Urban Scene Understanding with Text Recognition of Street View

Sagar S. Udgiri [1], *Pratik P. Mahajan* [1], *Onkar V. Narwade* [1], *Snehal V. Lokhande* [1]

[1]*Student, Dept. of Information Technology, Vidya Pratishthan's Kamalnayan Bajaj Institute of Engineering and Technology, Baramati, India.*

Corresponding Author: pratikmahajan16pm@gmail.com

Abstract: - In developing regions and cities, driving conditions can often be unpredictable and challenging due to difficulties in understanding the surrounding urban scenes. Intelligent Transportation Systems (ITS) can greatly benefit from the ability to comprehend the urban environment, particularly by leveraging the information provided by shop signboards and scene text. In this system, we used the method that utilizes urban scene images to extract valuable data for ITS applications. The system employs various image processing techniques to enhance the images and extract relevant text regions. To begin, the system converts the urban scene images into grayscale representations, which simplifies subsequent processing steps. Then the system employs contour detection techniques to identify potential text regions within the image. The Canny Edge Detection technique is then utilized to filter out non-text regions. Next, Optical Character Recognition (OCR) algorithms are employed to digitize the textual content within the identified text regions. The OCR process plays a crucial role in obtaining accurate and reliable textual information, which is vital for ITS applications. Finally, the system employs pyttsx3, a text-to-speech synthesis library, to convert the detected text into audible speech. By providing this capability, the system enhances accessibility by allowing users to listen to the extracted textual content, thereby enabling them to focus on the road while still receiving important information. By combining image processing techniques, contour detection, OCR, and text-to-speech synthesis, our developed system improves the performance of the understanding of urban scenes in developing regions and cities.

Key Words: - ***Image Processing, Text Recognition and Extraction, Canny Edge Detection Technique, Contours, Optical Character Recognition, pyttsx3.***

I. Introduction

In the context of developing regions and cities, driving conditions can be unpredictable due to difficulties in understanding the surrounding environment. To address this challenge and enhance Intelligent Transportation Systems (ITS), the ability to comprehend urban scenes and extract valuable information from them, such as shop signs and scene text, is crucial.

Manuscript revised May 27, 2023; accepted May 28, 2023. Date of publication May 29, 2023.

This paper available online at www.ijprse.com

ISSN (Online): 2582-7898; SJIF: 5.59

Our proposed system aims to leverage urban scene images to extract text from them, thereby contributing to safer and more efficient transportation. Understanding urban scenes through text recognition holds significant implications for various applications, including autonomous driving and ITS. By identifying and interpreting text elements from street views, we can gain valuable insights into the local geographic environment. This information becomes especially critical in autonomous driving scenarios where precise positioning is essential. Extracted text from shop signs and billboards can serve as anchor points for accurate localization and navigation, particularly in areas lacking digital map coverage or street view availability, such as small cities in mountainous regions, underdeveloped countries, or remote locations. Real-time text recognition from images of natural scenes, commonly referred to as 'Photo OCR,' has emerged as a prominent research area with numerous practical applications. The ability to extract text from images opens up possibilities for digital mapping, unmanned driving, scene understanding, and location-based services (LBS). Over the past decade, extensive efforts have been dedicated to advancing the field of text recognition from natural scenes, driven by the need for efficient and reliable solutions in various domains. To address these challenges, our system employs a combination of techniques, including contours, Canny Edge Detection, Optical Character Recognition (OCR), and conversion of extracted text into speech using a text-to-speech synthesizer. By leveraging these techniques, we aim to accurately extract text from shop signs and other textual elements present in urban scenes.

Motivation: Recent developments in the fields of image processing and natural language processing have been geared towards creating intelligent systems that would enhance quality of life. In ITS (Intelligent Transportation System) where Points of Interests are essential and to make a virtual map, we need to understand the urban scene so there should be a system which can help get the textual data of surrounding shop signs or boards. It can also help people with blindness to know the surroundings with voice message.

