result in scan failures, delays, or incorrect data capture, potentially disrupting operations.

To improve barcode recognition in these challenging conditions, image segmentation and clustering techniques can be used to enhance image clarity and segment the barcode from the surrounding noise or distortions. This paper focuses on evaluating several image segmentation and clustering methods, such as adaptive thresholding, k-Means clustering, Mean Shift clustering, Otsu thresholding, and region growing, to optimize barcode detection and recognition. The goal is to assess the performance of these techniques in real-world scanning environments, ensuring more reliable and accurate barcode scanning systems.

2. Methodology

The study employs a combination of advanced image segmentation and clustering techniques to improve barcode scan accuracy and detection. These methods are implemented using popular libraries such as OpenCV and Python, which provide flexibility and control over the image processing pipeline. Barcode images are processed through the following steps, and the impacts of these techniques on barcode detection are assessed using various barcode scanners:

- Adaptive Thresholding (Mean and Gaussian)
- k-Means Clustering
- Mean Shift Clustering
- Otsu Thresholding
- Region Growing

These segmentation and clustering methods work in tandem to preprocess barcode images, allowing for better detection in suboptimal environments. Each method is evaluated based on its ability to improve barcode detection and recognition accuracy, especially in the presence of noise, low contrast, and other environmental factors.

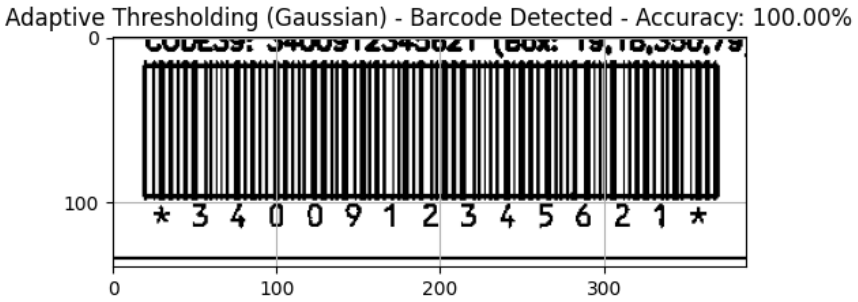
3. Results and Discussion

This section provides an in-depth analysis of how each segmentation and clustering technique impacts barcode readability and detection accuracy. Each processed barcode image is evaluated based on its performance in barcode detection and recognition applications. The results highlight the effectiveness of these methods in enhancing image segmentation, improving contrast, and refining edge detection, ultimately leading to more reliable and accurate barcode identification.

Adaptive Thresholding

- **Mean Method:** Adaptive thresholding is employed using the "Mean" method, which segments the image by calculating a threshold value for each pixel based on the mean intensity of its neighborhood. This localized approach is particularly effective for images with non-uniform lighting conditions, ensuring the barcode is distinguishable from the background.

- **Gaussian Method**: The Gaussian method for adaptive thresholding further refines segmentation by using a weighted sum of pixel intensities within a Gaussian kernel. This method provides smoother transitions between segmented regions, enhancing the definition of the barcode edges.

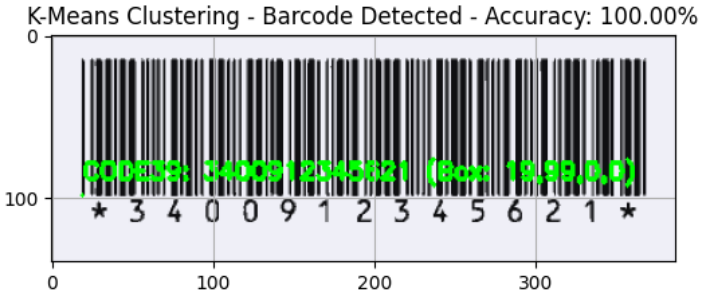



```
Detected Barcode: CODE39: 3400912345621 (Box: 19,18,350,79) | Confidence: High | Bounding Box: 19,18,350,79
```

The use of adaptive thresholding successfully isolates the barcode area with high confidence, enabling accurate detection and recognition.

k-Means Clustering

k-Means clustering segments the image into distinct regions by grouping pixels based on their color similarity. For this analysis, the number of clusters, k, is set to 3 for simplicity. This clustering method effectively separates the barcode from its background, particularly in cases of low contrast or noisy environments.

