2-Dimensional Visual Code Marker Detection Algorithm
EE 368 Final Project
Spring 2006

Julie E. Steinbrenner

***Abstract*—An algorithm for detection and decoding of 2-dimensional visual code markers is developed and implemented in MATLAB, according to a similar procedure as that described by Gohs [1]. The algorithm accepts the RGB values from a .jpg image and returns the location of visual code markers and the values of each of the 83 bits of binary information coded in the marker. Processing steps include adaptive thresholding, region labeling and property calculation, candidate region determination, and marker detection. Ten training images were used for testing and development of the algorithm. The current algorithm successfully detects all of the code markers in 8 of 10 training images. Reasons for the failure in the remaining two images are discussed. Sample results and recommended improvements are presented.**

I. Introduction

Visual code markers have been proposed for a variety of applications where humans would like to access to a significant amount of information using a cellular phone digital camera or similar portable device. While a variety of visual code marker formats have been proposed, thus far no particular design has emerged as a standard [2]-[4]. Gohs has proposed a rectangular 11 x 11 matrix with two guidebars and three cornerstone pixels as shown in Fig. 1 [1]. This combination of simple visual features may be used to extract many important parameters of the code marker image above and beyond the location and 83-bit code of the marker, including the tilt, rotation, and motion of the marker.

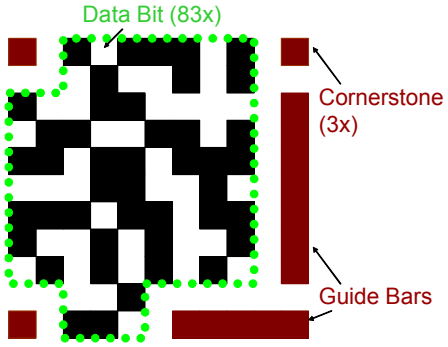

Fig. 1. Diagram of the structure of the Gohn 2-dimensional code marker. (Color added for identification.) Two guidebars are located in the lower right corner of the marker for identification and orientation. Cornerstone elements at the remaining three vertices aid in determination of the boundaries of the marker and its out-of-plane orientation. The origin is located at the center of upper left cornerstone.

For this project, a simple algorithm is developed to read the location and value of the marker, assuming a small tilt angle such that the right angles of the code marker are preserved within ± 30 degrees of orthogonal. Further development could relax this assumption.

II. Discussion of Alternative Algorithms

Various techniques were considered in the development of this approach based on techniques introduced in EE 368 and presented in the literature for bar code recognition. Initial implementation of these techniques revealed issues which excluded their selection for this task.

A brute-force method to convolve the image with a template of the guide bars and cornerstone pixels proved to be too time-consuming. While a series of hough transforms might be successfully implemented to rotate images to match the template orientation, the convolution step encounters several fatal obstacles, including extreme sensitivity to the template magnification, necessary compensation for image intensity variations, computational expense, and the requirement for a robust two-dimensional peak-finding routine between multiple image orientations. This brute-force method was not chosen due to these complications.

An interesting method proposed by Schulze isolates regions of interest where a code marker may occur by finding those regions where many strong intensity gradients are present [5]. However, this method was demonstrated in environments with random backgrounds containing mainly weak gradients. This assumption may not be valid for the images of interest to this project, therefore another method was chosen.

Several papers introduced methods which identified particular features of the code marker by a distinguishing characteristic which establishes its alignment [1], [6], [7]. The data bits are read from the image based on their position after this transformation. This is the method that was implemented according to the general procedure outlined in [1].

III. Algorithm Procedure

The nominally orthogonal guide bars of the code marker are used to distinguish the markers from other features in the image. Then the location of the origin of the code marker is

determined as the cornerstone locations are found one-by-one. The details of the algorithm are explained in this section.

