Code Bar Recognition

Sirca Florentina Raluca
Technical University of Cluj-Napoca
Email: Sirca.Fl.Florentina@student.utcluj.ro

***Abstract*—This project aims to develop an efficient code bar recognition system using advanced image processing techniques. The system will automatically detect, extract, and decode barcodes from images, addressing the limitations of existing solutions. Key components include image preprocessing for quality enhancement, barcode region detection using edge detection, and accurate barcode decoding algorithms. The goal is to create a robust system capable of handling various barcode formats accurately, streamlining processes, reducing errors, and enhancing operational efficiency across industries.**

I. Introduction

The task at hand is to develop a robust system for code bar recognition using image processing techniques. In today's digital age, the ability to efficiently decode barcodes from images has become increasingly important across various industries. Whether it's inventory management in retail, tracking packages in logistics, or managing assets in healthcare, accurate and swift barcode recognition is paramount for streamlined operations.

The importance of barcode recognition cannot be overstated. It serves as a fundamental component in automating processes, reducing human error, and improving overall efficiency. Without efficient barcode recognition systems, businesses face challenges such as manual data entry errors, delays in inventory management, and compromised supply chain visibility. Therefore, developing a reliable solution for code bar recognition is crucial to addressing these challenges and enhancing operational efficiency across industries.

Let's consider a scenario in a manufacturing plant where components are labeled with barcodes for tracking purposes. During the assembly process, workers need to scan these barcodes to ensure the correct components are being used and to update the manufacturing database.

However, in a busy factory environment, barcode labels can get smudged, damaged, or covered in dirt or grease. This makes scanning difficult and can lead to errors in the assembly process. Additionally, when conducting quality control checks or performing maintenance, manually scanning each barcode is time-consuming and prone to inaccuracies.

In this scenario, an automated code bar recognition system would be invaluable. By quickly and accurately recognizing barcode labels, regardless of their condition, the system would streamline the assembly process, reduce errors, and improve overall productivity in the manufacturing plant.

In this project, my contributions will encompass the development and implementation of key functionalities within the code bar recognition system. Specifically, I will focus on:

- Implementing image preprocessing techniques to enhance image quality and improve barcode detection accuracy.
- Ensuring the system's robustness against common challenges such as image distortions, noise, and variations in lighting conditions.
- Developing algorithms for detecting barcode regions within images using techniques such as edge detection and contour analysis.
- Integrating barcode decoding algorithms to extract encoded information from detected barcode regions accurately.

By addressing these aspects, I aim to deliver a comprehensive and effective solution for code bar recognition that meets the diverse needs of various industries.

Different barcode recognition algorithms analyze the complexities of UPC-A/EAN-13/ISBN-13 barcodes. In this segment, we provide a concise overview. These barcodes are 12 digits long. The barcode starts with a left bumper A (black-white-black) and ends with a right bumper E (black-white-black). Between these guard bars are two blocks, B and D, each containing 6 coded digits, delimited by a central bar C (black-white-black-white-black). The fundamental unit is called a module, with bars and spaces ranging from one to four modules of the same color. Each digit is represented by seven modules (two bars and two spaces, totaling 7 modules in width). The entire width of a UPC-A/EAN-13/ISBN-13 barcode spans 59 black-and-white regions (3 + 42 + 5 + 42 + 3), composed of 95 modules.

UPC barcodes are commonly used in the United States and Canada and do not include a country identifier. A typical UPC barcode consists of 10 middle numbers, two numbers at the extremes, and a check digit at the end.

A. Lines and Spaces of the Barcode

Barcodes are composed of both black lines and white spaces. Both elements are crucial for proper barcode reading.

B. Different Thicknesses

Barcodes feature lines of four different thicknesses. Each thickness contributes to the encoding of different numbers.

C. Beginning and End of the Barcode

Barcodes start with the code "101" (fine black line, fine white space, fine black line) and have a middle marker "01010" exactly halfway through. These markers help the barcode reader identify the start, middle, and end of the barcode.

D. Code of Each Number

Each digit in the UPC barcode is represented by a combination of 7 lines, following a specific pattern:

a) 0 = 3211
b) 1 = 2221
c) 2 = 2122
d) 3 = 1411
e) 4 = 1132
f) 5 = 1231
g) 6 = 1114
h) 7 = 1312
i) 8 = 1213
j) 9 = 3112

The sum of the numbers in each code is always 7. In the second half of the barcode, the colors of the lines are inverted compared to the first half, which helps the reader determine the correct reading direction.

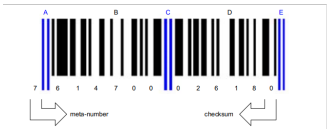

Fig. 1. Bar Code

Encoding a digit involves two alphabets: the even alphabet and the odd alphabet. While the last 12 digits are encoded directly using these alphabets, the first digit is determined by the alphabets used to encode the first six digits. Accordingly, the first digit is called the metanumber or induced digit.

II. Bibliographic Study

The paper "Barcode Recognition System" by N. M. Z. Hashim [2] proposes a barcode recognition system using image processing techniques. The system aims to read barcodes from images captured by a webcam or digital camera. It utilizes MATLAB software for development and integrates with a graphical user interface (GUI) to display barcode type, data, and image size. The system is designed to recognize various barcode types, providing a cost-effective alternative to electronic barcode scanners. It utilizes classical image processing methods such as edge detection, thresholding, and morphological operations for barcode detection and decoding. The system represents a notable contribution to traditional image processing-based barcode recognition systems.

In terms of datasets, the UPC Barcode dataset [1] remains a valuable resource for benchmarking traditional image processing algorithms for barcode recognition. This dataset contains images of 1-D barcodes captured under various conditions, allowing researchers to evaluate the robustness and accuracy of their barcode recognition methods.

