Visual Code Marker Detection

Miguel A. Guerrero (maguerre@stanford.edu)

***Abstract*— This report describes an algorithm designed to detect visual code markers as described in [1]. The algorithm assumes the images have resolution and quality similar to the ones captured by regular cellular phone cameras. Applications of the use of these visual code markers are described in [2].**

I. INTRODUCTION

THE objective of this work is to design a robust algorithm for the detection of visual code markers in low resolution images (640x480 typical). These code markers consist of a 2D array of 11x11 bits, out of which 83 contain valid information, while the remaining are pure marker reference bits with known values, intended to allow the recognition algorithm to uniquely identify the location and orientation of the code markers in a given image.

Fig. 1 shows the structure of a visual code marker. The bottom right corner is identified by a mirrored L shaped two-bar layout. The other three corners contain isolated reference marker bits. The highlighted center portion contains the information bits. The structure of the visual code marker allows uniquely identifying its orientation and position. For a more detailed description of the visual marker structure refer to [1].

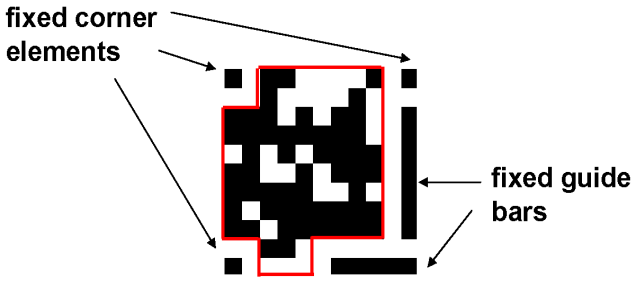

Fig. 1. Structure of a visual code marker

This report is structured as follows: section II describes the proposed algorithm. Section III justifies some of the choices vs. possible alternatives. Section IV describes the testing performed and results obtained. Section V gives a summary of the strengths and weaknesses of the algorithm and items for further improvement.

II. THE ALGORITHM

A. *Code marker location/orientation detection*

The algorithm is divided in four stages. The first stage of the algorithm targets the accurate location of the visual code markers in the image. The output of this stage is a list of code marker positions. Each one contains the image coordinates of the center of each of the four corners of the visual code marker.

Fig. 2. Code marker details of one of the training images.

The first stage consists of the following steps:

1. Conversion from RGB to YCbCr. The algorithm uses only the luminance component of the image.
2. Edge detection. The image is processed with the Canny edge detection algorithm [3]. The resulting image is a bi-level image where the contours of the image and specifically the reference markers are easily identifiable.

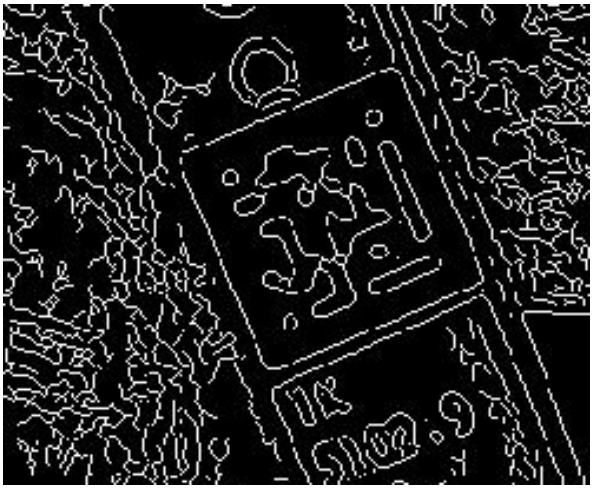

Fig. 3. Canny edge version of Fig 2.

3. Segmentation. Contours are segmented with the objective of identifying possible reference markers and discarding the remaining ones. The output of this step is two lists (A and B). Each list is filled-up according to the following criteria:
    a. List A: shapes with aspect ratio like any of the two fixed guide bars at the bottom-right corner of the visual code marker and having one or zero loops (to allow for small gaps)
    b. List B: shapes resembling markers at the top-left, top-right and bottom-left corners of the label, having aspect ratio close to unity and having one or zero loops (to allow for contours whose gap has been completely closed or small gaps)
4. For every element X in List A we perform the following steps:
    a. Follow the major axis direction in pixel space and record crossings with any other element belonging to list A.
    b. If a crossing is found within a range (function of the length of X) with another class A element, check if
        i. The two elements are distributed as a per the mirrored L arrangement by checking the angle they form and comparing to 90 degrees plus a tolerance value (to account for perspective deformations)
        ii. Check if single bit markers are found on the other three corners. This is performed extrapolating their expected position based on the measured length of the two class A elements and the angle they form with each other.
        iii. For each expected position, a search is done in pixel space around it within a range, with increasing Manhattan distance from the initial point. The first class B element found that has a convex area (i.e. area of its convex hull) in proportion to element X is returned.
        iv. If a class B element is found for every one of the three remaining corners, the centroid of the three corner contours is recorded and a code marker is considered potentially detected (subject to subsequent pruning on stage C), else is rejected.