A. Image Thresholding

The input RGB components are converted to grayscale values according to the approximation of averaging the Red and Green values to save computational resources [1]. A threshold was applied to the grayscale image in order to create a binary image for the remainder of the processing. An adaptive algorithm was required due to the inconsistency of illumination and background light intensity [8], [9]. The adaptive technique sets a local threshold value equal to 95% of the local average intensity. Local average intensities are calculated on a 20 x 20 mesh spanning the length and width of the image. This mesh is linearly interpolated to set the threshold at each pixel. This threshold technique was effective for all test images, leaving the important features of all code markers clearly visible. Fig. 2 shows a sample image converted to grayscale and then binary with the adaptive technique. The local threshold map is also shown. After thresholding, distinct regions in the binary image are labeled.

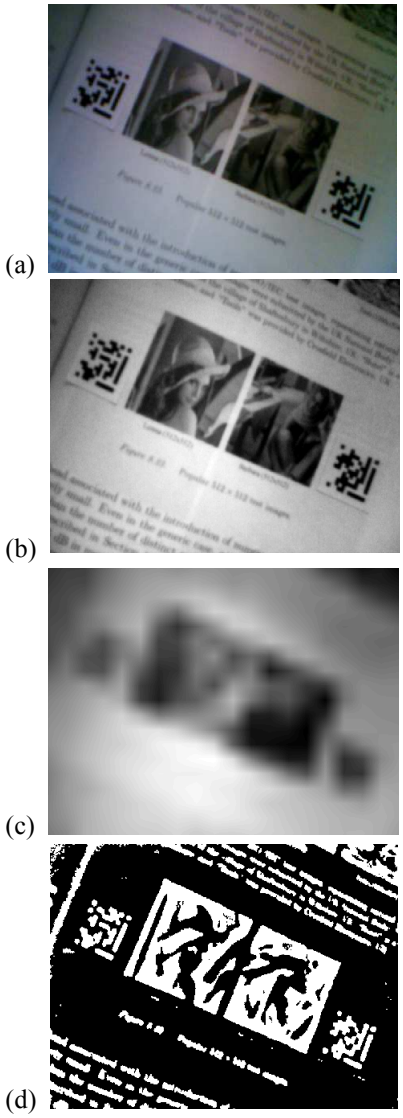

Fig. 2. Example images of preprocessing steps. (a) original image (b) grayscaled image (c) adaptive threshold (d) result of negative thresholding

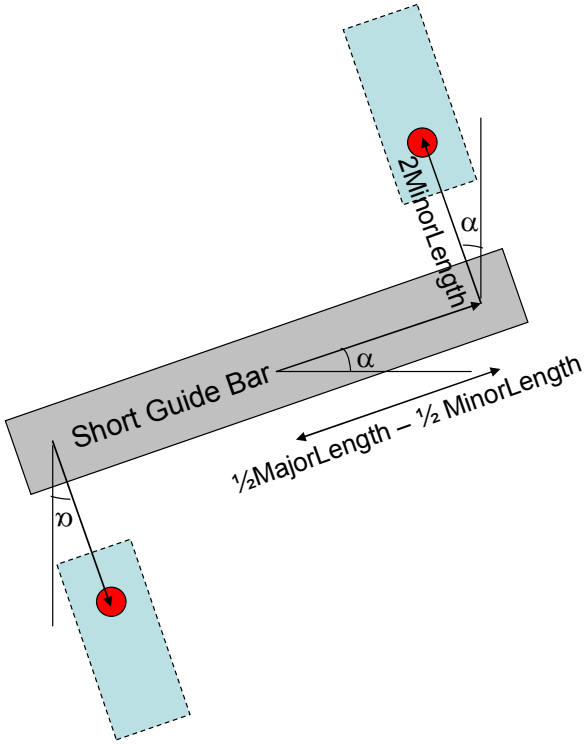

Fig. 3. Diagram of relative location of long guide bars from the centroid of the short guide bar. The red circles show the location which is evalulated to determine the presence of a long guide bar.

B. Guide Bar Identification

A distinguishing characteristic of the code marker guide bars is their well defined aspect ratio. The short bar on the bottom of the marker has a nominal aspect ratio of 5:1; the longer side bar has a nominal aspect ratio of 7:1. The aspect ratio of each labeled region in the image is calculated using the internal MATLAB function *regionprops* to calculate the 'MajorAxisLength', the 'MinorAxisLength', and the 'Orientation' of each region. Candidates which might be either a long or a short guide bar are chosen if the ratio of these lengths approaches the nominal aspect ratio. The requirement is relaxed to an interval around the nominal aspect ratio because it must allow for distortion due to image thresholding and camera angle.

