International Journal of Engineering & Technology, 7 (4.39) (2018) 199-205

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Fake Currency Detection Using Image Processing

Ankur Saxena[1] ,Pawan Kumar Singh[2], Ganesh Prasad Pal[3], Ravi Kumar Tewari[4]

[1]G L Bajaj Group of Institution, Mathura, U.P, India
[2 3 4] G.L. Bajaj Institute of Technology & Management, Greater Noida, UP., India
**Corresponding author E-mail: ankursaxena19june@gmail.com*

Abstract

Since last few years, as a result of the great technological advances in color printing, duplicating and scanning, counterfeiting problems have become more and more serious. In the past, only the printing house has the ability to make counterfeit note, but today it is possible for any person to print counterfeit bank notes simply by using a computer and a laser printer at house. Therefore the issue of efficiently verifying counterfeit banknotes from real ones via automatic machines has become more and more important. Counterfeit notes are a problem of almost every country but India has been hit really hard and has become a very acute problem. There is a need to design a system that will helpful for recognition of paper currency notes with fast speed and in less time. This proposed system describes an approach for verification of Indian banknotes. The currency will be checked out by using image processing techniques. The approach consists of a number of elements including processing of image, detection of edge, image segmentation, drawing out characteristic, comparing both images. The image processing approach is discussed with MATLAB to verify the parameters of note. Image processing involves changing the nature of an image in order to improve its visual information for human interpretation. The image processing software is a collection of functions that extends the capability of the MATLAB numeric computing environment. The result will be whether note is real or fake.

Keywords: *Digital Image Processing; Dilation; Grayscale; Image classification; linear correlation.*

1. Introduction

In the 21st century, digital technologies, such as complex scanners, copiers, and computer software, have vastly increased one's ability to counterfeit just about anything. For instance, even students can manufacture bank notes realistic enough to fraudulently purchase school lunch. Money, stamps, checks, coupons, IDs, signatures, and product labels have become the subject of ever-increasingly sophisticated counterfeiting techniques. In order to diminish such an illegal phenomenon, we propose to develop a system that can be integrated into scanners, copiers, and other electronic equipment frequented by counterfeiters. With current technology, fake money is within reach of even average citizens; with the "Can't Copy This", authorities could begin to curb counterfeiting among the masses. We plan to combat this problem through the application of image processing and pattern recognition. The Central Bank Counterfeit Deterrence Group (CBCDG) has already organized hardware and software manufacturers to fight this problem through the development of the Counterfeit Deterrence System (CDS) [1]. Despite this, people are able to outwit these safeguards and utilize commercial products to counterfeit currency. Therefore, more robust solutions are required. Some pre-existing efforts, in coordination with the CBCDG, have found their way into the mainstream. Current versions of Adobe Photoshop, Jasc Paint Shop Pro, and similar software have integrated simple schemes to thwart counterfeiters. Printers in Japan are required to have embedded features to place microscopic dots on all their printouts that indicate their sources through identifying serial numbers. While the details of these schemes are unknown, they have proven to be less than perfect since counterfeiting still presents to be an issue nowadays [2]. A different approach pursued by Media Sec Technologies involves embedding special (invisible) images on product labels and CDs to enable a scanning device to differentiate real products from fake look-alikes [3]. However, this approach will only stymie counterfeiters for so long since it does not actually address the heart of the problem. In order to stop counterfeiting for good, we have to prevent the duplication of anything that resembles treasury notes. Unfortunately, resemblance is a difficult concept for a computer to grasp without aid of software that is capable of carrying out detection and classification of prohibited objects that are not meant to be photocopied or scanned. This fundamental need is what sparks our group to come up with the idea of "Can't Copy This" as our term project. Using advanced signal processing techniques, we have come up with a scheme that detects targets of probable counterfeiting (treasury notes in our case) by conducting pattern recognition. The first step for our project is to construct a reference database that contains images of front and back of the various denominations (2, 10, 20, 100, 500, 1000). With the reference database, we are able to classify prohibited objects using correlation and ultimately prevent them from being copied or scanned. The nature of our technique differs from what has been done in the industry because our goal is to achieve prevention (more effective and thorough in our opinion) rather than detection and recovery.