Fig. 4. Contours that passed shape filtering. Red crosses represent centroids of each of the contours (present or filtered). In blue, detected top-left marker corner. The yellow squares represent the other 3 corners. Red lines represent the pixel space searched pixels surrounding each List A element.

B. Code marker contents recognition

The second stage of the algorithm targets the measurement of each of the bits of the visual code markers detected by the previous stage. Each measurement will produce two levels of confidence on the measurement. Stage B consists of the following steps:

1. For every detected label, bit detection is performed
    a. A sampling grid of 11x11 is created within the 4 detected corners. This grid divides each of the four sides of the quadrilateral defined by the four corners of the code marker in 10 equally spaced segments.
    b. Center bit pixel value is extracted by performing bilinear interpolation on the desired bit center position (as opposed to nearest pixel to the desired center).
    c. A threshold is calculated as the middle point between the maximum and the minimum gray level value extracted in the grid for all center bit positions.
    d. Two measures of confidence are created
        i. Conf_Rough: Number of marker bits matching expected values / total number of marker bits.

ii. Conf_Fine = min(abs(bitValue[i] – threshold))/128.

e. Conf_Rough is expected to be very close to 1 (all reference marker bits, with known value, properly detected)

f. Conf_Fine can be expected to be as low as 10% of the maximum possible value (1 indicating maximum variation around the threshold)

g. If Conf_Rough != 1 and Conf_Fine<0.10, the bit detection is re-done by shifting the grid 1 pixel on each direction on a 4-neighood of distance 1, recording the code value that yields highest confidence values.

2. Step 1 is repeated after performing *local adaptive histogram equalization* in a region of interest surrounding each label (defined by its bounding box plus a small 4 pixel margin) according to algorithm [4]. See Fig. 5.
3. If results of 2 are better (both confidence values higher) than the ones obtained on 1, they are taken, else results from 1 prevail.

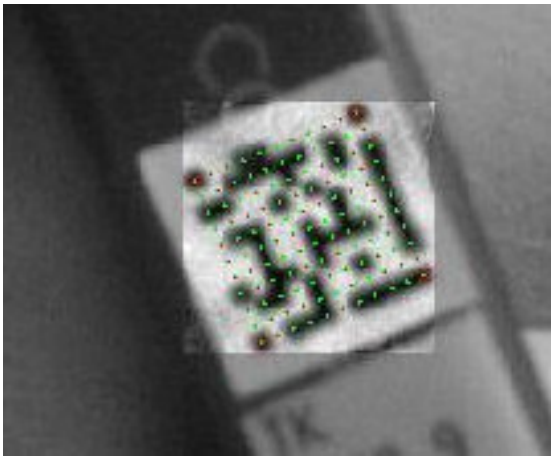

Fig. 5. Detail of local adaptive histogram equalized code marker with bit centers highlighted. The bounding box of the region of interest is defined by the four corners of the visual marker.

C. *Pruning*

Simple pruning is performed to reject false positives given by stage A.

1. If only a code marker has been found, no pruning is performed.
2. Else (two or more code markers found) code markers with Conf_Rough confidence bellow a threshold are pruned as they are likely a false positive. The rough confidence measurement is the ratio of marker bits properly detected vs. the total number of marker bits (38, see Fig. 1). The algorithm allows for two bit errors. More than two bit errors on marker bits is considered a false positive as it is not very likely for medium quality images. Detected false positives are removed from the final results.

D. *Fold-back*

If after performing steps A, B and C no label is detected, steps A, B and C are repeated after performing a global histogram equalization step to the image.

III. CHOICES AND TRADE-OFFS

A. *Missing steps*

Notably missing are the following pre-processing stages

- De-blurring – Canny edge detection algorithm performed well even with blurry input images, so this step didn't seem to add value.
- De-noising:
  - Salt/Pepper noise: not expected in the type of images for current application
  - Gaussian noise: noise can become an issue in low light images. De-noising is performed only during the bit detection stage in the neighborhood of the sampling points. This step is conditionally performed only if the size of the detected code marker ensures a bit area large enough to perform this filtering. This way the blurring effect of the de-noising does not impact negatively prior steps
- Initial global histogram equalization – Global histogram equalization is performed only on difficult images as a fold-back option, as it was observed to have and adverse effect in some of the images tested.

B. *Design choices*

Processing is done in luma (Y) space discarding chrominance information, to make it more robust to different lighting conditions. The black and white structure of the code markers could be useful but we decided not to use it for robustness.

Canny edge detection proved to be very robust in giving closed contours for the reference markers of the visual code labels in most cases. Other edge detection algorithms tried (Prewitt and Sobel kernels) underperformed Canny in the experiments.

The core of the algorithm is detecting the mirrored L shaped arrangement at bottom-right corner of each visual code marker. The main criteria is aspect ratio (major to minor axis length ratio), general shape (single or no loop), convex hull area measurements within specific ranges (function of picture size/resolution) and among different elements of the marker, and finally relative position (aprox. 90 degrees) of the contours. Other approaches to detect code marker orientation based on Hough transform were initially tried but discarded due to the noisy measurements obtained were the interesting peaks were low in magnitude relative to unrelated clutter (long lines, multiple aligned short segments, etc.).

