EE368 Project: Visual Code Marker Detection

Kahye Song
Group Number: 42
Email: kahye@stanford.edu

***Abstract*— A visual marker detection algorithm has been implemented and tested with twelve training images. The algorithm follows the general scheme suggested in [1] but the details have been modified to gain more robust estimation. The algorithm performs successfully for most of images but the result shows that it does not handle a marker which has been projected to an image plane with an angle deviated from the normal angle significantly.**

I. Introduction

The task of the project is to detect and read visual code markers through the cell-phone camera. Visual markers can be used as a hyperlink to a database which has the information of the object described by the marker. In the paper titled "Real-World Interaction with Camera-Phones", the author, Michael Rohs, describes the use of visual code markers in interacting with real-world objects.

In this project, a visual marker detection algorithm has been implemented based on [1] together with some modifications. The algorithm has been tested by a set of twelve training images and the corresponding ground truth data and the result has been evaluated using the provided evaluating function.

The visual code markers are 2-dimensional arrays. The array consists of 11x11 elements. Each element is either black or white. As shown in the figure below, we fix the elements in three of the corners to be black. One vertical guide bar (7 elements long) and one horizontal guide bar (5 elements long) are also included. The immediate neighbors of the corner elements and the guide bar elements are fixed to be white. This leaves us with 83 data elements which can be either black or white.

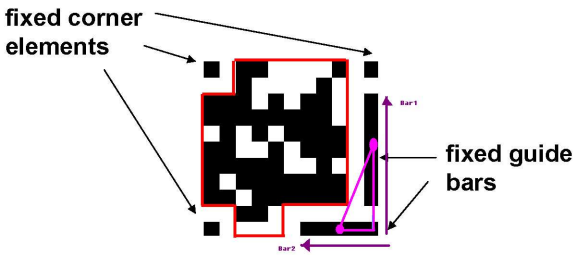

Fig. 1. Structure of visual code marker

Exploiting the structure of the marker, the algorithm tries to find the guide bar candidates and pair them based on the length ratio of the bars and the angle between the bars. When the visual marker locations are estimated, the corners are verified by checking whether the actual corresponding corners exist. Then all pixels are projected back to the code plain and decoded by counting the black pixels in the marker image corresponding each code coordinate.

II. Methodology

In this section, the whole procedure of the algorithm is described in the original sequence.

A. Gray scale and binary conversion

The 24bit RGB image is transformed into a gray-scale image by taking the average pixel values of the red and green channel at each position. The blue channel has been discarded due to poor resolution. Then this gray scale image is thresholded by the adaptive thresholding scheme which was given in [1]. The width of moving average s is set to $\frac{imagewidth}{16}$ to improve locality of the averaging and t as 20 to avoid blurred outline of features to be set to 1, which falsely connects regions.

Fig. 2. Gray-scaled training image 9

B. Region labeling and find high eccentricity regions

Each 1-bit valued(black) pixels are aggregated and labeled as a region if they are connected by 4-connectivity. Then each

Fig. 3. Binary version of training image 9

linear correlation of pixels in each region is computed and those with high absolute correlation are selected as guide bar candidates. For further testing, the direction/angle of the region and the length along this direction are estimated by finding the maximum and minimum x, y coordinate values of the region. This is because the bars that we are looking for can fit in a rectangle formed by max/min x and y coordinates as in Fig.

C. Test1: Find normal angle pairs

Since the guide bars form a right angle, we can find pairs of regions which are located in a normal direction to each other by using the region direction computed in II-B. Pick the pairs with large angle separation near 90 degrees. However, not to miss the original pairs with poor direction estimates, the pairs of angles between 55 and 90 degrees are saved at this stage.

D. Test2: Find pairs satisfying Pythagorean relations

To check whether each pair of bars are located close enough and to check if they can form a right triangle, the Pythagorean relations of each pair is tested as they do in Fig 1. among those selected in II-C. This is done by looking for the smallest $d = c^2 - a^2 - b^2$, where c is the side apposite to the right angle. The distance between the centers of the regions should not be too close below the decodable resolution limit(The smallest decodable marker can have 11 by 11 pixels.) and the three estimate side length should also meet the triangle criterion which is $c < a + b$.

E. Test3: Find the pairs with 5:7 length and area ratio

Among pairs selected in the first test, the ratio of length and area of regions are tested if they are close to 7:5. The pairs which have each ratio close to $\frac{7}{5}$ by 0.6 are selected. Then the common pairs selected by this test and 50 of those which have the smallest d values from the test in II-B.

F. Test4: Find the bottom right corner of the visual marker candidates

The pairs passed the tests up to Test3, they are considered to be guide bars. Bars in each pair are sorted as bar1 and bar2. Bar1 is the 7 bit length bar and Bar2 is the 5 bit one. Then we can find the bottom left corners of each marker candidate by simply finding the closest pair of ending points of bar1 and bar2 and the corresponding bar2 ending point should be the bottom right corner of the marker. At this stage, those pairs with too large separation of these ending points to compare with the length of bar1 and bar2 are removed since according to the structure, they should be 1 bit length away. Then the bar1 direction is set from this corner to upward and the direction of bar2 is from this corner to left. Refer to the figure below.

