Australian Currency Note Identifier for the Vision Impaired : Part II – Software Description

I Siewert, I Murray, T Dias
School of Electrical and Computer Engineering
Curtin University of Technology
Kent Street, Bentley, Western Australia 6102
Australia
i.siewert@ece.curtin.edu.au | i.murray@ece.curtin.edu.au | rdiast@curtin.edu.au

This second of a pair of articles describes the software component of a prototype currency identifier. Identification is performed by imaging and recognising the contents of the clear window found near the lower corner of each note [1,2], which is unique for each denomination. This development is of significance in Australia to people who suffer a large degree of vision impairment, and possibly also to the vision impaired population of the European Union, which may also plan to adopt polymer note technology.

1 Introduction

In this, the second of a two-part article on a currency identifier for the vision impaired, an explanation of the software developed for the prototype device outlined in Part I is described [2].

2 Background and Objectives

2.1 Image Recognition

The recognition of the clear window (Figure 1) and the pattern contained within was chosen as the method of distinguishing between the various notes due to its resistantance to wear and ease of illuminating and imaging. Ideally, the result after imaging and digitising will be a high-contrast image containing just two levels. Each pixel will either be 'on' where it has received light through the clear window, or 'off' if it has not. This two-level binary digital image contains all the information necessary to identify the window unambiguously, and hence the note [1].

The continuous curve of the window's extremity is considered as a single entity or character. Likewise, the patterns within can be considered as separate characters. These can be identified using optical character recognition (OCR) techniques. Four basic steps were taken when processing and recognising the complete image. These are:

1) Data capture
2) Pixel-level processing
3) Line-level analysis
4) Feature detection and recognition.

2.1.1 Data capture

The imaging of the note is carried out using a low cost CCD linear array. The note is swept across the array whilst being backlit by a light source. A full description of the hardware is contained in [2].

2.1.2 Pixel-level processing

Even though the image of the note's window and its contents are inherently binary, it is advantageous to first capture a grey-scale image, and then perform binarisation. The threshold between 'on' and 'off' pixels can be chosen to maximise certain criteria such as connectivity or separation between objects. A single threshold can be found and applied globally, or an adaptive algorithm can be used to find local thresholds, especially when the difference in levels between background and foreground vary greatly. Adaptive algorithms commonly use local windows to analyse the grey-level data to determine the local threshold. If the pixel values across the background and foreground are fairly consistent, global thresholding is preferable. Global methods use overall characteristics of the image such as moment statistics of the intensity histogram, which chooses a threshold that best preserves the first four moments as compared with the original grey-scale image.

2.1.3 Noise reduction

A common type of noise that appears as small holes and speckles in the image is known as 'salt and pepper' noise [3]. Mathematical morphology is commonly used in removing such noise. This is

an iterative process where a number of chosen small elements are added to the edges of image regions, followed by the deletion of the same element. This process of dilation and erosion has the effect of removing the small holes and thin indentations from the image. If erosion precedes dilation, the effect is to remove speckle and thin protrusions [5].

2.1.4 Line-level processing

Thinning is a line-level process where regions are successively eroded until a single pixel-width line remains. This line along the approximate centre of the region is the skeleton. A practical algorithm to find the skeleton must first identify end points so protrusions can be preserved in the final skeleton representation. The purpose of deriving a skeleton is to decompose thick lines and regions to simple structures that can be easily analysed in terms of the number of components, the connections between them, curvature of lines and other salient features.

The chain code of a thinned line is stored more efficiently than its pixel representation. More importantly, the chain code contains information concerning connectedness. The simplest chain code traverses a line in a particular direction and labels each pixel with one of eight directions to indicate where the next pixel is located. A disadvantage of this particular chain code is that topology information is not directly available.

