Architecture for Text Sign Localization and Recognition

Arturo Romero, José Moreno, Fernando Fernández, Juan Manuel Montero, Rubén San-Segundo

Grupo de Tecnología del Habla. E.T.S.I.Telecomunicación.
Universidad Politécnica de Madrid
arturoromero@die.upm.es

Abstract

This paper describes a low complexity strategy for detecting and recognizing text signs automatically. Traditional approaches use large image algorithms for detecting the text sign, followed by the application of an Optical Character Recognition (OCR) algorithm in the previously identified areas. This paper proposes a new architecture that applies the OCR to a whole lightly treated image and then carries out the text detection process on the OCR output. The strategy presented in this paper significantly reduces the processing time required for text localization in an image, while guaranteeing a high recognition rate. This strategy will facilitate the incorporation of video processing–based applications into the automatic detection of text sign similar to that of a smartphone. These applications will increase the autonomy of visually impaired people in their daily life.

1 Introduction

The partial or complete loss of vision is a challenge that some people are forced to confront during their lives. Representing visual information using alternative senses (such as hearing or touch) may be a useful strategy in order to make easier some situations that visually impaired people have to deal with in the daily life. Detecting visual information automatically (using visual signal processing technologies) involves significant challenges in order to be useful for blind people. One of the most outstanding ones is the high power consumption for processing the visual information in real time. Because of this, many systems developed so far focus on specific detection problems such as the reading of car license plates, the shape of some objects, etc. One of the most time consuming processes is to locate, in the visual image, the target object (i.e. car license plate) before extracting its shape or content.

This paper presents the design and implementation of a system for detecting and recognizing text signs automatically and translating them into speech. This system focuses on relevant information that visually impaired people cannot read by themselves, for instance, the emergency exit sign in a building or the direction signs in the street. For these kinds of applications, the system has to be included in a mobile device such as a smart-phone. Because of these constraints, it is necessary to implement reduced detection algorithms in order to make the system work in real time. Instead of using lengthy image algorithms for detecting the text sign and then applying an Optical Character Recognition (OCR) algorithm in the previously identified areas, this paper proposes the application of the OCR in a whole lightly-treated image followed by the text detection process on the OCR output. The OCR output is a text output, so its processing is less time consuming.

When applying the OCR on a whole image, the OCR focuses on the features of the text itself, such as corners, strokes, edges, etc. Because of this, the OCR introduces an elevated number of mistakes giving rise to confusing elements in the image, for example the corner of a window, with a written character. The system proposed in this paper uses a localization algorithm, based on the idea of the text alignment methods that discharge the characters mistakenly introduced by the OCR, and detects the content of a specific text sign.

Finally, the system will produce both an audible and an enhanced visual interpretation (for partially visually impaired people) of the sign.

This paper is organized as follows. Section 2 presents a state of the art on what has already been achieved in the area of text detection in images. The system proposed in this paper is described in Section 3. Finally, the implications and the main conclusions of the work are described in Section 4.

2 Previous Work

Many text detection and localization approaches use features related to text characters (Chen et al, 2004). Then, the edge regions are grouped together based on the common features (Jirattitichareon, 2006).

Texture is another widely used feature for text detection and localization (Kim et al, 2003; Jung et al, 2000). First a texture analysis method such as Gabor filtering is used to extract the features of the texture. Then, a classifier is used to identify a region as a text or non-text based on these features. Support vector machines are used in (Kim et al, 2003; Jung et al, 2000) for this purpose. In (Huang and Ma, 2010), the authors use LoG-Gabor filters to obtain the stroke map and apply Harris corner detection to find the seed points for the connected component analysis.

Some approaches assume that the text is written in the horizontal or vertical direction (Sin et al, 2002; Wu et al, 2005). However, this constraint is not practical due to the possible tilt of the camera. Therefore, an affine rectification step is usually added to improve the results (Chen et al, 2004).

Text that appears in video frames is an important cue for the content analysis and indexing of the video. Accordingly, several approaches have been proposed for the detection and tracking of text in videos (Shivakumara et al, 2011; Yu and Wang, 2010). Some of these methods are used to detect the artificial text added to the video, based on the fact that this type of text has a homogeneous color as opposed to the complex color distribution of the scene (Yu and Wang, 2010). Other methods try to detect any type of text.