2. Previous Work

Looking through the previous projects done in 18-551, we did not find any project that addresses the issue of counterfeiting. Nevertheless there have been a number of projects that deal with detection and classification. "Where's the Ball?" from Spring 2004 and "Face Detection for Surveillance" from Spring 2002 are some of the projects that fall into the category above [5, 6]. Some of the techniques that we considered are previously investigated by some the other groups; for instance blob labeling, linear correlation, KNN (peak to side-lobe ration), geometric moments, and morphological processing. However in terms of linear correlation, our project is slightly different because we correlate through the entire image while the other groups run correlation on small areas containing dominant identifying features, such as people's faces. As for details of work done previous in the industry, Adobe Photoshop CS digital editing package includes a counterfeit deterrence system designed to prevent users from accessing images of currency. When the deterrence system detects an attempt to access a currency image, it automatically aborts the operation, displaying a warning message and directs the user to a website with information on international counterfeiting laws. This anti-counterfeiting system actually exceeds the requirement of U.S law which allows color reproductions of U.S. bank notes so long as the reproductions are smaller than 75 percent or larger than 150 percents of actual size [2]. Some other companies with similar products actually voluntarily embed features of anti-counterfeiting similar to that of Adobe to support the counterfeiting laws because cases of counterfeit crime have soared tremendously in the recent years. Despite the similarity in concepts, the purpose of our project is actually very different from those of what exist in the industry nowadays. Our program prevents people from acquiring images of prohibited objects while most of the programs out there either prevent people from modifying an image or leave traces of where the printouts are made. To further emphasize the difference, our project aims to stop people from scanning or copying images of treasury notes, thereby fundamentally eliminating counterfeiting by targeting the root of this crime.

3. System Level Design

Our project consists of a HP PSC 1610 scanner, a PC and an EVM. The scanner will be used to scan in an image, which will then be transferred to the PC side by means of Windows Image Acquisitions (WIA). The PC receives the image and performs scaling, recoloring, image morphing and blob labelling. Then the PC sends the processed image to the EVM. On the EVM side, we first determine the rotational angle of the object using geometric moments. After acquiring the rotational angle, we then un-rotate the object using bilinear interpolation. Edge enhancement using Sobel technique is performed after the process of un-rotation to improve the KNN of our correlation process. Finally we perform linear correlation on the test image against our reference database and send the classification result back to the PC, where the results will be displayed by our GUI, called ScanView. To further illustrate our system level design, we will discuss briefly the tasks that are done on both the PC and the EVM sides: The PC communicates with the scanner using the Windows Acquisition Application (Handled by our ScanView software) to retrieve the image. The PC begins by resizing the given image to 100 dpi resolution. We found that the 100 dpi resolution produces the best results for our detection system. Our system does not rely on color recognition, so we must convert the image to grayscale. We create a binarized representation of the image to aide in our image extraction protocol. The PC then performs morphological processing on the binarized image, which is intended to close the tiny spaces between the numerals in the bottom right of the $5, $10 and $20 Note in order to avoid floating numbers. The PC then performs a blob labeling process that extracts individual objects from the input image and sends each extracted object to the EVM for image classification. Simultaneously, the PC transmits the reference database to the EVM for future correlation processing.

The first task the EVM must do with the received extracted images is to un-rotate them. We utilize a trivial mathematical computation that relies on moment extraction to determine the proper angle that the extracted object must be rotated to help in classification later. With the un-rotated image present, the EVM then performs a Sobel computation on both the extracted images and the list of reference images in our database to help determine the surrounding edge of each image. These edge enhanced images (extracted objects and reference images) are correlated together, which will be crucial in our peaksidelobe ratio functionality. The KNNs for these images are utilized as our classifier mechanism to determine if any of the extracted objects are illegal to copy. The legality of the objects is passed back to the GUI where we specify which objects cannot be copied (as well as a classic musical interlude).