2.1.5 Feature detection and recognition

The fitting of curves to thinned lines provides a more succinct description of the thinned lines. The chain code representation of the thinned line readily lends itself to this extra processing. A fitted line can be analysed in terms of curvature. Gently sloping portions will have low curvature while the curvature of sharp points will be high. An overall description of a contour can be determined by calculation of its Fourier transform along its entire length. The resulting Fourier coefficients yield a frequency domain description. Higher order terms give information on sharp features while lower order terms describe lower curvature portions of the contour.

2.2 Cross-correlation technique

The image recognition techniques already outlined require a precise CCD image of the entire window, which is then decomposed to extract features. This would be very difficult to achieve in practice. Since the passage of the note across the array is of arbitrary speed and direction and the scanning speed of the imaging device is limited, a linear or continuous image cannot be obtained. An area CCD and lens, as used in video cameras, could be used to capture a complete image. However, the cost is prohibitive. A new recognition method was then sought, which did not require a complete window image to be obtained and which offered improved reliability of recognition despite image degradation from a number of factors. A possible solution presented itself when it was realised that only small portions of each note's window needed to be captured, and then matched to precisely defined images held in memory. The processes of data capture, thresholding and noise reduction still need to be carried out, but feature extraction is no longer necessary.

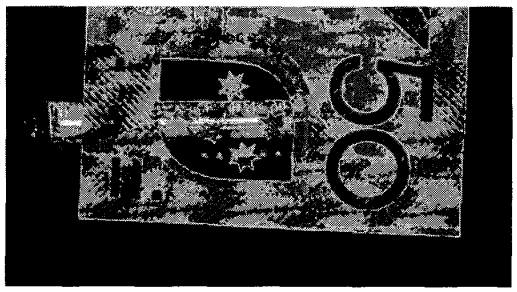

Figure 1: Swipe Action of Note over the CCD

In Figure 1, a fifty dollar note is shown over a device which images a single line through the window. The image obtained across the window could be represented by the sequence shown in Figure 2, where light is shown as '+' and dark is shown as '-'. The two stars within the window cause the two dark segments.

++++--------++++++++++++++++++++++++++++++--------++++++

Figure 2: Representation of Image Slice through Window of Fifty Dollar Note

This particular sequence is unique across all the window designs. Therefore the note can be identified as being a fifty dollar note. Since most image slices obtainable from the fifty dollar note through the stars in various directions are unique, the image slice matching could identify in almost any swipe direction. This can be extended to all the other notes, where the difference in internal detail, and thus the image slices, is sufficient to unequivocally identify them. It now remains to find a method of matching the image slices obtained from the note to those stored in memory.

The matching method relies on cross-correlation, in which two number sequences are compared. In

Figure 2, the light and dark segments are written with '+' and '-' signs. These are shorthand representations for the numbers '1' and '-1' respectively. One sequence represents a slice of window image obtained from the CCD, the other is a slice of window image stored in memory. The process, similar to convolution, can be visualised as one image 'sliding past' the other, with the degree of similarity being recorded at increments. If the sequences are similar, a large correlation will occur at the point when the two sequences are 'lined up' alongside one another to give the best match. This process is shown in Figure 3. The image slice from Figure 2 is shown once more at the top, and below is a stored image slice from a fifty dollar note, referred to hereafter as the 'template'. It is desired to find the correlation, and from this the number of elements which do not match. The process begins with the template lined up at the beginning of the image slice.

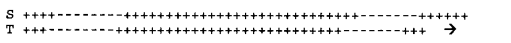

Figure 3: Correlation Example

Multiplying the two sequences, element by element, and summing the result calculates the correlation in this position. This is expressed mathematically as:

$$Correlation = r_{ST} = \sum_{m=1,n=1}^{length} S_m T_n \qquad (1)$$

where 'S' and 'T' are the image slice and template sequences, 'm' and 'n' are the element numbers of the image slice and template respectively, and 'length' is the number of elements in the template. For a template position other than that shown in 3, 'm' must be altered accordingly. Because light and dark elements have been assigned the values '1' and '-1', Equation 1 is more simply stated as:

'The number of matching elements - the number of mismatching elements'

The correlation for the position shown in Figure 3 is 42 - 6 = 36. If, after the correlation, only the template length and the correlation result is known, then the number of mismatched elements can be obtained from:

Number of mismatches = (Template length - Correlation value) / 2

For the example, the number of mismatches is: (48 - 36) / 2 = 6. This calculation yields the correct result, even if the correlation value is negative. For the example template length of 48, the correlation value has a possible range of -48 for no matching elements at all, to +48 for an exact match.

