# AN EFFECTIVE ALGORITHM FOR VISUAL CODE MARKER DETECTION

*Tao Xu, Chien-an Lai*

Department of Electrical Engineering, Stanford University

## ABSTRACT

This paper proposes an effective algorithm for visual code marker detection. It incorporates local binarization scheme and morphological image processing techniques with the pattern recognition method to achieve best performance. Experimental results have shown that the proposed detection algorithm is able to detect the visual code marker information quickly and accurately under different illuminance and camera direction conditions.

## 1. INTRODUCTION

With the integration of CCD cameras, mobile phones have become networked personal image capture devices. As image resolution improves and computing power increases, they can do more interesting things than just taking pictures and sending them as multi media messages over the mobile phone network. Michael Rohs presented a visual code system that turns camera-phones into mobile sensors for 2-dimensional visual codes [1]. In this novel application, how to detect visual code markers quickly and accurately becomes a crucial problem.

As a matter of fact, the extraction of visual code markers captured by cell-phone camera could be very difficult because of non-uniform illuminance conditions, different camera directions, different placement orientations of the code markers, and complicated background interference.

In this paper, we fully utilize the fixed pattern of various code markers and propose Local Binarization Scheme (LBS), Corner Detection for Generalized Quadrangle (CDGQ), and Parallelogram Regulation (PR) to solve the above problems successfully.

In the remainder of this paper, we systematically describe our algorithm in Section 2. Experimental results for total twelve training images are reported in Section 3, demonstrating the advantages of this proposed visual code marker detection algorithm. Appendix provides some methods which we tried but not adopted in the final detection algorithm.

## 2. THE PROPOSED DETECTION ALGORITHM

This section discusses the proposed marker detection algorithm in the image processing order.

### 2.1. Framework of Detection Algorithm

According to the image processing order, the proposed detection algorithm performs a series of operations upon the input RGB image, as shown in Figure 1.

The detection algorithm is primarily composed of eight modules: gray-level conversion, local binarization, image close and region removal operation, corner detection for generalized quadrangle, parallelogram regulation, sample interpolation, pattern matching, and read marker operation. We will discuss the key techniques in the following subsections.

### 2.2. Adaptive Local Binarization

As we know, the illumination is generally not exactly uniform in an arbitrary image. For example, in training image 3, the gray level of black region for the leftmost marker (i.e. bright area in the whole image) is about 50, and white region is about 170; However, the gray level of black region for the rightmost marker (i.e. dark area in the whole image) is about 25, and white region is about 100.

If we use a global threshold to the gray-level image, it would lead to neglect the bright area (small threshold) or introduce lots of noise in dark area (large threshold) as shown in Figure 2. (Note, here we set 1 for pixel values below the threshold; and 0 for pixel values above the threshold).

A smart scheme is the adaptive local threshold, i.e. use different thresholds for different areas in the whole image. In bright area, we use large threshold; in dark area, we use small threshold. By this way, we would like to retain the actual markers as clear as possible and remove the background noise as much as possible. Based on such consideration, we adaptively choose an appropriate threshold for every pixel which we call “Adaptive Local Binarization” (ALB).