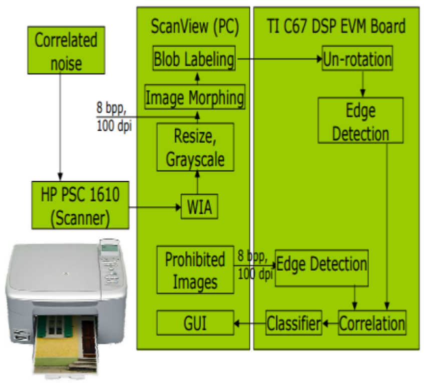

Figure 1: System-Level Block Diagram

This handshaking process between the PC and EVM takes some time. Ideally we preferred that our entire system would take no more than 10 seconds to receive an image, process that image, and return valid or invalid objects to be copied. However, we realized that our system takes just over 1 minute to perform our processes. This is due to many underlying transfers in memory. The transfer time from the scanner to the PC, via a USB port, takes approximately 3 to 5 seconds. We bounded our system to have a maximum extracted image of 500x500 pixels, at 8 bits per pixel (in grayscale); this translates to 250,000 bytes per extracted image. The estimated transfer time for the extracted image from the PC to the EVM is less than 0.1 seconds, using a Host Port Interface (HPI); which translates to 3 Megabytes per second. The reference database that the extracted image must be tested against holds 8 images (the front and back of each un-rotated bill), each approximately 1.5"x5"; which translates to 75,000 bytes per reference image (600,000 bytes for all 8 images). The PC must transfer the reference images as well to the EVM, which takes approximately 0.19 seconds (for all 8 images) using a HPI; which translates to 3 Megabytes per second. Each of the images, however, is transferred one at a time and is fairly time intensive. The transfer of all these images requires quite a bit of the EVM's memory. The external memory that is utilized is approximately 3.5 Megabytes and the internal memory that is utilized is approximately 64 Kilobytes. The off-chip SDRAM is used to store the extracted image (250,000 bytes), reference image (250,000), un-rotated image (1 Megabyte), and a large temporary buffer for Sobel and correlation outputs (2 Megabytes). The on-chip SDRAM is used for our ping-pong buffers, program memory, stack, etc.

4. Algorithms Used

Image Scaling The first algorithm that is performed in our program is image scaling, which simply takes the input image and resize to 100 dpi (using the scaling factor s) for obtaining a more desirable KNN for classification. This is achieved using nearest neighbor interpolation. For each destination pixel in the destination image, we use a scaling transformation (1) to map that pixel to a pixel in the source image. However, this pixel does not have to be an integral value, so to get the non-integral value of the pixel from the source image, we use nearest-neighbour interpolation –

essentially using the value of the pixel closest to our non-integral source pixel. Due to the way how Windows handles scaling with nearest neighbour interpolation, this might lead to a poor correlation result as illustrated by our demonstration and data analysis. Nevertheless, this step is essential to the completion of our system because our match filter is not scale invariant; therefore we have to make sure that the images that we are analyzing are the same resolution as the images in our reference database.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{s} \begin{bmatrix} x' \\ y' \end{bmatrix} \quad (1)$$

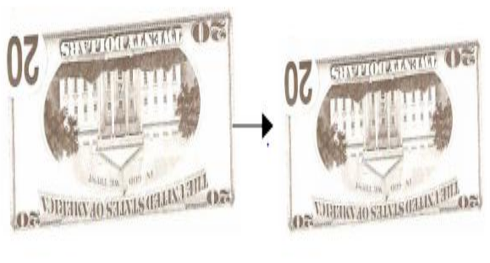

Figure 2: Rescaled Image

5. Greyscale Computation

The second algorithm that our program employs is grayscale computation, which transforms an image from the RGB to the grayscale domain. The reason that we perform this process is that it is difficult to carry out our operations in each of the color domain independently, and converting images to grayscale and working with light intensity rather than colors simplifies our problem. In addition, by using grayscale images instead of color images, we are able to save some memory. This is achieved through the use of NTSC-approved formula (2), generously provided by the Mathworks [7]. Note that this process is carried out on an inverted image, so the actual gray intensity is specified by (3).

gray = 0.3R + 0.6G + 0.1B (2) gray = 255 − (0.3R + 0.6G + 0.1B) (3)

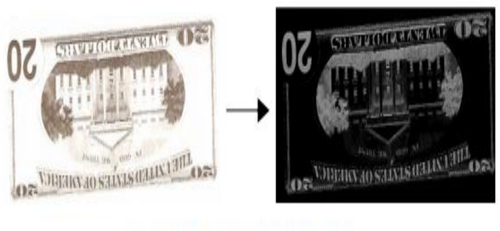

Figure 3: Object Converted to Grayscale

6. Black and White [Binarizing]

Following converting the images from the RGB to the grayscale domain, we proceed to binarize images by comparing each pixel's intensity against a threshold that is previously computed on grayscale images. After analyzing several input images, this threshold is found to be 1.5 / 255. This threshold was assumed to be approximately universal; however, during our demo, we discovered that thicker objects did significantly alter the hue of the background. Note that we perform binarization on inverted images; as a result, our background color will be black while our foreground color will be white. The reason that we decide to take this extra step is that binarized images would aid us to do our next algorithm morphological processing, which includes erosion, dilation and closing on an image as described in the section below

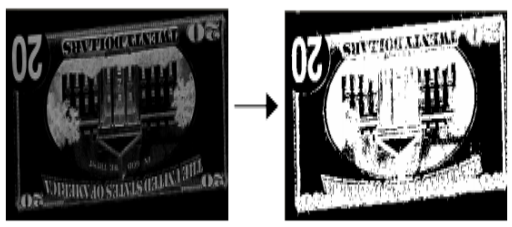

Figure 4: Object Binarized

Morphological Processing Morphological operations are used in the process of segmenting images. Image segmentation refers to the process of finding regions in an image that represent objects or meaningful parts of objects. Morphological processing can be used for filling small holes in objects, isolating adjacent or slightly overlapping objects, or joining broken boundaries into continuous segments. For our case, we are using it for joining the floating numbers on the bottom right of the back of the $5, $10, and $20 bills (Fig. 5). For our project, we perform closing which involves erosion followed by dilation, two of the principal morphological operations. Dilation allows objects to expand, thereby filling in small holes and connecting disjoint objects. Erosion shrinks objects by eroding their boundaries.