These methods usually have some common procedures for image treatment or text detection. Due to this situation, there are some implemented libraries or programs performing the basis image processing and character recognition.

A very well-known library for image processing is OpenCV (Open Source Computer Vision) (http://opencv.willowgarage.com/wiki/), provides a large amount of functions that allow the developers to work on a pixel level and implement higher level functions easily.

Among the character recognition world, Tesseract can be found as one of the best OCR programs. This open source engine was one of the top 3 engines in the 1995 UNLV Accuracy test. Between 1995 and 2006 it had little work done on it, but since then it has been improved extensively by Google and is probably one of the most accurate open source OCR engines available. Combined with the Leptonica Image Processing Library it can read a wide variety of image formats and convert them to text in over 40 languages (http://code.google.com/p/tesseract-ocr/).

3 Proposed approach

The system proposed in this paper can be divided into two big sections: hardware and software. The hardware section consists of a portable device and a camera, preferably being integrated at the back of the chosen device in order to locate text signs in one movement, if there are any. The software section is the main one in this project. A schematic representation of the process used to detect text signs are shown in Figure 1.

As shown in Figure 1, the system captures the image from the camera and performs an image treatment for converting a color image into a Black & White (B&W) image. Afterwards, the OCR algorithm looks for the text patterns contained in the whole B&W image, saving the text output into a .txt file. The text localization algorithm then detects whether or not the file given by the OCR contains any of the patterns included in the text sign database.

In a text sign database, the system stores all possible text signs that can be detected and recognized by the system. If any of the text patterns stored in the database is found, the system will provide both an audible and an enhanced visual representation. If not, the process for sign searching starts again with a new frame captured after a set time.

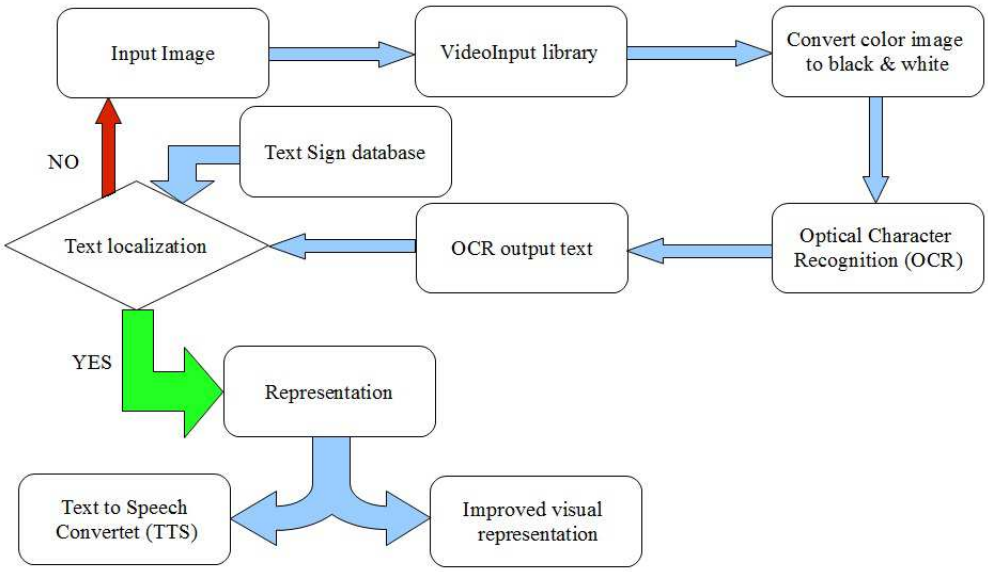

Figure 1. Process diagram

3.1 Capturing input image

First of all, the system must get the image from the required camera. If the system has several cameras, it is necessary to choose between them, keeping in mind some quality parameters, such as the camera resolution, the situation in the device (front or back), etc. As the system proposed is made up of a tablet with a webcam and an external high definition camera, the system will consider the camera with the highest resolution. To carry out this selection, the system takes advantage of the functions provided by the videoInput library (Web link: muonics.net/school/spring05/videoInput, last access Jan 2011) (written in C/C++ language).