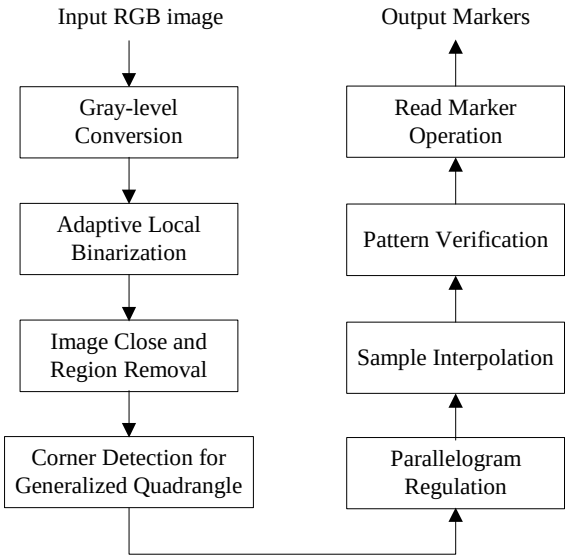
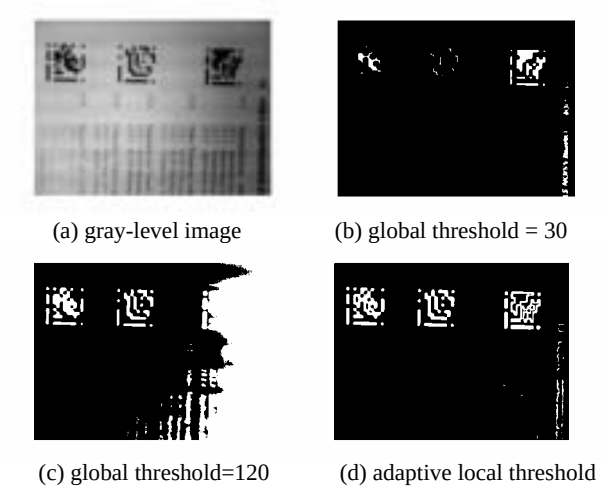



**Fig. 1.** Framework of the proposed detection algorithm

Adaptive local binarization is operated in the following way. Firstly, select an appropriate $L \times L$ window for each pixel. Secondly, compare the difference between maximum value M and minimum value m in this window. If this difference is larger than certain threshold N, we think there might be markers existing in this window because of the large pixel-value contrast (Note, for convenience, we call this type of window as type-1 window and type-2 for the other); otherwise, we think there are no markers existing in this window. Thirdly, for type-1 window, we set an adaptive threshold T to decide whether the pixel we consider is 0 or 1 in the obtained bi-level image; for type-2 window, we simply label the pixel value as 0 to remove the irrelevant background noise.

In our implementation, we propose L=11, N=60, T=M/1.5 since this set of parameters will achieve the best performance. From Figure 2, we can see the left two markers partially disappear in (b) when we use small global threshold; the rightmost marker is overwhelmed by large amount of background noise in (c) when we use large global threshold; however, by adaptive local threshold in (c), all markers are very clear especially all four corners of each marker which are essentially important for the following steps. Therefore, the adaptive local binarization outperforms the global binarization.

Here, we would like to emphasize again that whether all four corners of every marker could be shown correctly in bi-level image is crucial for the following steps since we will use these four corner coordinates to interpolate inner samples and further verify the pattern by a fixed template to decide true or false markers.



(a) gray-level image (b) global threshold = 30

(c) global threshold=120 (d) adaptive local threshold

**Fig. 2.** Binarization performance of global threshold and local threshold (training image 3)

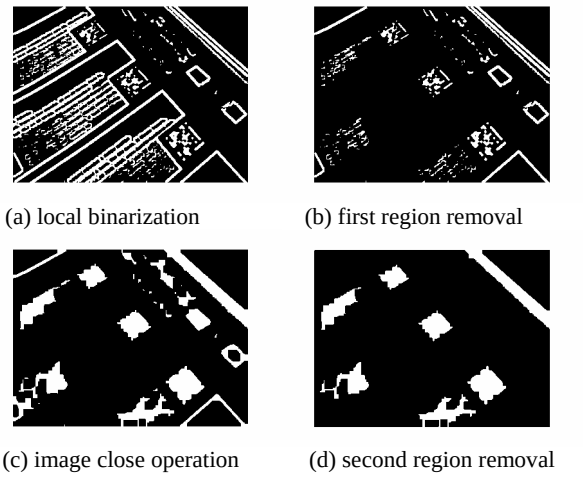
### 2.3. Image Close and Region Removal Operation

After we obtain the bi-level image, we need to perform morphological operation (image close) on it to get the marker "patch" for the following processes. The purpose of image close with region removal operation is to reduce background noise and further facilitate the next corner detection process.