Initially the closed loop nature of the reference marker

contour was a condition for it to be considered a potential reference marker. During stress testing this condition was removed as in many cases the contours had small gaps. Gap closing approaches based on morphological operators like dilation and closing were also used but limited the performance at even lower resolutions, where undesired gaps would end-up closed resulting in unwanted contour merging. The final approach is less constraining in this initial phase leaving some of the work for a later pruning stage.

The detection of the second element of the mirrored L in the bottom-right corner reference arrangement is performed by searching in pixel space on a small neighborhood (aprox. equal to the major axis length of the element on each direction from its centroid). This approach is O(n) with the number of List A elements. An approach based on finding possible intersections of the major directions of pairs of List A elements would be $O(n^2)$ and was purposely avoided to avoid high variability on execution time with image contents. It was also observed that pair-wise processing would yield more false positives than the pixel space search approach.

Bit detection is performed interpolating a uniform grid where each of the four sides of the quadrilateral composed by the reference markers is divided into 10 equal segments and remaining are linearly interpolated. Perspective correction was considered but not implemented finally due to the low return expected (only for pathological images where perspective is exaggerated). The shifting of the grid around the original pixel positions proved to be sufficient for the cases tested.

Local adaptive histogram equalization (in a slightly augmented bounding box surrounding the label) outperformed global histogram equalization and global adaptive one.

Finally, bilinear interpolation over center pixel positions outperformed nearest neighbor. This approach achieves some noise resilience by typically using the contribution of four neighbors to the bit detected value. Additionally pixel averaging (low pass filtering) is also performed on a 3x3 kernel when the estimated bit size allows it (above a threshold of pixels) to improve noise resilience in low luminosity images. This estimated bit size is derived from the overall code marker size based on the coordinates of its four corners.

IV. TESTING AND RESULTS

The algorithm was initially tested over an initial basic training set of 12 640x480 color images available for download in [1] until zero error rate was achieved. Further testing was performed to increase the robustness of the algorithm by generating rotated images of the above over 360 degrees using bilinear and nearest neighbor interpolation methods until zero error rate was achieved as well. Additional testing was done by scaling down this set of images to emulate lower resolution or smaller labels down to a factor of 0.6x the original size. The results of these experiments are reported in Table I.

Further testing was performed over an augmented set of training images (48 additional) to test the robustness of the algorithm under different lightning conditions. This extra stress testing helped in identifying some of the weakness of prior versions of the algorithm that relied in closed contours to detect reference markers (even when closing step was included) and to move towards a less restrictive initial stage along with a O(n) algorithm (n = number of potential reference markers found on the mirrored L arrangement) to locate labels and giving more importance to the final pruning stage. It also helped in identifying the importance of a filtering step on the contents recognition stage in poor lightning conditions.

TABLE 1
PERFORMANCE OF ALGORITHM AS OVER 12 SOURCE IMAGES ROTATED OVER 36 DIFFERENT ANGLES AND 5 SCALE FACTORS

	Rotation / scaling algorithm			
	Bilinear interpolation		**Nearest neighbor interpolation**	
Scaling factor	**Bit Miss-detections**	**Label Miss-locations**	**Bit Miss-detections**	**Label Miss-locations**
1	0/68724	0/828	0/68724	0/828
0.9	3/68724	0/828	3/68724	0/828
0.8	0/68724	0/828	8/68724	0/828
0.7	10/68724	0/828	19/68724	0/828
0.6	1/68724	7/828	44/68724	11/828

V. SUMMARY AND FURTHER WORK

The proposed algorithm focuses on detecting the distinctive mirrored L shaped elements of the labels based on simple contour metrics. The key of this stage is that is conservative and robust to broken or irregularly shaped contours. As such it requires further pruning based on robust confidence measurements. Bit detection capabilities are improved by local adaptive histogram equalization and localized filtering around bit centers.

The algorithm showed zero error rate over the initial training set including full range rotations. It is robust against moderated reductions in resolution and under different lighting conditions, as the testing over an extended training set showed.

The main weakness found during stress testing was on images with excess of light as for some captured at short distance with a flash light, causing reflections and high contrast variations on the reference markers. Further work should address this issue by compensating for it or introducing more elements as part of the label location stage.

REFERENCES

[1] Problem statement. http://www.stanford.edu/class/ee368/project.html

[2] Michael Rohs, “Real-World interaction with Camera -Phones” 2nd International Symposium on Ubiquitous Computing Systems (UCS 2004), Tokyo, Japan, November 2004

[3] Canny, John, "A Computational Approach to Edge Detection," IEEE Transactions on Pattern Analysis and Machine Intelligence,Vol. PAMI-8, No. 6, 1986, pp. 679-698.

[4] Contrast-limited adaptive histogram equalization (CLAHE). http://www.mathworks.com/access/helpdesk/help/toolbox/images/adapthisteq.html

[5] http://www.vs.inf.ethz.ch/res/proj/visualcodes/