

Optical Character Recognition using Dynamic Memory Image Algorithm

G. High Court Rajan¹, S. Pandiaraj², Dr. J. Jagadeesan³

¹M. Tech. Student, Department Of Computer Science & Engineering, SRM University, India

²Assistant Professor, Department Of Computer Science & Engineering, SRM University, India

³Head Of Department, Department Of Computer Science & Engineering, SRM University, India

Abstract : Embedded products have different types of display screens like LCD, Touch screen, CRT etc. For automating the development testing of such devices, we need to recognize the text displayed on it using a camera. It is always a challenge to recognize the LCD display screen using OCR engines. Due to the custom display nature of these LCD controllers as well as the small font size, character recognition is a complex job. Traditional way of recognizing such screens uses template matching techniques. Such image matching techniques take precious time which drastically increases depending upon the number of characters and their size. In this paper, we are going to discuss an algorithm, which helps to recognize such small text in an accurate and speedy manner using low cost resources like webcam and open source software.

Keywords: Dot Matrix LCD, Dynamic Memory Image, Embedded, Image Processing, OCR

I. INTRODUCTION

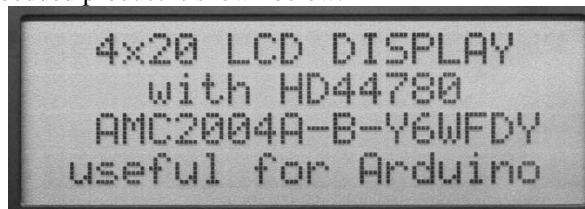
Recognizing text accurately from small LCDs of embedded products using computer vision techniques, within a couple of seconds is a major challenge. Computer Vision based solutions uses image matching techniques and OCR engines to recognize text. Existing OCR engines can recognize text accurately depending upon the type of font used. Many of the print font could be OCR'ed properly. However, when the embedded device has LCD controller based display, OCR engines are not effective. Alternative image processing solutions uses template matching techniques. In those techniques, the image to be searched is swept across the reference image. Such sweeping algorithm requires significant amount of time.

In this paper, we are going to discuss about an image processing algorithm, where we could recognize LCD display accurately in a short period of time. First, we will describe an embedded system with LCD display and the challenges in testing the product during its development. We will study the behavior of OCR engine on such a display. We will then talk about the image matching techniques available today. We specifically focus on sweeping algorithm along with its performance limitations. We will then explore a solution using Dynamic Memory Image Algorithm where we could accurately recognize such characters in the shortest time.

Finally, we will discuss some future enhancements that could be done based on this which could take the performance to the next level.

II. CHALLENGES IN TESTING EMBEDDED PRODUCTS

Embedded products uses variety of display devices like LCD, LED, Touch Screen etc. Example of one such LCD display used in an embedded product is shown below.



Typical LCD Display in an embedded product

Development testing of embedded products with display is typically done manually using a dedicated test team. The testers need to visually verify the display of the screen against the test requirements. Such manual testing takes time and compromises the quality of results.

Many such Graphical User Interface (GUI) tests can be automated using Computer Vision based technologies. Industry grade solutions available in the market comes with a high price tag and hence not useful to many product development companies. Alternate solution is to use low cost webcam as the primary camera device and read the display using image processing techniques and OCR engines.

III. OCR ENGINE BEHAVIOR

Many OCR engines available in the market could recognize the text well with printed text or books. However, such engines fail considerably when tried with embedded products. The following picture taken from a typical LCD display could not be recognized with a powerful OCR engine like “Tesseract” from HP / Google.

LCD Display (Captured with webcam)	Tesseract Output

Some of the OCR quality could be improved by following a complex procedure called “training of the OCR engine”. However, despite the training procedure, the quality of the text recognition could not be guaranteed for accuracy. Hence, the text recognized could not be fully relied upon for testing purposes.

IV. EXISTING IMAGE MATCHING TECHNIQUES

Computer Vision application relies on the template matching methods to recognize images. If we were to search for a given image from a database of images, we need to apply template matching algorithm and get matching scores. The matching algorithms provide the matching score for every location in the image. The location with maximum score is most likely to have a better match.

Consider the example shown here, where we need to search for a given face from a picture.

Existing algorithms provides a high matching score (> 80%) for the correct face and relatively less score (< 60%) for the other face. Every other place would have a very low matching score (< 30%). In this way, we could find a matching image from a given one.

V. SWEEPING ALGORITHM FOR IMAGE MATCHING

The image searching algorithm works in a method called “sweeping”. As shown in the above picture, the image to be searched is swept across the target image and matching score on every location is calculated. If the original image has a width of ‘m’ pixels and a height of ‘n’ pixels, then a matching score needs to be calculated for each of the (m, n) locations.