3.2 Image treatment

Once the image from the camera is captured by the system, the image has to be converted into a B&W one. For this purpose, the OpenCV library (http://www.seas.upenn.edu/~bensapp/opencvdocs/ref/opencvref_cv.htm (last access Jan 2011)) provides the suitable tools to perform a useful treatment. Firstly, the image must be converted to a gray scale. The CvtColor function makes the conversion from the RGB color space to a gray scale as follows: RGB to Gray: $Y \leftarrow 0.299 \bullet R + 0.587 \bullet G + 0.114 \bullet B$

In order to obtain a right conversion, the image type must be squared up (8-bit, 16-bit or floating-point image).

Figure 2. Color image (left) and gray scale image (right).

From this gray scale image, the system is able to generate the B&W image that it will be used in the following steps of the process. To obtain this B&W image, the Threshold function extracts a bi-level (binary) out of the previous gray scale image the way the threshold type parameter indicates. Considering that the text in sign can be represented in positive (letters are darker than background) or negative forms (letters are lighter than background), the proposed system uses both possibilities: binary and binary inverted type (input and output shown in Fig 4), that correspond to the following equations respectively:

$$dst(x,y)=\begin{cases} max.value & if\ src(x,y)>threshold \\ 0 & otherwise \end{cases}$$

$$dst(x,y)=\begin{cases} 0 & if\ src(x,y)>threshold \\ max.value & otherwise \end{cases}$$

In the formulas, dst stands for destiny and src for source. In our case, maximum value is the integer 255 (white in an 8-bit space color image, 0 is black). Finally, the threshold is the value that defines the border between the pixels which will turn black and those turning white. This threshold is controlled by a scroll-bar (Figure 3) defined in the black and white window, being able to be adapted automatically depending on the ambient light.

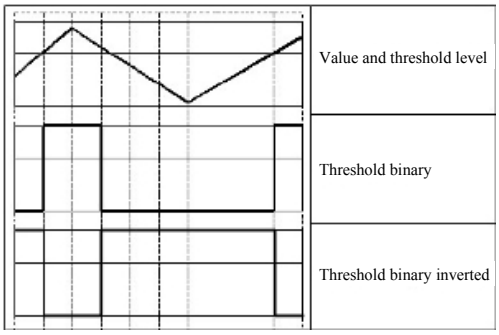

Figure 3. Input and output of the Threshold function

Figure 4. Scrollbar in the black and white window

3.3 Optical Character Recognition

The OCR used in the system is Tesseract (Web link: http://code.google.com/p/tesseract-ocr/ (last access Jan 2011)). Tesseract is open software that was primarily designed for pulling out the text from .pdf files. This design is the reason for the previous image treatment, providing a similar image to those that the OCR is expecting. The OCR is expecting text images in a positive way: letters darker than background.

Tesseract first reads the input file line by line searching for black pixels. This is the edge detection process, in which the OCR scans the input and splits black pixels into blobs that will be processed to extract outlines. This first part of the work made by the OCR is very interesting because it gives us the possibility to deal better with the light and shadows problem in the image capture. Sometimes it will be easier to obtain a white background and black letters from the sign and other times white letters in front of a black background will be clearer. In both cases, when the characters are wide enough, the outlines obtained after the edge detection are similar.

After the edge detection, the lines are adjusted for skew and the gradient or rotation of the text is calculated. Then, a high-level procedure segments the blobs into possible words (considering the spacial disposition of the blobs) and space them. Finally, a classification method converts the analyzed features into letters and makes a quality check, saving the result.

Although Tesseract can read a wide variety of image formats and convert them to text in over 40 languages, a low accuracy could be improved by training it. This is useful when the character type is uncommon or a clear image of the text is not being enough to obtain appropriate results.

Using the OCR could be a consuming process depending on the CPU used by the device. As our system is designed for portable devices, which usually have not very high-powered CPUs, it is not possible to repeat the whole process for all frames (25 per second). Otherwise, a possible user does not need this time resolution, analyzing one frame every second or every 0.5 seconds would be enough: everybody needs a minimum interval of time to change his or her position significantly. The time between text searches can be adjusted by the user by means of a scrollbar situated in the main window (Figure 5).