Next, each short guide bar candidate is analyzed to identify whether there is a long guide bar in the correct relative position. This will establish the presence of a code marker. This procedure is accomplished using a geometric transformation assuming a 90˚ angle between the guide bars. Fig. 3 shows a diagram of the two locations tested for the presence of a long guide bar, relative to the centroid of the short bar.

A small region of pixels near these locations are sampled in order to identify any labeled regions in the vicinity. If this region corresponds to a long bar candidate, the relative angle between these two regions are calculated to determine if these regions are nearly perpendicular. This technique eliminates parallel rows of rectangular shapes which may arise from thresholded photos of text. If the regions are perpendicular within ± 30°, they are identified as a code marker candidate.

C. Cornerstone Marking

The three other corners of the code marker need to be located in order to find the origin of the marker in the image and to establish a grid for determination of the value of each of the data bits. The cornerstones on the two sides adjacent to the corner with the guide bars. A similar technique as employed to identify the long bar candidate is used to find the cornerstone. The center of the search location for the upper right corner is determined by extending a line from the centroid of the long bar, aligned with its orientation, past the end of the long bar by 1.5 times the width of the long bar to land on the expected center of the cornerstone. A similar calculation is performed from the short bar in order to find the lower left cornerstone.

The origin is final marker to be located. The center of the search area is determined on the assumption that the edges of the code marker are approximately parallel. The estimated location of the origin is determined by establishing two sides of a parallelogram from the previously identified three corners and extrapolating the location of the fourth. The location of the origin is the centroid of the region found in the vicinity of this search area.

D. Data Pixel Reading

Once the four vertices of the origin have been located, a grid is established to probe each of the data pixels. The mathematics of this geometric interpolation are extracted from Rohs paper [1] and then formatted into the output array required for this analysis.

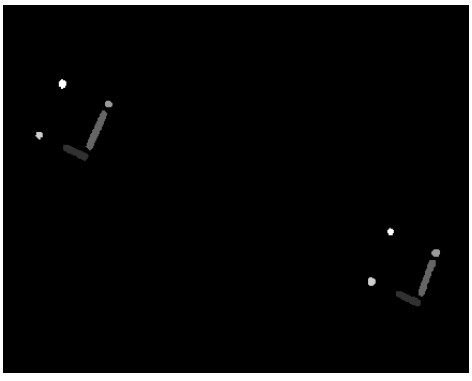

Fig. 4. Correctly identified guide bars and cornerstones for both code markers in sample image. Gray shades identify the short guide bar, long guide bar, and each of the three cornerstones.

IV. Test Image Results

The algorithm was tested on ten sample images. Fig. 4 shows the guide bars and cornerstones identified for the sample image. Correct results such as this example were found for 16 of the 20 code markers in the 10 ten images. Fig. 5 summarizes the success rate of code marker location identification by image. Image 3 had 0/3 markers identified. Image 9 had 2/3 identified. Otherwise, every code marker was identified successfully.

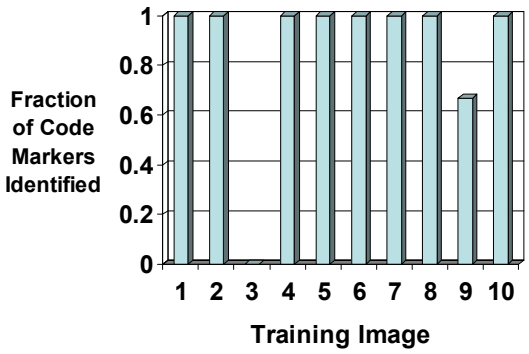

Fig. 5. Fraction of code markers identified in each of the 10 training images for a total of 16 of 20 markers properly identified. Image 3 had no success in identifying code markers due to the horizontal orientation of the code markers. Figure 9 had one unidentified marker.