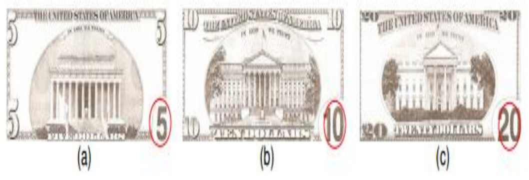

Figure 5: Problematic Reference Images of the (a) $5, (b) $10, and (c) $20 bills

The dilation process, similar to convolution in a manner, slides the overlaying structuring element across the image. Dilation does nothing to background pixels, but it grows foreground pixels by performing a bitwise OR operation on every pixel under the structural element when the center of the element is on the foreground. The dilation process is illustrated in Figure 6.

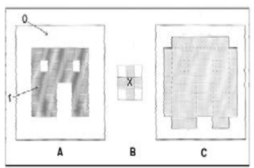

Figure 6: Dilation process – A) Original image, B) Structural element, C) Image after dilation [8]

Erosion does the opposite of dilation, thinning foreground areas, by changing foreground pixels to background pixels when all the '1's in the structural element are not present in the corresponding locations in the original image. Thus, erosion only affects foreground pixels on the edge of foreground regions.

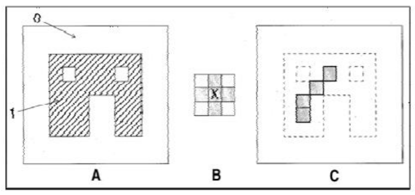

Figure 7: Erosion – A) Original Image, B) Structural element; x = origin, C) Image after erosion; original in dashes [8]

Morphological closing is simply the process of performing dilation, followed by erosion, using the same structural element for both operations. Note that the sequence of operations is important; an erosion followed by a dilation is termed “opening,” and will yield entirely different results. The size and shape of the structural element also significantly impacts the result. A square structural element will emphasize sharper edges than a circular element, and an element’s size dictates how large a section it can close. In general, an n x n structural element can close a gap of size n-1. As illustrated in the diagram below, the original image has a number of small holes and spacing in the object, which get “closed” or filled after the closing process. The size of the structuring element can influence the size of the area which the closing process closes while its shape will affect how sharp the object will end up after the closing process.

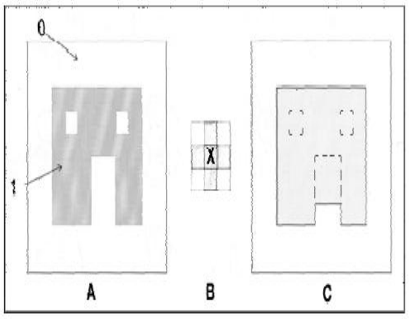

Figure 8: Closing – A) Original Image, B) Structural element; x = origin, C) Image after closing; dilation followed by erosion; original in dashes [8]

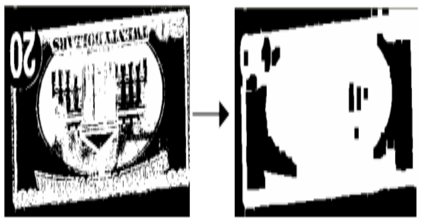
Figure 9: Morphological Processing

BlobLabelling The blob labeling technique is intended for image segmentation, particle size characterization as well as microscope magnification calibration. In our case, we are using it for the first purpose stated. Our blob labelling uses a four-way connected square blob pixels with four neighbours, which makes the background eight-way connected. Blob labelling images involves “scanning” the image in raster order, labelling pixels according to their already labelled neighbours. If there are no labelled neighbours, then a new label is used. Some of the common forms of the labels are integers stored in a new “image” or array which will become blob images and are used to be associated to the corresponding pixels in the original image. If the blobs are skinny and curved, then various ends or humps will appear as separate objects during this labeling scan, and will be joined together later on when sufficient rows have been scanned to connect all the pieces together. While this process continues, two or more different labels will be associated together as one object. This is achieved by maintaining an equivalence table during the initial labeling scan, then simplifying the table so that all labels in any one blob are known to be equivalent to one label that is eventually used for the whole blob. This re-labelling iteration is done in a second scan of the blob image. To further illustrate the methodology described above, we provide a simple transitional diagram (Fig. 10) outlining the process of blob labelling.