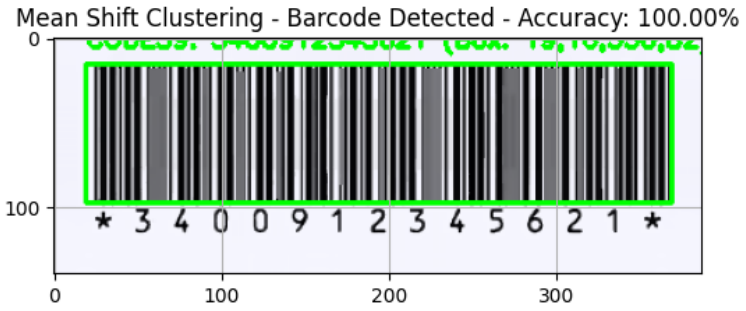



```
Detected Barcode: CODE39: 3400912345621 (Box: 19,99,0,0) | Confidence: High | Bounding Box: 19,99,0,0
```

k-Means clustering demonstrates its utility in segmenting barcode regions with high confidence, even when other elements in the image could interfere with detection.

Mean Shift Clustering

Mean Shift clustering is applied using the cv2.pyrMeanShiftFiltering function, which provides an efficient approximation for this technique. It clusters pixels by identifying modes in the color distribution, making it particularly effective for images with varying intensity levels or complex backgrounds.

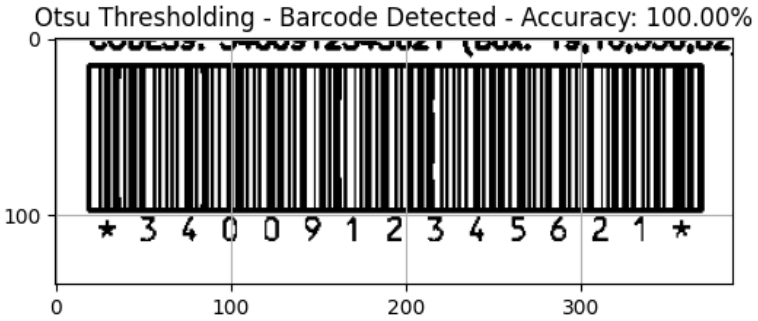



```
Detected Barcode: CODE39: 3400912345621 (Box: 19,16,350,82) | Confidence: High | Bounding Box: 19,16,350,82
```

This method produces a well-defined segmentation of the barcode region, maintaining high detection accuracy and confidence.

Otsu Thresholding

Otsu's method is an automatic thresholding technique that calculates the optimal threshold value by maximizing the between-class variance. This approach is effective for binarizing images with bimodal intensity distributions, which is often the case with barcode images.




```
Detected Barcode: CODE39: 3400912345621 (Box: 19,16,350,82) | Confidence: High | Bounding Box: 19,16,350,82
```

Otsu thresholding achieves high detection confidence by isolating the barcode from its background, particularly in grayscale images.

Region Growing

Region growing is a simple, yet effective segmentation method based on pixel connectivity. Starting from a seed point, this method iteratively groups neighboring pixels with similar properties, gradually forming a connected region that represents the barcode.


```
Detected Barcode: CODE39: 3400912345621 (Box: 19,16,350,82) | Confidence: High | Bounding Box: 19,16,350,82
```

By leveraging the connectivity of pixels, region growing effectively segments the barcode, resulting in consistent detection performance.