Since it is desired to find the best match within the longer sequence, the template must be incremented and the correlation process repeated. The highest value at any increment then becomes the correlation.

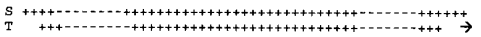

Figure 4: Position of Best Match.

The best match for the example sequences occurs at the position shown in Figure 4. In this case there are only two mismatches. This also occurred at the previous increment. To be sure that the highest correlation has been found, shifting and correlating must continue until the template reaches the end of the slice. If a correlation above a certain threshold is found for a certain stored window slice, while the correlation is low for others, the conclusion can be drawn that the window, and therefore the note has been identified, with a degree of uncertainty.

3 Software Design

This section presents the major software routines developed in detail [1].

3.1 Software overview

The software is designed to correctly control the scanning of the CCD linear array and to process and attempt to match the pixel information it produces. If a match can be found, then the voice chip plays the correct voice message. A complete list of interrupt service routines and subroutines together with their main functions are shown in Table 1.

Table 1: Subroutines and Interrupt Service Routines

Name of Subroutine or ISR	Function
CLKL_ISR	Store pixels or produce SI signal, determine and store transitions.
SO_SIG_ISR	Flash LED, determine if scan valid, set SI flag for next scan.
INITIALISE	Set all applicable TMSC50 registers, set start-up message.
START_CCD	Reload and start timer.
STOP_CCD	Stop timer.

CHECK_VALID	Determine if scan just ended is valid.
SEARCH	Locate a similar template length.
RECONSTRUCT	Write template into buffer from the stored segment run-lengths.
CORRELATE	Find the number of mismatched pixels between the scan and the found template
REVERSE	Write template into buffer backwards.
ANNOUNCE	Play voice message.
NEXT_SLICE	Set flags to obtain another scan.

In order to correctly scan the array, Serial Input (SI) signals must be generated to initiate each new scan, timed for correct positioning with respect to the clock signal. This is done within the Clock Low ISR (CLKL_ISR). At the end of the scan, the array produces a Serial Out (SO) signal. The LED is flashed and the scan is then examined within the ISR to see what further processing (if any) should occur on the collected pixels. Further processing is warranted if internal features within a window were detected, and a template can be found which has a similar length. The presence of internal features is determined within the CHECK_VALID subroutine, by examining the number of light to dark transitions that occurred. The length of the window is also checked to see if it falls within the valid range of stored template lengths. If the scan was valid, then a binary search for a similar template length is performed by the SEARCH subroutine. Regardless of the outcome of these two subroutines, an SI signal is again generated to scan the array. However, incoming pixels will be ignored if the last slice is valid and is being correlated to the found template. Continuous background scanning of the array prevents the build-up of excess charge in the pixel wells, due to the extended integration time.

Before the window can be correlated against the found template, the template must be reconstructed from its stored segment lengths. This is done using the RECONSTRUCT subroutine. Correlation occurs using the CORRELATE subroutine. Within this subroutine, the reconstructed template is 'stepped along' the whole length of the array slice in order to find the best match. If a sufficient match cannot be found, then the template is written into the template buffer backwards, using the REVERSE subroutine. If a sufficient match can be found in either direction, then an announcement is made via the ANNOUNCE subroutine. After the correlation has been completed, another scan is grabbed and the process repeats. This is accomplished by calling the NEXT_SLICE subroutine [1].