0	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	1	1	0	1
0	0	1	1	1

Step1

0	0	0	0	0
0	1	0	0	0
0	0	2	0	0
0	3	2	0	4
0	0	2	2	2

Step2

0	0	0	0	0
0	1	0	0	0
0	0	2	0	0
0	2	2	0	2
0	0	2	2	2

Step3

Figure 10: Blob Labeling Example

After the re-labelling iteration using the equivalence table where it states 3->2 and 4->2, the blob labelling process concludes that there are two blobs or objects in this image. In addition the following are some the properties associated with our blob labeling technique.
1. The input is an inverted binary image.
2. The output is a blob image or an ugly blob mask iage.
3. Light intensity threshold based from the binarizing process is used to identify pixels as either object or background.
The reason we incorporate the blob-labelling technique in our project is that an input image may have multiple objects contained in it; blob-labelling is a simple method to extract disjoint objects from an image [9]. In order to properly detect and classify each object, we must be able to tell them apart from one another, extract them from the original multi-object image, and make them into a separate image where only one object is present. Dealing with images where only one object is present greatly simplifies our life in terms of classification.

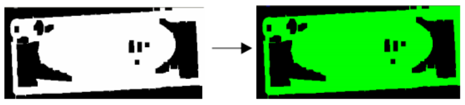
Figure 11: Blob Coloring

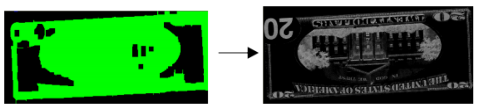
Figure 12: Extracted Object from Blob Labeled Image

Orientation Estimation using Geometric Moments We use geometric moments to estimate the orientation of rotated objects. Geometric moments are easy to implement and offer fast computation of image rotation; however, they are also highly sensitive to image noise and also exhibit large variation in the dynamic range of values, thus yielding wrong results in a correlation matching algorithm. By computing the principal axis of rotation of the input image using geometric moments as described in equations (4) and (5), we are able to conclude by what angle we need to rotate the image to align its principal axis to those of reference images (un-rotated) [10]. After obtaining the rotational angle, we can proceed to un-rotate the image using bilinear interpolation.

$$m_{p,q} = \sum_{y=-\infty}^{\infty} \sum_{x=-\infty}^{\infty} x^p y^q I(x,y) \quad (4)$$

$$\tan 2\theta = \frac{2m_{1,1}}{m_{2,0} - m_{0,2}} \quad (5)$$

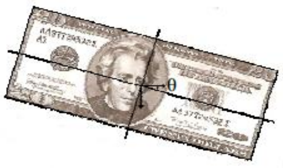

Figure 13: Principle Axes of a Rotated Image

Un-rotation Using Bilinear Interpolation Before two images can be compared using linear correlation, their principle axes must be properly aligned, since the linear correlation operation is highly sensitive to rotation. Image rotation is generally performed by mapping the un-rotated pixels to pixels in the input image using a transformation matrix. Since we are performing un-rotation on digital images, the un-rotated image cannot be a perfect representation of the original image except in certain specific circumstances (rotation by any multiple of 90 degrees, or if the input image is rotationally symmetric). This is a result of the fact that the transformation matrix does not constrain the coordinates in the original image to integral values. Therefore, we must resolve these non-integral values using a form of interpolation; we choose bilinear interpolation because of its smoothing properties and ease of computation. To un-rotate an image, one simply maps every output pixel to the corresponding input pixel using the following transformation (6):

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \quad (6)$$

The simplicity of this formula lies in the pre-computation of the cosine and sine of theta, which reduces rotation to simple additions and multiplies.

Once we have the corresponding input pixel, we have to perform bilinear interpolation to compute the value at that pixel (and the corresponding output pixel). Bilinear interpolation performs the weighted average of the four surrounding pixels in the input image using the area overlap of the input pixel over each surrounding pixel as the weights for each surrounding pixel. Some pixels lie along the edge of the input image, in which case, some surrounding pixels may be outside the valid image area. To produce correct results, one simply must use the value of the nearest valid pixel for those surrounding pixels outside the valid image area. Assuming a mapping from the destination image to the source image such that the desired pixel lies (x, y) away from the upper-left nearest source pixel, the value of the bilinear interpolation, p, is computed according to

$$p = p_{0,0}(1-\Delta x)(1-\Delta y) + p_{0,1}(1-\Delta x)(\Delta y) + p_{1,0}(\Delta x)(1-\Delta y) + p_{1,1}(\Delta x)(\Delta y) \quad (7)$$

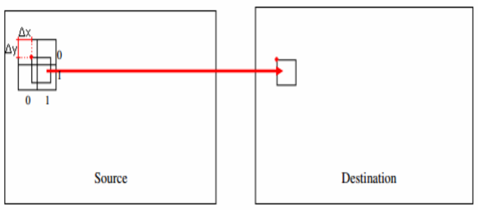

Figure 14: Bilinear Interpolation

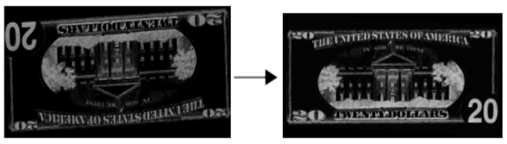

Figure 15: Image Un-rotation

Implementation The implementation of our system is done is matlab using image acquisition tool box and machine learning.