II. Literature Survey

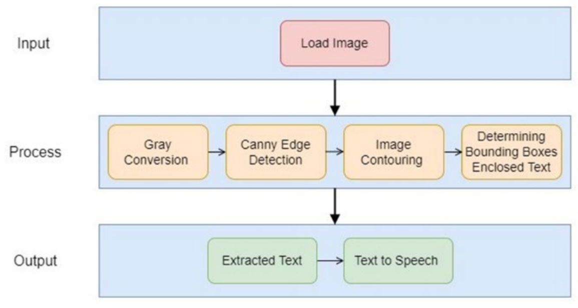
The concept of "Canny Edge Detection Based on OpenCV" was proposed by the authors Zhao Xu, Xu Baojie, and Wu Guoxin [2]. The Canny operator is extensively employed and enhanced in this conventional edge detection technique. Hysteresis threshold is used by Canny, and the various values of the threshold have a substantial impact on the detection result, although the number cannot intuitively describe the detection result. Through the use of OpenCV programming, four algorithms are presented. For edge detection, it is practical that the programming may visibly present the detection results under various thresholds.

The authors Weibin Rong, Zhanjing Li, Wei Zhang and Lining Sun [3] proposed "An Improved Canny Edge Detection Algorthim". This popular edge detection technique extensively employs and improves the Canny operator. Canny uses the hysteresis threshold, and the different values of the threshold have a significant impact on the detection result, even though the number alone cannot accurately convey the detection result. To introduce four algorithms, the OpenCV programming language is used. It is useful for the programmer to visibly provide the detection results for edge detection under different thresholds.

The authors Jack Greenhalgh and Majid Mirmehdi [4] proposed system for the automatic detection and recognition of text in traffic signs. This system locates a large number of candidates using hue, saturation, and value colour thresholding and maximally stable extremal areas. The candidates are then narrowed down using restrictions based on temporal and structural information. The temporal integration of text results across successive frames increases accuracy. This paper's methodology yields an overall Fmeasure of 0.87. To enhance OCR outcomes, perspective rectification and temporal fusion of candidate text sections were applied.
The research of Swati Vikas Kodgire, G. S. Sable [5] This study covers image capture, separation of the image's text part from its surrounding areas, image pre-processing on the region of interest, and then conversion of text into speech following character and word extraction. To separate text from a document, all potential manuscript text sections must be found. The processes that are carried out in order are text detection, line detection, character identification, feature extraction, and training of extracted features.

From Boris Epshtein, Eyal Ofek, Yonatan Wexler [6] We learned about the Stroke Width Transform, which combines dense estimate (calculated at each pixel) and non-local scope (stroke width depends on information sometimes present in very distant pixels). The algorithm can recognize texts in a wide variety of typefaces and languages because to the simplicity feature in this paper. The orientation of the recovered strokes should be taken into account while grouping the letters. This might also make it possible to recognize curved text lines.

2.1 System Architecture



Text Recognition of Boards

In the first phase of the system architecture, the "Input" phase, there is a module called "Load Image" that requests an image from the user through a graphical user interface (GUI). This module is responsible for interacting with the user and allowing them to select an image file to be processed by the system. It involves the following steps:

- The GUI presents an option for the user to load an image file.\\
- When the user selects the "Upload Image" option, the module initiates a file dialog, which prompts the user to select an image file from their computer.\\
- The file dialog allows the user to navigate through their file system and choose the desired image file.
- Once the user selects the image file, it is passed to the next phase of the system architecture for further processing.

The purpose of this module is to provide a user-friendly way for the system to receive input in the form of an image. By using the GUI and file dialog, the user can conveniently choose an image file from their local system, which can then be used for subsequent processing steps in the system.