3.2 Main program

The main program performs three tasks before entering an endless loop, designated MAIN_LOOP. First, the TMS320C50 registers are initialised with the INITIALISE subroutine. Next, the START_CCD subroutine starts scanning the linear array. This is followed by the playing of a 'ready' message using the ANNOUNCE subroutine.

A determination is made between each scan as to whether it is valid. A scan is valid if at least two light to dark transitions occurred, and the length of the collected window is within range of the stored templates. If valid, a search is performed through the table of template lengths to see if any have comparable lengths. If a template can be found, the pointer to the template in the template storage area is loaded. The run-lengths of light and dark pixels are then written into the template buffer with the RECONSTRUCT subroutine.

The collected scan in the buffer can now be correlated using the CORRELATE subroutine to determine its closeness of match with the reconstructed template. If the correlation is not within the set BOUND, then the template is reconstructed again back-to-front using the REVERSE subroutine, and correlated again. If the correlation is sufficient, the denomination corresponding to the stored template is announced to the user from the ANNOUNCE subroutine. After setting the flags in the NEXT_SLICE subroutine to obtain the next available scan, a branch is then made back to the beginning of the MAIN_LOOP and the entire process repeats [1].

3.3 Correlate subroutine

This important subroutine forms the heart of the recognition process by correlating the template located by the search subroutine and correlating it against the image slice from the linear array. On entering, the VS flag is first checked to see if the scan is valid. This will not be so if the window

length could not be found in the table. The VS flag is cleared at this point, as no more tests need to be performed on it. If the scan is not valid, then nothing can be done, and a conditional return is executed.

If the scan was valid, then the incoming scan stored in the buffer needs to be correlated against the found template. To do this, the buffer needs to be transferred into program memory from data memory. This is neatly accomplished by setting the CNF bit (bit 12) of status register 1 (ST1). The starting address of the program memory buffer is now loaded into the block move address register (BMAR) register, as required by the MADS instruction. The minimum possible value of the correlation is -512, if all pixels are mismatched. The value stored in CORRELATION is initialised to this maximum mismatch value for later comparison. AR0 and the circular buffer start register (CBSR1) are now initialised to the start of the template buffer. The circular buffer end register (CBER1) is loaded with the template buffer location of the final element of the current reconstructed template. Adding the template length to the template start address already in the accumulator and subtracting one calculates this address. The correlation process dictates that the template be 'slid' in increments along the window of the current scan in the buffer. The number of increments is the difference between the length of the window and the length of the template. The length of the window is loaded into the accumulator and the length of the found template is subtracted from it, since this is always shorter (or equal). At each increment, the length of the template is correlated against a section of the window in the buffer.

The repeat block correlates the template over its entire length. After the accumulation of the last product, the accumulator holds the correlation value. This is swapped into the accumulator buffer (ACCB)[6,7] and the previous stored correlation value (-512 for the first block repeat) is loaded into the accumulator. This is then stored back into the CORRELATION variable. After all block repeats have been performed, the highest value of correlation found at any increment remains. On the first iteration of the block, the first element of the template is correlated against the first element of the buffer, the second against the second, and so forth. On subsequent iterations (if any), the template is 'stepped along' one element at each iteration, so that on the second iteration, the first element of the template is being correlated against the second element of the buffer, and so on. At this point, if the value in the block repeat counter register (BRCR) is not zero, then control is passed immediately back to the start of the block for the next repeat of the block.

Once all block repeats are finished, CORRELATION holds the greatest correlation value found at any increment of the template 'along' the buffer. This could be a negative or positive number. To find the number of mismatched pixels, the correlation value is subtracted from the length of the template, and then divided by two. The final result is the number of mismatched pixels, and this is stored in MISMATCH.

