EE368 Class Project, Spring 2005-2006

Visual Code Marker Detection

Chien-Yu Chen
chienyu@stanford.edu

I. Introduction

In this paper, we present a method to detect special visual code markers. Fig.1 shows an example of the visual code markers. The visual code markers we are considering have two fixed guide bars and three fixed cornerstones. Our method uses these elements to locate the visual code markers and then read the codes on markers.

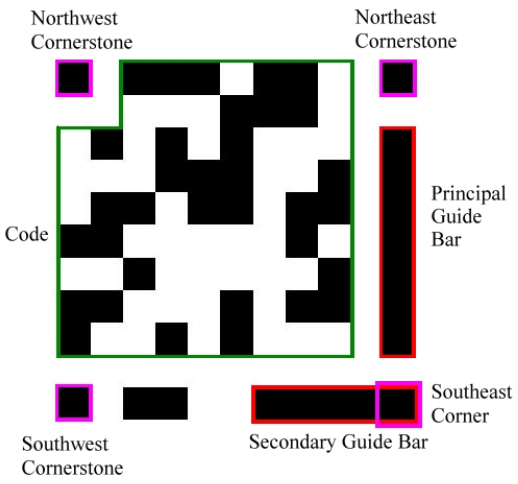


Fig 1. An example of a visual code marker.

Our method is based on the method proposed in [1]. It firstly looks for the principal guide bar candidates. For each candidate, it looks for its correspondent secondary guide bar. Next, for each pair of guide bars, it looks for its three correspondent cornerstones. After a pair of guide bars and three cornerstones are located, we calculate the transformation between the visual code marker coordinate and the image coordinate and find the mapped position on the image plane for each position on the marker plane.

Since the images are captured by cell phones, preprocessing is necessary for performing the detection well. Our method employs an adaptive thresholding proposed in [2] to binaries the original image. We made some modifications to achieve better performance on this specific application.

II. Preprocessing

The original images are color images captured by cell phones but the input images for marker detection in our method are black-white images. Firstly, an original color image is converted to a grayscale image by averaging the green and the red components of the original image because green and red components dominate the brightness more than blue component. After having a grayscale image, we employ an adaptive thresholding [1] to get the binarized image.

Given a grayscale image, we compute the moving average $g(n)$ of the grayscale values at each position (x,y) where n is the sequence number of a pixel that depends on the direction we go along. When we go from left to right and from top to bottom, we have

$$n = (x-1)\cdot image\_width + y \qquad \text{for pixel } (x,y)$$

When we go alternately from right to left and from left to right, we have

$$\begin{aligned} n &= (x-1)\cdot image\_width + y && \text{while } x \bmod 2 = 0 \\ n &= x\cdot image\_width - y + 1 && \text{otherwise} \end{aligned}$$

The moving average $g(n)$ is computed as

$$g(n) = g(n-1)\cdot\left(1-\frac{1}{s}\right) + im(x,y)$$
$$g(1) = 0.5\cdot(s-1)$$

$im(x,y)$ is the grayscale value of pixel (x,y), n is the sequence number of pixel (x,y) and s is the width of region to be considered having similar lighting conditions. The adaptive thresholding usually has better results when s is 1/8 of the image width.

To binarize the grayscale image in alternate directions, we need to take the moving average of previous line into consider to reduce the “every-other-line effect” mentioned in [2].

Therefore, when n is in alternate directions,

$$h(x,y) = \frac{g(n) + g\left(n - \frac{image\_width}{2}\right)}{2}$$

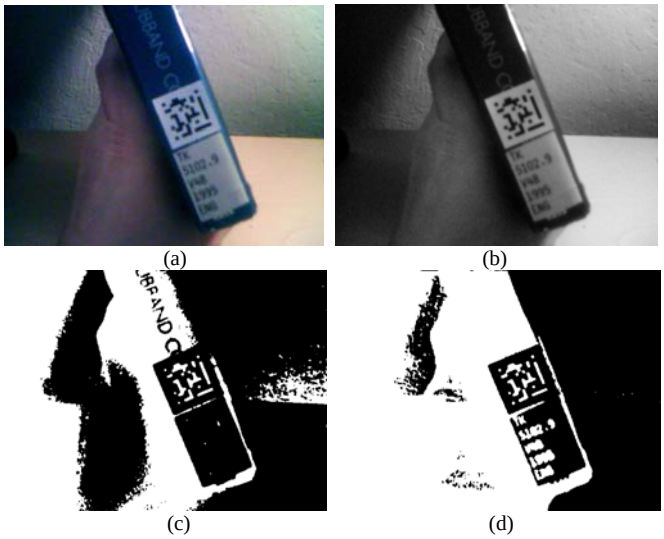


Fig 2. Results of adaptive thresholding.
(a) original color image. (b) grayscale image. (c) resulting image with adaptive thresholding in left-to-right direction. (d) resulting image with adaptive thresholding in alternate directions. Notice that the text below the marker can only be seen in (d).