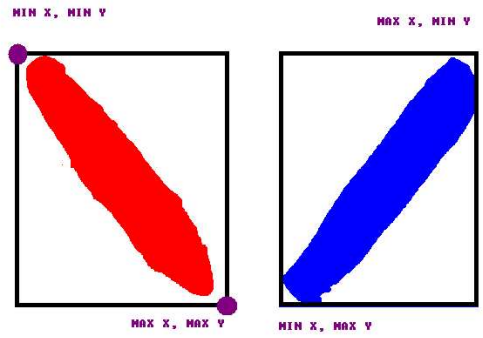

Fig. 4. Square fitting of each regions

G. Test5: Outer product test

The two bars should be located in an order to form a negative outer product in image domain, refer to Fig. 1. , which means if we rotate bar1 counterclockwise by 90 degrees it should overlap with bar2 not the other way around. Therefore those pairs which do not meet this criterion are removed.

H. Test6: Remove not very square like corners

Since we know that the whole mark size is 11 by 11, we can estimate preliminary positions of the other corners by adding bar1 and bar2 direction vectors with right scaling. Then we can check whether this polygon formed by corner estimates are actually approximately square by looking at the ratio of the sides. The angle has been already tested in II-C. Those pairs which do not meet this criterion are removed.

I. Test7: Remove marker candidates with off size

If multiple candidates are left, we also test whether all the candidates have approximately same size by comparing the each side size with the mean of the all sides. This is based on the assumption that those markers on a single image have the same size. If any of them is too large or small, it is removed.

J. Test8: Corner position verification

At this stage, the estimates of corner positions except the bottom-right corner are verified. We try to find a region with low eccentricity and approximate 1 pixel size with respect to the bar1 and bar2 around the estimated position. Find the one with closest to the estimates and confirm it as the actual corner if the distance between the estimate and the found position is not too large.

K. Projection to code domain and decoding

Once the corners are verified, the pixels within the polygon designated by the corners are projected to the code domain. Then by quantizing the projected location of each pixel, assign each pixel value to the corresponding code coordinate. Count all the black pixels of each code coordinate and compare the sum with the mean of all the values and set the decoded bit as one if the count is larger than the average count and zero if it is not.

III. Result on training images

The result on the twelve training images were quite successful. Each decoded result and the detected origin has been evaluated by 'evaluate.m'. The total score was 1801 and the total processing time of twelve images was 55.4544 seconds (all of them only in a minute!) on SCIEN machines. All the origin position of visual markers in each images are correctly detected and the number of correctly decoded bits are in the table below.

marker	1	2	3	4	5	6	7	8	9	10	11	12
1	83	80	49	83	83	83	83	82	82	59	83	77
2	-	83	83	-	59	-	82	-	83	79	-	82
3	-	-	83	-	74	-	-	-	83	83	-	-

TABLE I

Summery of correctly decoded bits of each marker in each training image

There were no false positives or repeats. Here are the images with detected markers and the detected fixed corner locations in red circles. The magnet squares are the estimated corner locations and red circles are corrected ones. The green crosses are the corners of marker window to extract only the marker from the whole binary image.

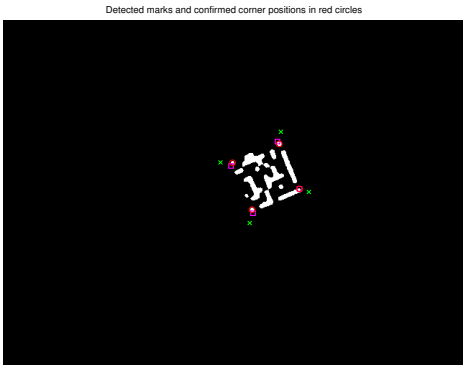

Fig. 5. Training image 1: Detected marks and the confirmed corners in red circles

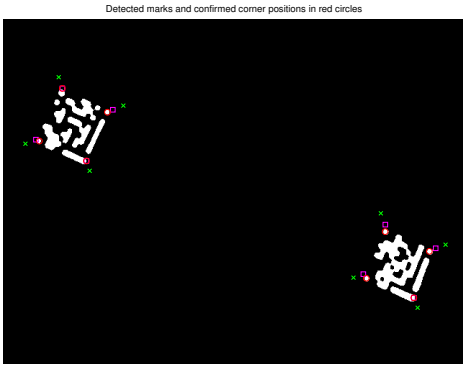

Fig. 6. Training image 2: Detected marks and the confirmed corners in red circles

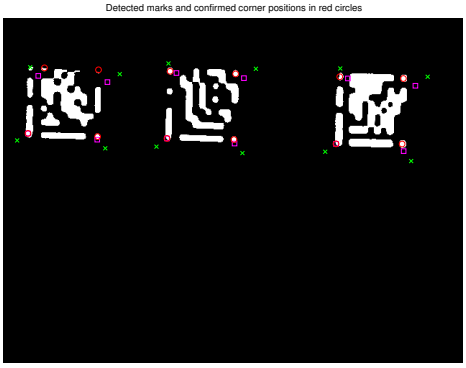

Fig. 7. Training image 3: Detected marks and the confirmed corners in red circles