Implementation of Algorithms When the reference database is loaded (during the start of the program), each reference image is converted to an inverted grayscale image (resolution assumed to be 100dpi) from whatever form it was before. Currently, our software supports 24-bit and 32-bit bitmaps. Once the inverted grayscale image is computed, the rotational angle of each reference image's principle axes relative to the Cartesian coordinate system is determined using geometric moments as described in the algorithm section above. This process is performed only once, during program initialization, so we don't count it towards our processing time.

Once an image has been acquired and is loaded into program memory, we must execute a preset procedure for detecting and extracting all objects of interest in an image. Because of the size of our reference images, we will only consider those objects whose each dimension is within the range of 100 and 500 pixels; other objects are considered to be too big or too small to be money and will not be tested. Each object, as it is extracted from the scanned image, is sent to the KNN for processing and classification

The first step of our algorithm involves resizing the scanned image appropriate so that its resolution matches that of images in the reference database, i.e. 100dpi. In order to achieve this goal, we use the CopyImage() function and specify the desired width and height. This function uses whatever interpolation method deemed appropriate by the Windows operating system; we assume it uses nearest-neighbor interpolation, as described in the Algorithms section of our paper. Note that this yields poor resizing results, leading to poor recognition of objects; we recommend, at least for our demonstration software, that all images be scanned at 100dpi to maximize the effectiveness of our classification technique. Two solutions to this problem which we did not have the time to explore include better interpolation (such as bilinear or bicubic) or constructing a reference database with images at each resolution supported by the scanner. This process works on any color depth, so we use the original scanned image for this step.

The next two steps, morphological processing and blob-labelling, both operate on binary images. The EVM requires a grayscale image of each extracted object. These algorithms also assume that the background is zero and the foreground is nonzero, the opposite of what is the case in most scanned images. Therefore, we must convert the color image (24- or 32-bit) to both grayscale and binary, as well as invert the image, using the algorithms described in the Algorithms section. Our implementation for both uses the inverted form of the scaled color image, since the binary threshold does not have to be an integral value. In our tests, we determined that a threshold of 1.5 out of 255 properly binarized scanned images as long as the hue of the background was not significantly altered by scanning a thicker object (such as a book or even an ID card). A possible solution to this shortcoming would be to automatically detect the background threshold of each scanned image during runtime; however, this was not the primary focus of our project, and when one attempts to counterfeit money, he or she wants as little noise present in the scanned image as possible; the added hue takes the form of background noise.

Once the grayscale and binary versions of the inverted scaled image are available, our implementation performs morphological closing to attach foreground regions separated by the binarizing process. We perform morphological closing according to the algorithm described above, using a 5x5 square kernel for both erosion and dilation. This procedure modifies the binary image to reflect the result of the closing operation. The morphological processing code is based on the code from the IM image processing library [14].

Once closing has been completed, our preprocessing is almost complete. The remaining phase on the PC involves detecting objects in the scanned image and sending the individual objects to the EVM for processing. This procedure involves blob-labeling the binary image resulting from the closing operation performed above. Blob-labeling, as described in the Algorithms section, involves two passes through the binary image, and the use of a map data structure to store identifier mappings for joined regions. Although perhaps not terribly efficient, our iterative implementation of the algorithm essentially eliminates the possibility of stack overflows typical with a recursive implementation during the processing of large images. Each binary pixel is replaced with a 32-bit number identifying the object to which that pixel belongs (zero = background).

We can then extract each object from the blob-labeled image in at most N passes through the entire image, where N is the number of objects. To create our extracted object image for a particular object, we locate all the pixels that belong to that object, compute the bounding rectangle from those pixels, create a new grayscale bit-map the size of the bounding rectangle, and copy the grayscale pixel values from the inverted scaled grayscale image for each pixel described by the blob-labeled image as belonging to that object. We assume that objects are actually whole, so that every pixel in a row between two pixels belonging to an object belongs to that same object, so we copy whole pixel ranges into our extracted image.