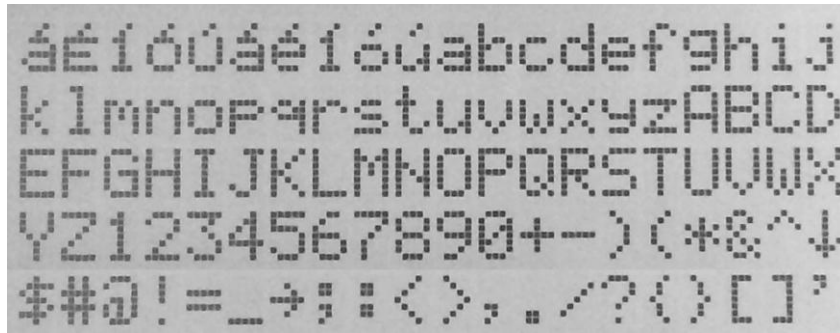
The algorithm works in two dimensions of width and height. This is needed to ensure that the entire image is searched and every possible location is included in the search. Though this improve the quality, the performance takes a hit as explained in the next section.



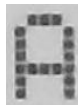
VI. PERFORMANCE LIMITATIONS OF SWEEPING ALGORITHM

Let us focus on the performance limitations of the sweeping algorithm in recognizing the LCD display. Consider an example that the LCD display has a character set of 100 characters. Each character is represented in a box of 62x92 pixels. Then the template image will have a width of 1240 pixels and a height of 460 pixels as shown in the following picture.

Template Image (1240x460 pixels)



Character to be recognized (62x92 pixels)



The template image would be captured on the same environment with custom screen displaying all the characters. This needs to be done once per controller to get the original display image. The total number of pixels in the template image would be 1240x460, which is 5,70,400. In other words, the sweeping algorithm to search for a single character like ‘A’, need to match images at a minimum of 4,33,504 locations.

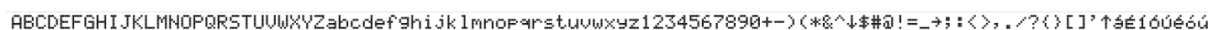
For an embedded product which has a LCD display with 20 columns and 4 rows (for a total of 80 characters), such an algorithm requires more than 8 seconds. (*Measured on a 2nd Generation Intel i5 processor running at 3 GHz with 4GB DDR3 RAM.*)

In other words, we would need a minimum of 8 seconds to recognize the 80 characters in the display. This time increases significantly when the number of characters in the character set increases. Typically, the controllers can display around 255 characters (including multiple languages). In such a scenario, the minimum time for the screen reading would be more than 20 seconds.

Such a long time to recognize the screen could not help in automating the testing of embedded products. The screen changes could happen more quickly than this.

VII. DYNAMIC MEMORY IMAGE ALGORITHM

One of the key challenges for the sweeping algorithm is the need to sweep in two dimensions of width and height. We could dramatically improve the performance by sweeping in one dimension like along the width only. For this, we need a template image as shown below



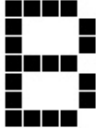
(Note: Size reduced to fit the page)

In such a scenario, for the same number of 100 characters, the total width would be 6200 pixels. The sweeping algorithm needs to match 62x92 image at just 6199 locations only. This could dramatically improve the performance by 70 times (*Measured on the same environment.*)

The challenge is generating such a template image and maintaining it for maximum number of characters. It is highly challenging to edit such an image using graphics editors. For this problem, we propose our Dynamic Memory Image algorithm.

Dynamic Memory Image algorithm creates the template image in memory at runtime using font data for each character. Each character could be represented as a bitmapped data. In this example, each character could be represented as 5x7 matrixes. Each column is represented by a byte. Every character could be represented with 5 bytes of data. The following table shows example data for two characters A, B.

Character	Font Data	Dynamic Image Generated at Runtime
A	{0x7E, 0x11, 0x11, 0x11, 0x7E}	