4 Results

Running the software and observing any problems tested the completed system. The LED position was adjusted for the most even spread. As already outlined, the evenness of spread is less than ideal. This was therefore a difficult task. Allied with this is the required flash time to fully illuminate every pixel above the comparator threshold. A problem was then discovered, as it appeared that two pixels were less sensitive than all the others. Since these were towards the middle of the array and usually under the note, this was ignored. However, since transitions are noted by comparing adjacent pixels, this has the potential to cause failure of the recognition process in the prototype. This is because any single stray pixel produces a light to dark and a dark to light transition. If these occur before or after the window, then the beginning and/or end of the window will be erroneous. This highlights the importance of low-pass filtering the image. Slices were then read in from the windows of several notes and stored. This was done by placing the desired slice of the particular note across the array, stopping execution of the software, and reading the transitions from TRANS_ARRAY. This gives the pixel position of each transition from the start of the window. The run-lengths of each segment were calculated from these and stored into the template storage area, with the correct message to play appended. The overall lengths of each template together with their associated pointers were stored into the template table in ascending order so that the binary search algorithm could locate them. When the software was then run, the prototype correctly identified all the templates as each note was slowly passed over the array.

As the prototype has a very limited set of templates and processes every valid slice, it is possible that a slice able to be matched passes unnoticed while another valid, but unstored slice is being processed. Therefore, swiping too fast causes recognition to fail. Storing strategic slices can alleviate this problem. However, the best solution is not to attempt to process scans as they come in, but to store them for later processing once the note has passed.

5 Conclusion

5.1 Achievements

The prototype recognises five, ten, fifty and one hundred dollar notes when they are swept horizontally over the array [1,2]. This shows the usefulness of the correlation technique as a method for recognising the clear window of Australian notes. A useful device for the vision impaired could stem from the prototype if several problems can be overcome.

5.2 Future Improvements

5.2.1 Slice storage

As little time exists to process the scans as they arrive, all incoming slices should be stored for later processing. Scans can be compactly stored in terms of their transitions. If a number of scans are stored, then each can be correlated and a decision made by majority rule. This would greatly improve the accuracy of the device.

5.2.2 Fast Fourier Transform

Correlation in the prototype is performed using the direct method. However, using Fast Fourier Transforms (FFTs) will provide a speed increase when the note's window is long [4], and particularly when the window extent cannot be identified. This will happen when the note has been damaged around the window. In this case, it may be necessary to try a large number of templates against a long scan, in the hope of extracting a match from the surrounding noise. This represents the worst situation that can occur. However, it should be said that the vast majority of notes are in good condition. This is because they are withdrawn very quickly from circulation when wear is evident.

5.2.3 Image filtering

The prototype can be misled by isolated pixels. This has the potential to cause failure if the window extremities cannot be identified. Isolated pixels also occur inside the window due to dirt and creases. While not being a major problem, low-pass filtering of the scan would remove these as well. The simplest method that could be applied is to examine a small block of contiguous pixels to see if a transition has truly occurred over its length, or is merely a sporadic event. This can be done by digitally filtering the input with a FIR filter.

5.2.4 Note authentication

Finally, in order to add to the usefulness of the device, some form of authentication of the note could be performed. This would need to be low in cost. The polymer itself may possibly provide the easiest means to accomplish this. If the polymer material has certain optical peculiarities, then this could be exploited to give some peace of mind as to the note's authenticity as well as performing the identification.

6 References

[1] Siewert, I., "*Australian Currency Note Identifier for the Vision Impaired*", Final Year Thesis, School of Electrical and Computer Engineering, Curtin University of Technology, 1998

[2] Siewert, I., Murray, I., Dias, T, "*Australian Currency Note Identifier for the Vision Impaired- Part I: Hardware Description*", Submitted to ANZIIS 2001

[3] Levaldi, S., *Digital Image Analysis*, Pitman, 1984

[4] Smith, W.S., Smith, J.M., *Handbook of Real-time Fast Fourier Transforms*, IEEE, 1995

[5] Kong, T.Y., Rosenfeld, A., *Topological Algorithms for Digital Image Processing*, Elsevier, 1996

[6] Texas Instruments, *TMS320C5x DSP Starter Kit User's Guide*, 1996

[7] Texas Instruments, *TMS320C5x User's Guide*, 1997