Each extracted object image, along with its dimensions and size (in bytes), position in the original image, and other useful information, are saved in a simple data structure for extracted objects and placed at the end of a queue of images to be processed. The processing occurs asynchronously; the communication with the EVM occurs in a separate thread to be discussed in the next section.\

Data Analysis Before we conducted our data collection, we compiled several databases: a reference database for classifying detected objects, a training database for estimating parameters, and a test database for testing our system. Each database contains images, both front and back, of play money. We use play money to avoid the legal issues involved with digitizing federal reserve notes. All images are in color (converted to grayscale and binary where required) and have a resolution of 100dpi. Our reference set consists of one image for the front and back of each bill ($1, $5, $10, $20), in total, 8 images. The reference images are completely un-rotated and have (approximately) no noise. Our training set consists of several images of each bill, at different rotation angles. This set is used to assist us in determining our KNN thresholds and quantifying the required rotation sensitivity. This set consists of 23 images. Our testing set consists of several images of each bill, not included in the training set, at different angles of rotation. In addition, the testing set contains images with multiple objects, a mixture of play money and other objects commonly scanned (photos, business cards, etc.). This set consists of 22 images. The first test we implemented was utilizing the peaks of an objects correlation with our database to aide in our classification system. We believed that the peaks for each object would be sufficient enough to discriminate between positive classifications (a match for the object in question) and negative classifications (the other bills in the database, as well as other images). Figure 20 shows the results from our testing. Figure 20 shows that there can be negative classifications with a higher peak than a positive classification, which would make it nearly impossible to determine a useful threshold between the two. Therefore, we decided to look elsewhere for a better test that would help our classifier. To improve on our previous test, we decided to see how effective peak-side lobe ratios are in our classification system. The KNN implementation takes into account not only the peak of the object's correlation with our database, but also those of the surrounding neighborhood. Figure 21 shows how effective KNNs are in discriminating between positive classifications and negative classifications. Figure 21 shows a clear delineation between the positive and negative classifications, which would make it easier to determine a threshold between the two. Since the KNN implementation seemed to work well, we decided to keep it in our classification system.

Towards converting from a grayscale to a binary image, we must compare the grayscale intensity as computed by the NTSC formula to a cutoff intensity (out of 255.0). This next experiment assisted us in determining an acceptable background threshold. In MATLAB, we took several images and converted them to black and white according to any of several threshold values. After experimenting with several images, we determined the threshold to be one constant value over all input images, approximately 1.5.

In addition to the binary threshold, we also had to determine the size of the structural element used by our morphological closing algorithm. The structural element must fit two criteria: it must be large enough to effectively reunite objects separated during binary

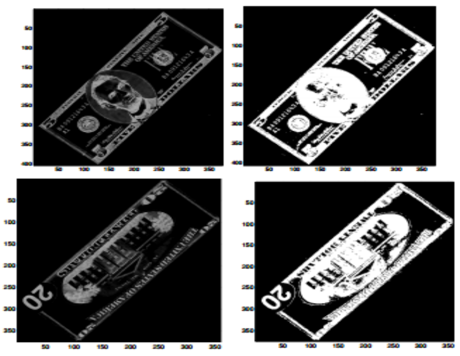
Figure16: Binary Thresholding of Various Images

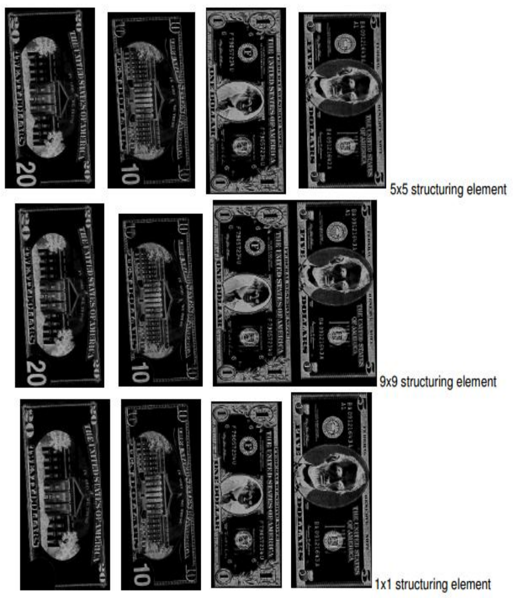

Figure 17: Results of Morphological Closing using Different Structural Elements

thresholding, and it must be small enough not to unite \objects far enough away from each other that we would consider them distinct. We experimented with several different kernel sizes:

Evidently, the 5x5 structuring element fits both criteria in the image we chose. The 9x9 element successfully reunites separated objects (i.e. the number on the bottom right of the back of the $20 and $10 bills) but joins objects that are definitely distinct (the $1 and $5 bills). The 1x1 structuring element does not successfully reunite the number on the bottom right of the back of the $20 bill. Once we completed these experiments, we were ready to determine whether or not our entire system worked. We ran our set of test images through our software and recorded which images were handled correctly and recorded the results. Out of the 43 correlations from our test database, there were only two false misses. This shows that our system is approximately 4.65% inaccurate, but it correctly biases towards allowing money through rather than preventing legal images from being scanned or copied. We demonstrated several images, containing both money and legal objects. Tests during our demonstration confirmed the expected functionality; however, we noted that one test involving a group member's ID card failed to produce either correct or consistent results. On two out of three attempts, the card was classified as either the front of the $5 bill or the back of the $10 bill. We found that although the KNN was abnormally high for these false hits, the magnitude of the correlation peak was relatively small; consequently, we modified our software to validate both the correlation peak and the KNN during image classification. This removed the false positives without demonstrably affecting the efficacy of our software.

7. Conclusion

There are many methods for identifying a fake note which we have discussed and each one has its own significance. One should be cautious while detecting a fake note. Our paper enables a layman to identify a fake note and empower every citizen to detect fake notes which may reduce corruption in our country. Our MATLAB technique when deployed in mobile phone with a scanner or camera so that it will detect fake notes which gives the power to a common man to control fake currency circulation in our country.

8. Future Scope

MATLAB is predominantly used for Simulation purpose, to design algorithms and test mathematical models for hardware implementation like Signal processing that can run one-time operations for large data sets. Matlab specializes in Matrix operations which is very important for researchers to modify its features for its high performance measures.

References

[1] Central Bank Counterfeit Deterrence Group. Available online http://www.rulesforuse.org/. Accessed 9 Oct, 2005.

[2] Ulbrich, Chris. “Currency Detector Easy to Defeat.” Wired News. Available online

[3] http://www.wired.com/news/print/0,1294,61890,00.html. Accessed 8 Oct, 2005.

[4] Resende, Patricia. “MediaSec seeks more than the eye can see.” Mass High Tech. Available online http://www.masshightech.com/displayarticledetail.asp?art_id=59816. Accessed 9 Oct, 2005.

[5] “Know Your Money.” United States Secret Service. Available online http://www.treas.gov/usss/money_illustrations.shtml. Accessed 11 Dec, 2005.

[6] Pearson, Chris; Tsao, Amy; Yu, Christina; Theisz, Matthew. “Where’s the Ball?” 18-551, Spring 2004. Availableonline :https://blackboard.andrew.cmu.edu/courses/1/F05-18551/content/_108177_1/stheBallfinalreport.pdf. Accessed 6 Nov, 2005.

[7] Baliga, Avinash; Bang, Dan; Cohen, Jason; Schwicking, Carsten. “Face Detection for Surveillance.” 18-551, Spring 2002. Available online https://blackboard.andrew.cmu.edu/courses/1/F05-18551/content/_108213_1/Group2Final.pdf. Accessed 6 Nov, 2005.

[8] “Technical Solutions Solution Number: 1-1ASCU.” The Mathworks. Available

[9] http://www.mathworks.com/support/solutions/data/1-1ASCU.html?solution=1-1ASCU. December 4, 2005.

[10] Umbaugh, Scott E. “Morphology Overview.” Computer Vision and Image Processing. Prentice Hall PTR, 1998. Available online

[11] http://zone.ni.com/devzone/conceptd.nsf/webmain/8CA8DE2E888 1C1AB8625682E0079CE74. Accessed 11 Dec 2005.

[12] Gonzalez, R. C.; Woods, R. E.; Eddins, S. L. Digital Image Processing Using MATLAB®. Pearson, 2004.

[13] Hu, Ming-Kuei. “Visual Pattern Recognition by Moment Invariants.” IEEE Transactions on Information Theory. IEEE, Feb, 1962.

[14] Yi Li; Zhiyan Wang; Haizan Zeng. “Correlation Filter: An Accurate Approach to Detect and Locate Low Contrast Character Strings in Complex Table Environment.” IEEE Transactions on Pattern Analysis and Machine Intelligence. IEEE, Dec, 2004.

[15] Savvides, M.; Kumar, B. V. K.; Khosla, P. “Face Verification using Correlation Filters.” Available online http://www.ece.cmu.edu/~kumar/Biometrics_AutoID.pdf. Accessed 11 Dec, 2005.

[16] “Windows Image Acquisition (WIA) 1.0.” Microsoft. Available

[17] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wia/wia/overviews/startpage.asp. November 16, 2005.

[18] “IM – An Imaging Tool.” Tecgraf – Computer Graphics Technology Group. PUC-Rio, Brazil. Available http://www.tecgraf.puc-rio.br/im/. November 16, 2005.