Firstly, we perform the first region removal operation before image closes. This is based on an observation that, roughly speaking, the largest pixel number of any connected region in markers after adaptive local binarization cannot exceed the number 3500. So, we remove the large regions whose pixel numbers are above 3500. It is proven effective to reduce the number of background regions and retain the markers at the same time. For example, in training image 10, Figure 3 (a) shows the bi-level image obtained by adaptive local binarization and (b) shows the result after the first removal operation. We can easily see lots of large background regions are removed and the marker regions are retained correctly.

Secondly, we use image close technique to merge every marker into marker "patch" as shown in Figure 3 (c). Here, we would like to point out that the image close operation needs to make at least three marker corners (totally four) merged into one connected region for correctly extracting the positions of all marker corners in the following steps.

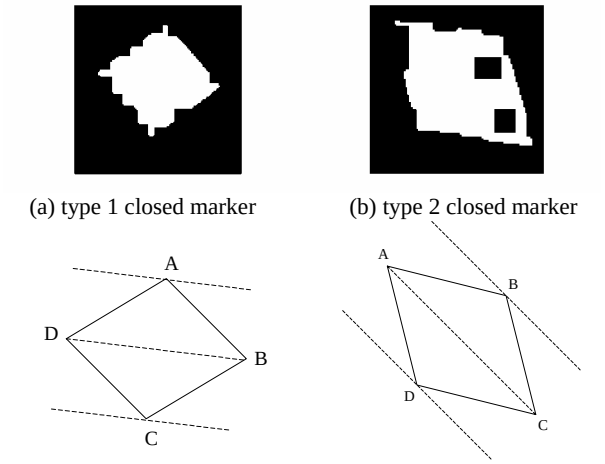
At last, according to another key observation that the pixel number of merged marker region generally ranges from 2400 to 11000. Therefore, we can perform the second region removal operation to remove the background



(a) local binarization (b) first region removal

(c) image close operation (d) second region removal

**Fig. 3.** Illustration of image close and region removal operation (training image 5)




(a) type 1 closed marker (b) type 2 closed marker

(c) type 1 topological structure (d) type 2 topological structure

**Fig. 4.** Two types of closed markers and their topological structures

"patches" whose pixel number are outside the above range. Figure 3 (d) shows the result of the second removal operation.

### 2.4. Corner Detection for Generalized Quadrangle

As we said, accurate corner detection is crucial for the whole performance. We propose Corner Detection for Generalized Quadrangle (CDGQ) to solve this problem.

Roughly speaking, there are two types of closed marker "patch" which we call type 1 and 2 closed markers. Figure 4 (a) and (b) show the typical structures of two types of closed markers. If we connect all four corners of these marker patches by straight lines, we will obtain an approximate parallelogram (i.e. its topological structure) which are shown in Figure 4 (c) and (d).

Now, we would like to point out that the concept of "quadrangle" here is a generalized concept. From Figure 4 (a) and (b), we see such "quadrangle" has jagged edges and might have holes inside. So, common corner detection algorithm such as Haralick corner detector doesn't work here because of these jagged edges and holes inside code markers.

The key difference of these two types of closed markers (parallelograms) is that each of four corners of type-1 closed marker occupies each of four extreme coordinates, i.e. minimal x coordinate (point A), maximal x coordinate (point C), minimal y coordinate (point D), maximal y coordinate (point B); however, for type-2 closed marker, point A occupies both minimal x coordinate and minimal y coordinate, point C occupies both maximal x coordinate and maximal y coordinate. Therefore, if we only use minimal or maximal coordinates to determine four corners, we cannot handle type-2 situation although it can solve type-1 situation very well (for convenience, we call it as extreme-value method).