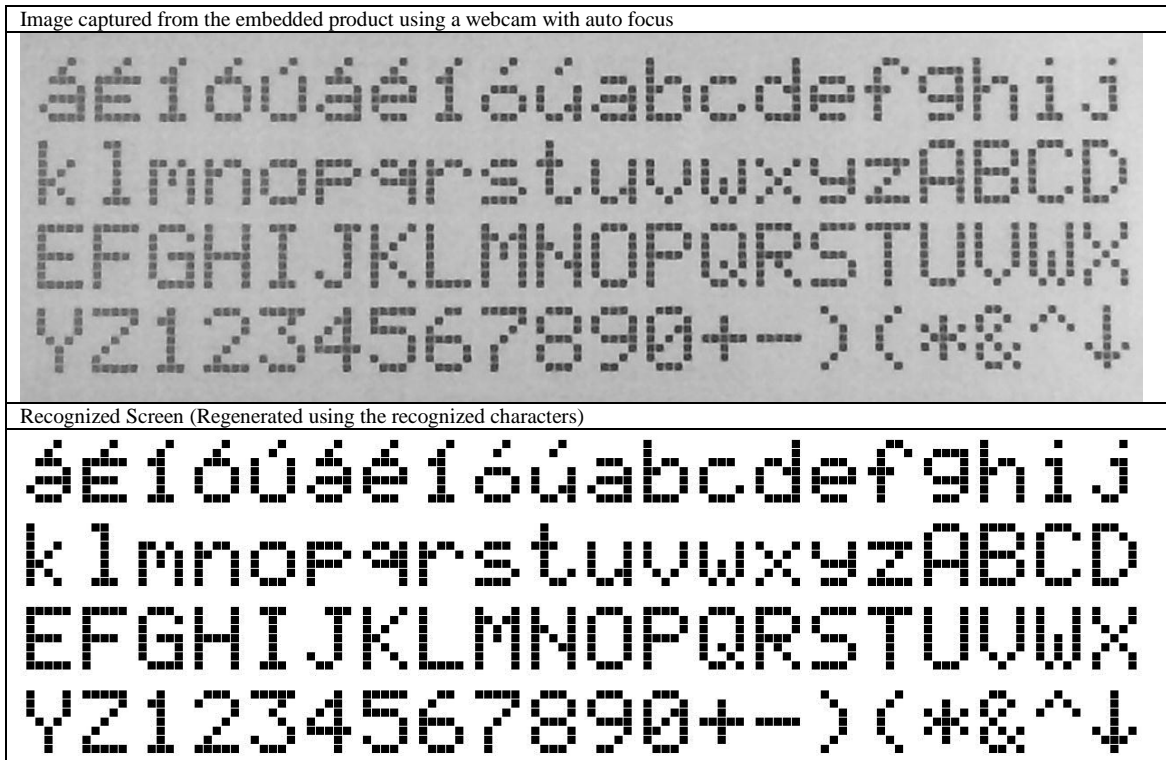
B	{0x7F, 0x49, 0x49, 0x49, 0x36}	
...		

Once we have the font data for all characters supported by the LCD controller, then the entire template image could be generated at runtime in a single dimension. The same sweeping algorithm could then be used with this template image to operate in one dimension (along the width alone) for maximum performance.

The font data can be computed manually, only once per character. In the above example, for character A, the 1st column has a bit map of ‘111110’, which can be coded as 0x7E. 2nd Column has a bit map of ‘0010001’, which can be coded as 0x11. The font data can also be auto generated using dedicated applications like bitmap font editors.

VIII. PERFORMANCE BENEFITS

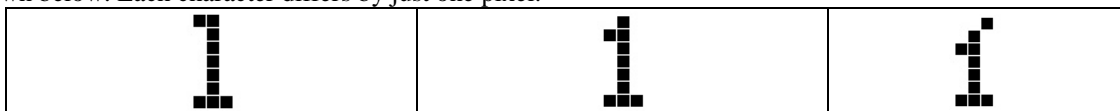
On the same environment, the time to recognize 80 characters was less than 2 seconds with 100% accuracy. Such accurate screen reading in a short period of time opens a lot of possibilities to automate the testing of embedded products. All screen navigations could be checked automatically and any bug in implementation could be detected automatically. This reduces the time needed to perform a single round of testing as well as quality of test results.



IX. FUTURE ENHANCEMENTS

The dynamic memory image algorithm can be further enhanced by matching one character at a time and stopping the algorithm once a specific score has been reached. In other words, we could attempt to match each character from the template image with the character to be recognized.

We need to figure a way to distinguish similar characters which would have a close matching score as shown below. Each character differs by just one pixel.



Another way to improve the performance further could be to reverse generate the bitmap data from the screen capture and avoid the sweeping algorithm completely.

In other words, attempt to get pixel data (On / Off) for each character and generate the bitmap data. Once we have the bitmap data of the screen display, then we could compare the bytes using memory compare functions and determine the exact character being displayed. This could result in the best possible performance. There are high chances that the 80 characters in a given screen could be recognized within few milliseconds. However, we need to be watchful on the accuracy as it may be challenging to get the pixel data due to any camera alignment issues.

X. CONCLUSION

As we can see, the dynamic memory image algorithm addresses the performance and quality issues in recognizing the text of the LCD displays. The solution can be implemented using low cost and affordable resources like an ordinary webcam, standard PC along with open source libraries like OpenCV. Ability to recognize the LCD display accurately and quickly, helps the automated testing of embedded product's GUI, improving its quality and time to market. There are scope to improve the solution to the next level also.

REFERENCES

Journal Papers:

- [1] Tekin, E. Smith-Kettlewell Eye Res. Inst., San Francisco, CA, USA; Coughlan, J.M. ; Huiying Shen, Real-time detection and reading of LED/LCD displays for visually impaired persons, Applications of Computer Vision (WACV), January 2011 IEEE Workshop.

Books:

- [2] Daniel, David, Mahmood, Roy, Emami, Jason, Mastering OpenCV with Practical Computer Vision Projects (PACKT Publishing, December 2012)
- [3] Gary Bradski & Adrian Kaebler, Learning OpenCV - Computer Vision with the OpenCV Library (O'REILLY 2008)

Chapters in Books:

- [4] Chapter 5: Number Plate Recognition Using SVM and Neural Networks
- [5] Chapter 7: Histograms and Matching