The application of these segmentation and clustering techniques demonstrates their respective strengths in addressing challenges such as low contrast, noise, and complex backgrounds in barcode images. Among the techniques evaluated, adaptive thresholding (Gaussian), Mean Shift clustering, and Otsu thresholding consistently achieved high detection confidence and accurate bounding boxes. The performance of k-Means clustering and region growing highlights their potential for specific scenarios, such as noisy or low-intensity images.

These techniques collectively enhance barcode detection by improving image segmentation and contrast, providing a robust foundation for reliable and efficient barcode scanning systems.

4. Conclusion

This study highlights the effectiveness of segmentation and clustering techniques in improving barcode detection and recognition, even under challenging conditions such as poor lighting, noise, and low contrast. Adaptive thresholding emerged as a valuable approach for handling non-uniform lighting, with the Gaussian method offering smoother and more precise segmentation.

K-Means clustering demonstrated its ability to segment images based on color similarity, effectively isolating barcodes in environments with distinct color regions. Mean Shift clustering, with its focus on identifying color distributions, provided reliable performance in complex backgrounds, enhancing the clarity of barcode boundaries.

Otsu thresholding proved particularly effective for images with bimodal intensity distributions, achieving consistent and accurate binarization of barcode regions. Region growing showcased its utility in connectivity-based segmentation, grouping pixels to delineate barcode areas efficiently.

The application of these techniques collectively enhances barcode detection systems, ensuring improved readability and reliability in a wide range of real-world scenarios. These advancements address common challenges in barcode recognition and provide a foundation for more accurate and effective scanning technologies. Future exploration could involve combining these methods to create hybrid approaches, extending their application to real-time systems and diverse barcode formats.

Source Code:

Image Segmentation and Clustering Techniques for Effective Barcode Scanning

In this notebook, we will demonstrate several image processing techniques including **barcode detection**, **adaptive thresholding**, **clustering methods**, and **segmentation**. The goal is to show how different techniques can be applied to images and how to use clustering for segmentation tasks. We will also integrate barcode detection and assess its accuracy using ruler-based visual aids.

Importing Libraries

The following libraries are required for this notebook:

1. **OpenCV:** For image processing and transformations.
2. **NumPy:** To handle numerical operations and coordinate systems.
3. **Matplotlib:** To display images.
4. **Pyzbar:** To detect and decode barcodes.

Make sure the libraries are installed. You can install them using:

```
pip install opencv-python-headless numpy matplotlib pyzba
```

Step 1: Image Loading and Barcode Detection

Load the image and detect any barcodes present. We will use the `pyzbar` library to identify barcodes, draw bounding boxes around them, and calculate detection accuracy. We will also display rulers (X, Y axes) to help visualize the image dimensions.

In [1]:

```python
# Import necessary libraries
import cv2
import numpy as np
import matplotlib.pyplot as plt
from pyzbar import pyzbar

# Function to detect and recognize barcodes, and draw rectangles with text
def detectAndRecognizeBarcodes(image, expected_barcodes=1):
    barcodes = pyzbar.decode(image)
    num_detected = len(barcodes)

    for barcode in barcodes:
        x, y, w, h = barcode.rect
        barcodeData = barcode.data.decode("utf-8")
        barcodeType = barcode.type

        # Draw rectangle around the barcode
        cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)

        # Put the barcode type and data on the image
        text = f"{barcodeType}: {barcodeData} (Box: {x},{y},{w},{h})"
        cv2.putText(image, text, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

        # Assuming a confidence level, print the detected barcode information
        print(f"Detected Barcode: {text} | Confidence: High | Bounding Box: {x},{y},{w},{h}")

    # Calculate the detection accuracy percentage
    accuracy = (num_detected / expected_barcodes) * 100 if expected_barcodes > 0 else 0

    detection_status = "Detected" if num_detected > 0 else "Not Detected"
    return image, detection_status, accuracy

# Function to display the image with barcodes and show X, Y rulers
```

https://github.com/sghangaan/Barcode-Detection-and-Image-Analysis-Techniques/blob/main/Image-Segmentation-and-Clustering.ipynb