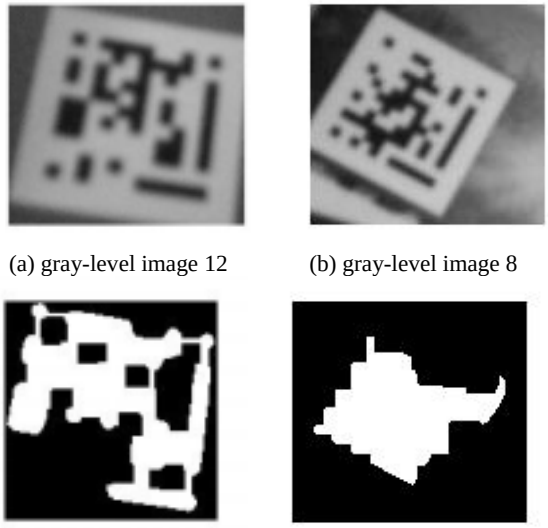
We propose Corner Detection for Generalized Quadrangle (CDGQ) to deal with both closed marker structures. This novel corner detection algorithm finds four corners in the following way.

Firstly, by extreme-value method discussed above, we get four points A, B, C, and D. Note, these four points might overlap to become actual 3 or 2 points but we still denote them as four points.

Secondly, compute the distance between every two-point pair (i.e. 6 distances). Here, we would like to emphasize again that the distance might be zero if these two points overlap.

Thirdly, pick up the pair of points which corresponds to the largest distance. For example, (B,D) in type-1 closed marker structure and (A,C) in type-2 closed marker structure as shown in Figure 4 (c) and (d).

At last, connect the chosen point pair (dashed line in Figure 4) and parallelly shift the dashed line till the border of the closed marker region to obtain two other "tangent" points , i.e. (A,C) in type-1 situation and (B,D) in type-2 situation as shown in Figure 4 (c) and (d). Until now, we get all four corners for both situations.



(a) gray-level image 12 (b) gray-level image 8

(c) Closed image 12 (d) Closed image 8

**Fig. 5.** Illustration of situations that "Parallelogram Regulation" is needed.

### 2.5. Parallelogram Regulation

We can correctly find all four corners of each marker by CDGQ if it is connected into a single region in the image close operation. However, due to the sparseness of some marker patterns (i.e., the number of white elements is much larger than that of black elements, as shown in Figure 5 (a) and (c)) or the merging of surrounding background noise (as shown in Figure 5 (b) and (d)), sometimes we cannot close the markers successfully. In this situation, actual four corners are not the corners found in the above CDGQ algorithm.

In order to estimate the correct corners in a given region, we firstly need to make some assumptions. (1) There is at most 1 wrong corner in each connected marker. 2) The shape of connected marker region is roughly a parallelogram. For the first assumption, since the fixed part of each marker consists of two black bars along its edges, it is easier to connect the corresponding three corners (i.e. up-right, down-right, and down-left in standard marker pattern). As we said, the appropriate parameters for image close are able to guarantee this assumption in very most cases. For the second assumption, if the picture is taken from different camera directions, the shape of the marker should be roughly a parallelogram. Since it is always possible to interpolate the missing point using any other three corners, the parallelogram regulation will not be unique if we don't make some further assumptions. So we assume that the picture is roughly

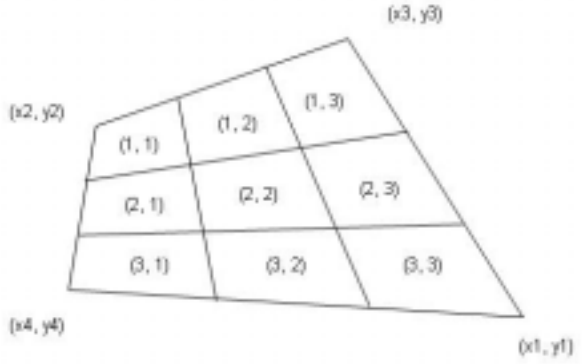



**Fig. 6.** Illustrate of sample interpolation

taken from the top of the image so that this parallelogram is close to a square.