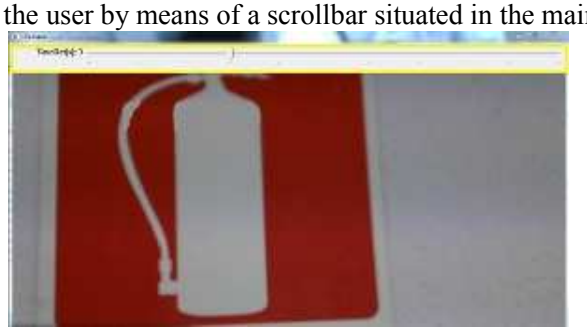

Figure 5. Scrollbar controlling time between text searches

When the OCR finishes, the text output is saved in a .txt file. This file is available for a limited time because it will be overwritten with the new results obtained by Tesseract in the following search.

3.4 Text Localization

When the OCR finishes its work, the system must look into the given file for any coincidence with one of the signs stored in the database. The OCR output (a string) must be compared to the text of the signs (text patterns) stored in the database. This comparison is made by using an algorithm inspired by the Levenshtein distance (Navarro, 2001; Woltzenlogel, 2007)

The Levenshtein distance is a way of measuring the distance between two strings. It is used by the correctors of the text processors, offering words with a maximum distance between them and the wrong written one. The main challenge faced by the proposal presented in this paper is the way of dealing with the noise that the OCR includes when it finds some patterns in the image similar to a character. Same examples can be seen in figures 6 and 7.

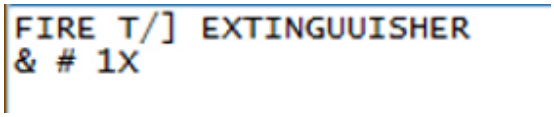

Figure 6. Output file when processing an 'FIRE EXTINGUISHER' sign

Figure 7. Output file when processing a 'EXIT' sign

As is shown, sometimes there are too many mistaken characters introduced by the OCR. That is why in this case the Levenshtein distance is not useful in its original form. To fix this problem, the proposed algorithm compares the text contained in the file handled by the OCR in a different way with every sign saved in the database. When comparing one, it looks for the first character of the sign in the file text. If there is a hit, it will look for the next character of the sign text starting now from the following position which was successful. If not, it will look for the next character from the last starting position. The algorithm saves the sign which has the highest number of hits only in case the number of hits divided by the number of characters contained in the text sign exceeds a defined and tunable percentage.

Figures 8, 9 and 10 illustrate this process, where the input is EMERGENCY EXIT.

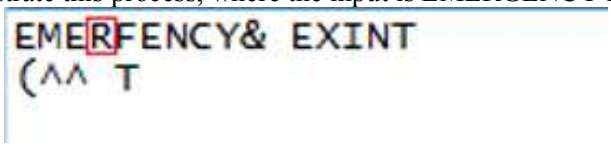

Figure 8. Hit on character 'R'

As shown in Figure 8, there is a hit on the character 'R', so the algorithm is going to look for the character 'G' (the following one in the sign text) starting from the position where character 'F' is situated in the file, as shown in figure 9:

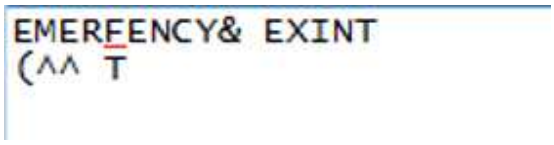

Figure 9. Starting position for the next search during comparison

The algorithm is not going to find any 'G' character among the remaining characters. From this position, the next hit will be with character 'E':

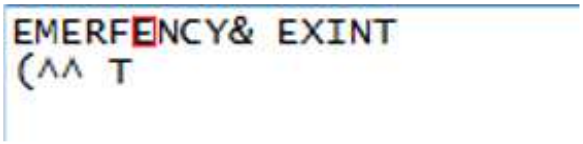

Figure 10. Hit on 'E'

When the algorithm ends, two results are reported:

- 13 hits (spaces are included)
- 0.62 (13 hits over 21 characters in the signs) ratio (62 %).

If the percentage (developing tuning) is enough, and the number of hits is the highest one, this signal will be represented.

Also, in this example, it is possible to see the importance of the number of hits. The sign EMERGENCY EXIT also contains the sign EXIT. If the success ratio were the only controlling variable, as EXIT is contained in EMERGENCY EXIT and also being shorter, it is more likely to have a higher ratio (percentage). In this case, the system would propose EXIT as the sign detected by the system, thus producing an error. Nevertheless, by using the number of hits as a controlling variable, the system may be able to distinguish examples similar to this.