In the second phase of the system architecture, the "Process" phase, several image operations are performed on the loaded image. These operations include:

2.1.1 Gray Conversion:

- The loaded image is converted from its original color representation (such as RGB) to grayscale.
- Grayscale conversion simplifies subsequent image processing steps by reducing the image to a single channel representing the intensity values.

2.1.2 Canny Edge Detection:

- Canny edge detection is applied to the grayscale image.
- This algorithm detects the edges present in the image by calculating the gradient magnitude of the image and identifying areas with significant changes in intensity.
- The output of this step is a binary image where edges are represented as white pixels and the background as black pixels.

2.1.3. Image Contouring:

- Contour detection is performed on the edge-detected image.
- Contours are the boundaries of objects or regions in an image.
- The algorithm identifies and extracts the contours present in the image.
- These contours can represent the boundaries of objects, including the text regions in the image.

2.1.4. Determining Bounding Boxes:

- Bounding boxes are determined for the identified contours.
- The algorithm calculates the minimum rectangle that encloses each contour.
- The resulting bounding boxes define the rectangular regions that likely contain text in the image.
- These bounding boxes can be used to isolate and extract the regions of interest (ROIs) where the text is located.

By performing these operations in the "Process" phase, the system prepares the image for further text extraction and analysis in subsequent phases. The grayscale conversion, Canny edge detection, contouring, and bounding box determination help identify and isolate the text regions, which can then be extracted and processed to obtain the textual content.

In the third phase of the system architecture, the "Output" phase, there are two operations: "The Extracted Text" and "Text-to-Speech".

2.1.5 The Extracted Text:

- After the text regions have been identified and extracted in the previous "Process" phase, the system performs optical character recognition (OCR) to extract the text from these regions.
- The extracted text is then displayed to the user, providing them with the textual content present in the image.
- This extracted text can be presented in a user interface element, such as a text box, allowing the user to view and read the text.

2.1.6 Text-to-Speech:

- In addition to displaying the extracted text, the system offers a "Text-to-Speech" functionality.
- The extracted text is converted into speech using a text-to-speech (TTS) engine.
- The system utilizes a module, such as "pyttsx3," to initialize the TTS engine and configure properties such as speech rate and voice volume.
- The TTS engine then processes the extracted text and converts it into audible speech.
- The synthesized speech is played back to the user, allowing them to listen to the content of the image.

By providing both the extracted text and the option for text-to-speech conversion, the "Output" phase ensures that the user can access the textual information in multiple formats—visual and auditory. This enhances the accessibility and usability of the system for users with different preferences and needs**.**

III. Algorithms

3.1 Canny Edge Detection:

Step 1: Convert the input image to grayscale.

- Let I(x, y) be the input image.
- Let G(x, y) be the grayscale image obtained from I(x, y).

Step 2: Apply Gaussian smoothing to reduce noise.

- Define a Gaussian filter mask with size k x k, denoted as H.
- Convolve G(x, y) with H to obtain a smoothed image, denoted as S(x, y).

Step 3: Compute the gradient magnitude and direction.

- Compute the gradient in the x-direction, denoted as Gx(x, y), using a suitable gradient operator.
- Compute the gradient in the y-direction, denoted as Gy(x, y), using the same gradient operator.
- Compute the gradient magnitude, denoted as M (x, y), using the Euclidean norm of Gx(x, y) and Gy(x, y).
- Compute the gradient direction, denoted as θ(x, y) , using the arctan function applied to Gy(x, y) and Gx(x, y).

Step 4: Apply non-maximum suppression to thin out the edges.

- For each pixel (x, y) in M(x, y):

∗ Determine the closest neighboring pixels in the gradient direction θ(x, y).
∗ Compare the magnitude M(x, y) with the magnitudes of the neighboring pixels.
∗ Set M(x, y) to 0 if it is not the maximum magnitude among the neighboring pixels.

Step 5: Apply double thresholding to classify pixels as strong, weak, or non-edges.