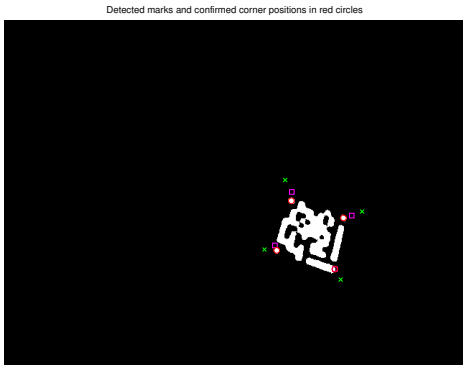

Fig. 8. Training image 4: Detected marks and the confirmed corners in red circles

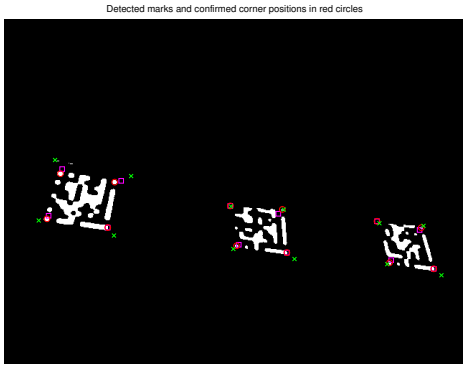

Fig. 9. Training image 5: Detected marks and the confirmed corners in red circles

Fig. 10. Training image 6: Detected marks and the confirmed corners in red circles

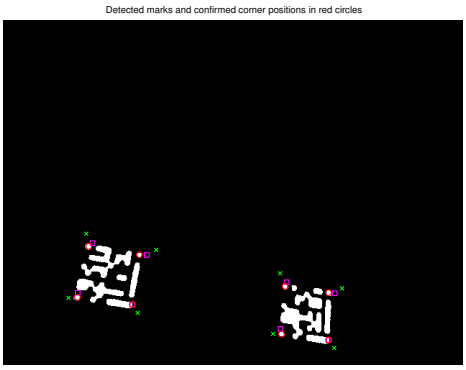

Fig. 11. Training image 7: Detected marks and the confirmed corners in red circles

Fig. 12. Training image 8: Detected marks and the confirmed corners in red circles

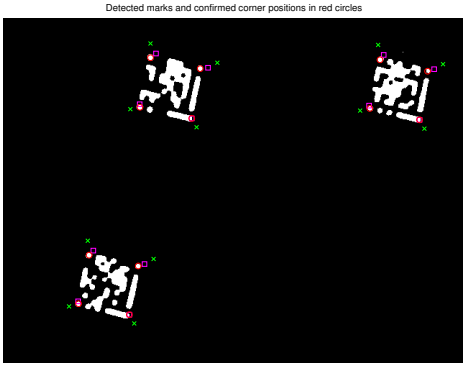

Fig. 13. Training image 9: Detected marks and the confirmed corners in red circles

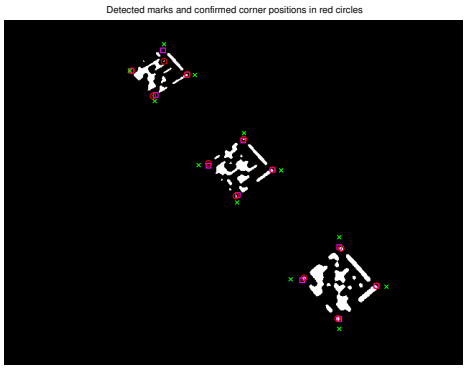

Fig. 14. Training image 10: Detected marks and the confirmed corners in red circles

Fig. 15. Training image 11: Detected marks and the confirmed corners in red circles

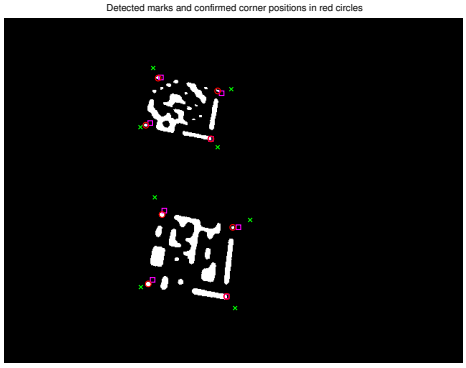

Fig. 16. Training image 12: Detected marks and the confirmed corners in red circles

IV. Discussion and conclusion

The algorithm works quite successfully with very short processing time for most of the image except 3,5 and 10. Minor misses are mostly caused by blur around bars which expands the actual size of black area and falsely decodes a single bits. However, serious misses are cause by the uneven projection of marker images. If the projection angle is tilted significantly, the estimates of the marker corners do not form an exact square and these can be misplaced slightly off. Then this results in wrong making of the visual mark and cause false adjustment of corner positions and false decoding. This is clearly seen in Fig 7.

To fix this problem, when the marker image projection should include the tilting effect when assigning each pixel to each code coordinate. This can be done in future to improve the detector performance.

V. Reference

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