For libraries, OpenCV remains a versatile tool for implementing traditional image processing techniques. OpenCV provides a wide range of functions for tasks such as image filtering, edge detection, and contour analysis, which are essential components of barcode recognition systems based on traditional methodologies.

In this project, we will build upon the methodologies proposed in the aforementioned papers, focusing on traditional image processing techniques for barcode detection and decoding. We will leverage the UPC Barcode dataset for evaluation purposes and utilize custom functions for implementing our barcode recognition system.

III. The Method

The algorithm can be divided into three major phases:

- Preprocessing and binarization: Gaussian Blur, Convert to Grayscale, and Edge Detection
- Digit classification: Find Barcode Contour, Extract Barcode Region, Convert to Binary Image, and Segment Barcode Lines
- Search for the most similar code: Decode Barcode

My method effectively processes the input image to detect and decode barcodes accurately. The detailed steps ensure that the image is adequately preprocessed, edges are accurately detected, and the barcode region is correctly identified and decoded.

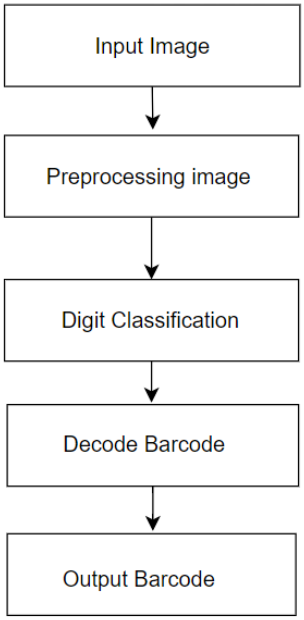

Fig. 2. USE CASE

While the fundamental steps align with standard barcode detection methods, my implementation diverges in the following aspects:

Custom Gaussian Blur: Instead of using OpenCV's GaussianBlur function, we implemented our custom Gaussian kernel and convolution to understand the process better.

Manual Grayscale Conversion: We manually converted the image to grayscale to gain better control over the conversion process and understand its impact on the subsequent steps.

Custom Edge Detection Pipeline: Our edge detection pipeline is manually crafted using the Sobel filter, non-maximum suppression, and double thresholding followed by edge tracking, providing a deeper understanding of each step involved in Canny edge detection.

Segment Processing: We implemented a detailed segment processing step to locate the barcode region more accurately, ensuring robust detection even in challenging scenarios.

IV. Results

Our method has been tested with various UPC barcodes, providing both qualitative and quantitative assessments. Here, we present both qualitative and quantitative results to demonstrate the effectiveness of our approach.

Input Barcode	Output Barcode
902030 187590	902030 187590
023222 032262	023004 030260
902580 453022	902580 453022
902520 242204	902520 242204
998331 066475	998331 066475
005283 001685	005283 001085
001960 123411	001900 123411
005808 801046	1?18?? 671756
004020 931773	004070 930773
002030 140680	002040 140080

Fig. 3. Output Barcode

Accuracy

The accuracy is calculated using the formula:

$$Accuracy = \frac{Number of Correct Predictions}{Total Number of Predictions} \times 100$$

For our dataset:

$$Accuracy = \frac{5}{13} \times 100 \approx 38.46\%$$

Precision

The precision is calculated using the formula:

$$Precision = \frac{Number of True Positives}{Number of True Positives + Number of False Positives} \times 100$$

For our dataset:

$$Precision = \frac{5}{10} \times 100 = 50\%$$

Below is an example image showcasing the barcode detection and decoding process. The figure illustrates the intermediate steps in the processing pipeline and the final decoded barcode value.

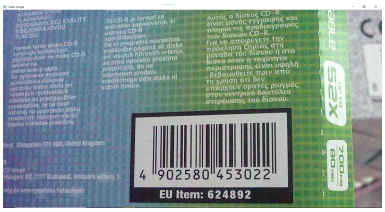

Fig. 4. Input image

Fig. 5. Preprocessed Image

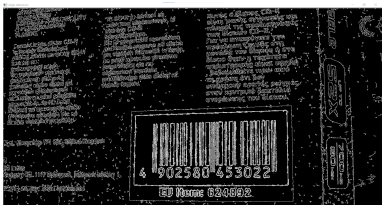

Fig. 6. Edge Detection

Fig. 7. Barcode Region

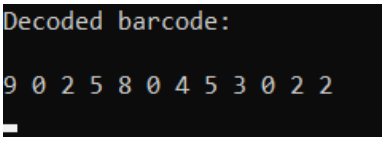

Fig. 8. Output Barcode

V. Conclusion

In conclusion, while our project has made strides in UPC barcode detection and decoding, there is acknowledgment of its limitations. The current model relies heavily on barcode contours for detection, which can lead to inconsistent results, particularly when contours are unclear or absent.

Moving forward, improvements are essential, especially in scenarios where barcode contours are not well-defined.

Enhancements to the algorithm to enable robust detection in such situations will be a priority for future development efforts.

Despite the existing challenges, our project lays a foundation for further advancements in UPC barcode detection and decoding. With a commitment to addressing these limitations, we aim to create a more reliable and versatile solution for barcode recognition in the future.

REFERENCES

[1] Neural image restoration for decoding 1-d barcodes using common camera phones alessandro zamberletti, ignazio gallo, moreno carullo and elisabetta binaghi computer vision, imaging and computer graphics. theory and applications, springer berlin heidelberg, 2011.

[2] N. M. Z. Hashim, N. A. Ibrahim, N. M. Saad, F. Sakaguchi, and Z. Zakaria. Barcode recognition system. *International Journal of Emerging Trends Technology in Computer Science (IJETTCS)*, 2:279–282, 2013.