As we know, each angle of a square is 90 degrees and the length difference of any two adjacent edges is 0. By these two properties, we set two criteria to determine whether we need to use Parallelogram Regulation **(**PR) and which point needs to be regulated. One criterion L is the ratio of length difference between maximum and minimum edges to the average of all four edges. The other A is the difference between maximum and minimum angles. In practical implementation, we set the threshold for L as 0.3 and the threshold for A as 0.5 radian. If L is larger than 0.3 or A is larger than 0.5 radian, we perform PR.

Once we decide to perform PR, we need to decide which point we should fix. We adopt the following method. Firstly, denote P as the diagonal point of the possibly wrong point. Secondly, check the adjacent edge lengths and angle of point P because these two parameters will not be affected by the diagonal wrong point. Therefore, we also set two criteria like above situation. One criterion L' is the ratio of adjacent edge length difference of point P to the average of all four edges. The other A' is the difference between the angle of point P and right angle (90 degrees). In practical implementation, we set the threshold for L' as 0.2 and the threshold for A' as 15 degrees. If both L' and A' are smaller than their thresholds, we claim that the diagonal point of point P is wrong and estimate it using PR.

### 2.6. Sample Interpolation

After we find the four vertexes of a marker, we need to interpolate inner samples, i.e., find the corresponding position for each element. Since the picture is taken from different camera direction and distance, the length of a marker on that image is changing linearly. Therefore, the best way to sample each element is to linearly interpolate the position of the inner sampling points. We derive the

formula for computing the position of each inner sampling point (i, j) which can be written as

$$x(i,j) = [x_2(11.5-i)(11.5-j) + x_1(i-0.5)(j-0.5) + x_3(11.5-i)(j-0.5) + x_4(i-0.5)(11.5-j)]/121$$
$$y(i,j) = [y_2(11.5-i)(11.5-j) + y_1(i-0.5)(j-0.5) + y_3(11.5-i)(j-0.5) + y_4(i-0.5)(11.5-j)]/121$$

where $(x_1, y_1) \ldots\ldots (x_4, y_4)$ are the coordinates of all four corners of this marker.

Once we obtain the sampling point, we need to judge whether this element is black or white. We cannot use a global threshold to judge since the illumination is different for each marker. So, like the adaptive local binarization, we can use the local threshold for each marker by averaging the value of gray pixels inside a marker.

### 2.7. Pattern Verification

In this step, we need to find the true marker(s) by utilizing the fixed pattern of a standard marker. Firstly, since the marker may be rotated, we have to find which corner is the upper left element. When we obtain all four corners in CDGQ discussed above as shown in Figure 4 (c) and (d), we actually lose the corner sequence (clockwise or counter clockwise). Therefore, besides the rotation of the obtained 11 by 11 matrix, we also need to consider the upside-down situation, i.e. transpose the 11 by 11 matrix to fully check the possibility of existing markers. After matching all these 8 situations with the standard template, we will pick the one which matches best and if the best matched value is above threshold 25 (Note, the matched value ranges from -38 to 38), we will claim this region is a true marker and record the pattern and origin of the corresponding matrix.