3.5 Representation

Once a sign is detected, it must be presented for the user. Considering possible users with different levels of visual impairment, the system will provide two different representations. First of all and thinking of blind people, the main presentation must be an audible one by converting the text into speech. For this purpose, there are several Text to Speech systems available. The choice of one of them is not so relevant because any of them will read any given text sign. The differences could be found during the inclusion in the system (some of them could be easier to include). In this work, the Loquendo (Web link: http://www.loquendo.com (last access Jan 2011)) system was integrated.

Figure 11. System output during representation

The second representation is a visual improved output. The text detected by the system will be printed out on the screen in a bigger and brighter format, helping the users with visual difficulties. After the representation, the complete process will finish and start again with a new frame (extracted from the video signal) after a certain time (1 or 0.5 seconds depending on the user configuration). Figure 11 shows a cap-

ture of the complete system when representing the EXTINTOR sign information (audio is assumed but not represented in the figure).

4 Conclusions

This paper has presented a low complexity strategy for detecting and recognizing text signs automatically. Instead of using lengthy image algorithms for detecting the text sign, and then, applying an Optical Character Recognition (OCR) algorithm in the areas previously identified, this paper proposes the application of the OCR in a whole lightly treated image and then carries out the text detection process on the OCR output.

This paper has also described the implementation of a real time system. The system is made up of five main modules: the first one captures the image (or sample) from a video camera. Secondly, the color image is converted into a gray scale and then into a black & white image. The B&W image is passed to the OCR that will provide a text string. The text localization process is carried out on this text string considering several text patterns (from the most relevant text signs) that have been stored in a database. Finally, the text pattern detected is converted into speech for blind people and into a bigger and brighter text for the visually impaired.

Acknowledgments

The work leading to these results has received funding from the European Union under grant agreement n° 287678. It has also been supported by TIMPANO (TIN2011-28169-C05-03), ITALIHA (CAM-UPM), INAPRA (MICINN, DPI2010-21247-C02-02), SD-TEAM (MEC, TIN2008-06856-C05-03) and MA2VICMR (Comunidad Autónoma de Madrid, S2009/TIC-1542) projects. Authors also thank Mark Hallett for the English revision of this paper and all the other members of the Speech Technology Group for the continuous and fruitful discussion on these topics.

References

X. Chen, J. Yang, J. Zhang, and A. Waibel, “Automatic detection and recognition of signs from natural scenes,” IEEE Trans. Image Process., vol. 13, no. 1, pp. 87–99, Jan. 2004.

X. Huang and H. Ma, “Automatic detection and localization of natural scene text in video,” in Proc. 2010 20th Int. Conf. Pattern Recognition (ICPR), 2010, pp. 3216–3219.

W. Jirattitichareon and T. H. Chalidabhongse, “Automatic detection and segmentation of text in low quality Thai sign images,” in Proc. IEEE Asia Pacific Conf. Circuits and Systems, 2006, pp. 1000–1003.

K. Jung, J. Han, K. Kim, and S. Park, “Support vector machines for text location in news video images,” in Proc. TENCON, 2000, vol. 2, pp. 176–180.

K. Kim, K. Jung, and J. H. Kim, “Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 25, no. 12, pp. 1631–1638, Dec. 2003.

G. Navarro (2001). "A guided tour to approximate string matching". ACM Computing Surveys 33 (1): 31–88

P. Shivakumara, T. Q. Phan, and C. L. Tan, “A Laplacian approach to multi-oriented text detection in video,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 33, no. 2, pp. 412–419, Feb. 2011.

B. Sin, S. Kim, and B. Cho, “Locating characters in scene images using frequency features,” in Proc. IEEE Int. Conf. Pattern Recognition, 2002, vol. 3, pp. 489–492.

B. Woltzenlogel Paleo. An approximate gazetteer for GATE based on levenshtein distance. Student Section of the European Summer School in Logic, Language and Information (ESSLLI), 2007.

W. Wu, X. Chen, and J. Yang, “Detection of text on road signs from video,” IEEE Trans. Intell. Transp. Syst., vol. 6, no. 4, pp. 378–390, Dec. 2005.

J. Yu and Y.Wang, “Apply SOM to video artificial text area detection,” in Proc. Int. Conf. Internet Computing in Science and Engineering, 2009, pp. 137–141.