Among successfully located code markers, there was a 100% success rate among data pixels read using the geometric transformation based on corner pixel locations. Fig. 6 shows the code marker maps extracted from the processed sample image, which identically map to the code markers visible in the original image. The average execution time per image was less than 3 seconds.

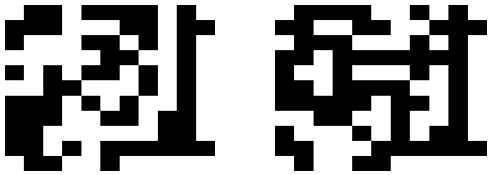

Fig. 6. Code marker data bits extracted from image processing, matching those seen in the original image. (Shown in negative)

V. Discussion

This decoding algorithm has a high success rate for evaluation of the sample images. The adaptive thresholding method appropriately converts the image to binary, where a single global threshold value would suffer from variations in illumination and background intensity.

The guide bar recognition scheme capably identifies pairs of guide bars in most images based on the aspect ratio of the regions and their relative locations. This scheme suffers

from variations in aspect ratio due to camera angle and thresholding uncertainty, however this weakness is overcome by increasing the bounds of acceptable aspect ratios.

The cornerstone identification scheme is successful most of the time in finding the cornerstones, but it relies on the assumption that the code marker is rotated, not skewed so that the angles in the image are close to perpendicular. The scheme is also sensitive to the extent of the region that is searched in order to find the cornerstones. These values were set to optimize the success rate for the training image, but this search scheme should be improved to an adaptive method to improve the robustness of the algorithm.

The failure to identify cornerstones in image 3 was the result of the rotation of these images which caused an error in the determination of the search location for the cornerstones. The cause for this error was not immediately apparent and requires further investigation into geometries at this angle of rotation.

The failure in image 9 was also the result of a misidentification of one of the cornerstone markers. This occurrence further supports the need for a better search algorithm to find cornerstone regions.

However, the data pixel reading routine gave very good results. In every instance that cornerstone identification was successful all 83 data pixels were correctly identified.

VI. Conclusion

This algorithm forms the basis for an effective algorithm to quickly identify code markers in images taken with a cell phone camera. Despite a few bugs which require more attention, the algorithm successfully identified code markers and read their values for the images tested.

References

[1] M. Rohs, "Real-World Interaction with Camera-Phones," *UCS*, 2004.

[2] H. Kato and K.T. Tan, "2D barcodes for Mobile Phones," *2nd International Conference on Mobile Technology, Applications and Systems,* 15-17 Nov. 2005, pp. 1-8.

[3] E. Ouaviani, A. Pavan, M. Bottazzi, E. Brunelli, F. Caselli, M. Guerrero, "A common image processing framework for 2D barcode reading," *Seventh International Conference on Image Processing and its Applications*, 13-15 July 1999, pp. 652-655.

[4] E. Ohbuchi, H. Hanaizumi, L.A. Hock, "Barcode Readers using the Camera Device in Mobile Phones", *Proceedings of the 2004 International Conference on Cyberworlds*, 18-20 Nov 2004, pp. 260 – 265.

[5] K. Schultz. "Autonomous Underwater Barcode Recognition", *Proceedings of SPIE*, vol. 5203, pp. 148-154.

[6] H.I. Hang, J.K. Joung, "Implementation of Algorithm to Decode Two-Dimensional Barcode PDF-417", *International Conference on Signal Processing,* 26-30 Aug 2002, pp. 1791-1794.

[7] H. Uno and Y. Hayasaki. "Two Dimension Bar Code Recognition Method and Stamp Pattern Recognition Method by Using Image Pattern", *NEC Research and Development*, Apr 1999, pp. 165-170.

[8] J. Sauvola, M. Pietikainen, "Adaptive document image binarization", *Pattern Recognition*, 2000, pp. 225-236.

[9] F.H.Y. Chan, F.K. Lam, H. Zhu, "Adaptive Thresholding by Variational Method", *IEEE Transactions on Image Processing*, 1998, pp. 468 – 473.