## 3. EXPERIMENTAL RESULTS

**Tabel. 1.** Performance of the proposed detection algorithm

| | | | |
|---|---|---|---|
| Training image 1 | | | |
| | Marker 1 | | |
| Correct bits | 83 | | |
| Execution time (second) | | 9.0587 | |
| Training image 2 | | | |
| | Marker 1 | Marker 2 | |
| Correct bits | 83 | 83 | |
| Execution time (second) | | 11.6157 | |
| Training image 3 | | | |
| | Marker 1 | Marker 2 | Marker 3 |
| Correct bits | 83 | 63 | 83 |
| Execution time (second) | | 11.4069 | |
| Training image 4 | | | |
| | Marker 1 | | |
| Correct bits | 83 | | |
| Execution time (second) | | 10.6035 | |
| Training image 5 | | | |
| | Marker 1 | Marker 2 | Marker 3 |
| Correct bits | 83 | 82 | 83 |
| Execution time (second) | | 14.2162 | |
| Training image 6 | | | |
| | Marker 1 | | |
| Correct bits | 82 | | |
| Execution time (second) | | 10.8251 | |
| Training image 7 | | | |
| | Marker 1 | Marker 2 | |
| Correct bits | 83 | 83 | |
| Execution time (second) | | 11.5988 | |
| Training image 8 | | | |
| | Marker 1 | | |
| Correct bits | 83 | | |
| Execution time (second) | | 12.4464 | |
| Training image 9 | | | |
| | Marker 1 | Marker 2 | Marker 3 |
| Correct bits | 83 | 83 | 83 |
| Execution time (second) | | 13.1669 | |
| Training image 10 | | | |
| | Marker 1 | Marker 2 | Marker 3 |
| Correct bits | 81 | 83 | 83 |
| Execution time (second) | | 13.7535 | |
| Training image 11 | | | |
| | Marker 1 | | |
| Correct bits | 83 | | |
| Execution time (second) | | 10.4617 | |
| Training image 12 | | | |
| | Marker 1 | Marker 2 | |
| Correct bits | 83 | 73 | |
| Execution time (second) | | 12.5940 | |
| Total Score | | 1875 | |
| Total Time (second) | | 141.7474 | |

From Table 1, we can find that the execution time for each image is less than 15 seconds and the marker pattern can be detected correctly most of the time. There are mainly 2 non-satisfactory markers for our algorithm.

The first one is the second marker in training image 3. This marker is sparse (in black) at the left side; therefore, the upper left corner is not detected. Instead, a black element around the corner is detected as a corner by CDGQ. Unfortunately, because the point is so close to the real corner, PR conditional is not satisfied. Therefore 20 errors are occurred by the imprecision of the corner position.

The second one is the marker 2 in training image 12. The shape of this marker is trapezoid and PR is applied on it. Therefore the position of estimated corner is not correct, which causes several errors.

## 4. CONCLUSIONS

In this paper, we propose Adaptive Local Binarization (ALB) scheme, Corner Detection for Generalized Quadrangle (CDGQ), Parallelogram Regulation (PR) for marker detection. With the help of these novel methods, we can achieve the best performance. Experimental results have shown that the proposed detection algorithm is quick and accurate regardless of different illuminance and camera direction conditions.

## APPENDIX

As a starting point for this project, we try to use edge detection to get the rotation angle for each marker. However, the length of the fixed pattern inside a marker is much too weak comparing to the other background edges. Therefore we decide to use binarization scheme and morphological image processing to filter out several backgrounds. After realization this pre-processing, there may still exist strong background edges. We also find that the most difficult part is to decide the size and shape or each marker. That is the main reason we decide to use morphological characteristics rather than edge detection.

As for the first step in CDGQ, by extreme-value method, we get four points A, B, C, and D. Actually, A, B, C, and D may not be unique, especially when the angle of rotation is 0, 90, 180, and 270 degrees. Instead of using the median point, we figure out that each corner should be shifted to one side depending on the slope of that point.

## 5. BREAKDOWN

Tao Xu:
(1) Adaptive Local Binarization (ALB)
(2) Image close and region removal
(3) Co-work on Corner Detection for Generalized Quadrangle (CDGQ) and Pattern Verification

Chien-an Lai:
(1) Parallelogram Regulation (PR)
(2) Sample Interpolation and Pattern Verification
(3) Co-work on Corner Detection for Generalized Quadrangle (CDGQ)

## 5. REFERENCES

[1] Michael Rohs, “Real-World Interaction with Camera Phones,”
[2] Bernd Girod, Lecture notes from EE368 Digital Image Processing