- Define two thresholds, Tlow and Thigh.
- For each pixel (x, y) in M(x, y):

∗ If M(x, y) is above Thigh, classify it as a strong edge pixel.
∗ If M(x, y) is between Tlow and Thigh, classify it as a weak edge pixel.
∗ If M(x, y) is below Tlow, classify it as a non-edge pixel.

Step 6: Perform edge tracking by hysteresis.

- Start with the strong edge pixels and trace along the weak edges connected to them.
- Mark the weak edge pixels that are connected to the strong edges as strong edges.
- Repeat this process until no more weak edges can be connected to the strong edges.

3.2 Contours:

Step 1: Convert the input image to grayscale.

- Let I(x, y) be the input image.
- Let G(x, y) be the grayscale image obtained from I(x, y).

Step 2: Apply pre-processing techniques to enhance the image quality.

- Perform noise reduction techniques, such as Gaussian smoothing or median filtering, to reduce noise in G(x, y).
- Apply contrast enhancement techniques, such as histogram equalization, to improve the visibility of text regions.

Step 3: Perform edge detection to identify potential contours.

- Apply edge detection algorithms, such as Canny Edge Detection or Sobel operators, to obtain the edges in G(x, y).

- Threshold the edge image to obtain a binary image, where edges are represented as white pixels and the background is black.

Step 4: Find contours in the binary image.

- Identify connected components in the binary image using algorithms like connected component labeling or contour tracing.
- Extract contours by tracing the boundaries of connected components.

Step 5: Filter and refine the contours.

- Apply filtering techniques based on contour properties, such as area, aspect ratio, or curvature, to remove noise and retain relevant contours.
- Refine the contours using techniques like approximation or smoothing to obtain smoother and more accurate contour representations.

Step 6: Extract bounding boxes enclosing text regions

- Calculate the bounding boxes that tightly enclose the contours using algorithms like minimum bounding rectangles or rotated bounding boxes.

IV. CONCLUSION

To understand the urban scene for intelligent driving system we have implemented the system. This system is designed to detect shop signs using Contours, Canny Edge Detection Technique, OCR (Optical Character Recognition). Using the Contours with Canny edge detection techniques will increase the accuracy of text detection. It is feasible to retrieve information quickly by using this strategy. After text extraction, we used Text to Speech Synthesizer to turn the text into speech.

REFERENCES

[1]. Chongsheng Zhang, Guowen Peng, Feifei Fu,Wei wang, "Street View Text Recognition With Deep Learning for Urban Scene Understanding in Intelligent Transportation Systems, IEEE Transactions On Intelligent Transportation Systems, Vol. 22, No. 7, July 2021.

[2]. Zhao Xu, Xu Baojie, and Wu Guoxin, “Canny Edge Detection Based on OpenCV.",2017 IEEE 13th International Conference on Electronic Measurement and Instruments.

[3]. Weibin Rong, Zhanjing Li, Wei Zhang and Lining Sun,"”An Improved Canny Edge Detection Algorithm”. Proceeding of 2014 IEEE International Conference on Mechatronics and Automation August 3-6.

[4]. Jack Greenhalgh and Majid Mirmehdi,"Recognizing Text-Based Traffic Signs", IEEE Transactions on Intelligent Transportation Systems, Vol. 16, No. 3, June 2015.

[5]. Swati Vikas Kodgire; G. S. Sable,"A Review on Optical Character Recognition and Text to Speech Conversion.", International Journal of Science and Research (IJSR) (Vol.5, No. 6).

[6]. Boris Epshtein, Eyal Ofek, Yonatan Wexler, “Detecting Text in Natural Scenes with Stroke Width Transform".

[7]. Y. Xu, Y. Wang, W. Zhou, Y. Wang, Z. Yang, and X. Bai, "Text Field: Learning a deep direction field for irregular scene 9 text detection, IEEE Trans. Image Process., vol. 28, no. 11, pp. 5566–5579, Nov. 2019.