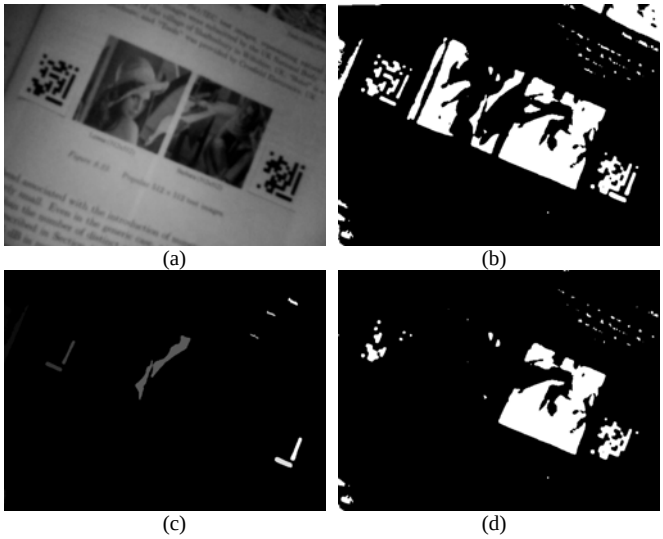


Fig. 3. Bar candidates and cornerstone candidates.
(a) grayscale image. (b) binarized image. (c) bar candidates. (d) cornerstone candidates.

and when *n* is in left-to-right direction,

$$h(x, y) = g(n)$$

Finally, we binarize the image as

$$bw(x, y) = 1 \quad \text{if} \quad im(x,y) < h(x,y) \cdot p/s$$
$$bw(x, y) = 0 \quad otherwise$$

p is a parameter that determines the percentage of values larger than the moving average that is classified as 1. Notice that our goal is to convert the black marker elements into white objects.

Fig. 2 shows the results of one-direction adaptive thresholding and alternate-direction adaptive thresholding.

Although [1] mentioned alternate direction reduced the "every-other-line" effect, we can still see clear artifacts. Therefore, after alternate-direction adaptive thresholding, we apply a closing with a kernel *[1;1]* to the resulting image to get rid of the effect.

III. VISUAL CODE MARKER LOCATING

Locating Pairs of Guide Bars

We label all the regions on the preprocessed image and use *regionprops* function to calculate the area, the orientation of the major axis, the length of major axis, the length of minor axis, and the centroid of each region. A region with the ratio of the length of major axis and the length of minor axis larger than a number and smaller than another is considered a bar candidate. In our method, a bar candidate has that ratio between 3 and 12, is at least 30-pixel large, and its size is larger than 0.6 times its major axis length times its minor axis length. Fig.3 shows the results of bar candidate detection.

For each bar candidate, we estimate the position of the secondary bar. Illustrated in Fig. 4, for a bar candidate *i*, there are two positions where the secondary bar may be.

1)
$$est\_x = Centroid(i).x + est\_width \times 5.0 \times \sin(Orientation(i))$$
$$est\_y = Centroid(i).y - est\_width \times 5.0 \times \cos(Orientation(i))$$

2)
$$est\_x = Centroid(i).x - est\_width \times 5.0 \times \sin(Orientation(i))$$
$$est\_y = Centroid(i).y + est\_width \times 5.0 \times \cos(Orientation(i))$$

The est_width is the estimation of the width of a single grid of the marker on image plane. The width is estimated by the weighted average of the length of the major axis and the length of the minor axis.

$$est\_width = \frac{MajorAxisLength(i)/7 + MinorAxisLength(i)/5}{2}$$

The angel between the principal bar and the secondary bar is allowed to be within some range because of perspective projection. Also, the length of the secondary bar is ideally 5/7 of the length of the principal bar. Thus, we then verify if there is another bar candidate near the estimated location and it must satisfy the following criteria:

1. The angel between the principal bar and the secondary bar is between $\pi/3$ and $2\pi/3$.

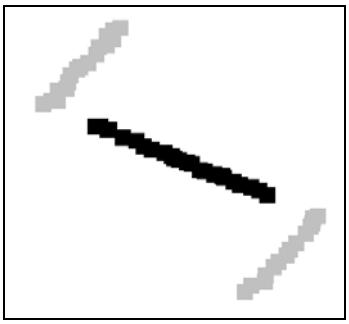


Fig 4. Two possible positions of a principal guide bar.
The black bar is the principal guide bar we found and two grays bars indicate two possible positions of the secondary guide bar.

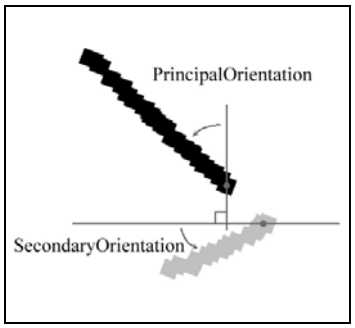


Fig 5. PrincipalOrientation and SecondaryOrientation.
Notice that *PrincipalOrientation* and *SecondaryOrientation* may not be same due to perspective projection.

2. $abs(MajorAxisLength(secondary\_bar) - MajorAxisLength(principal\_bar) \times 5/7) < 1.1 \times MajorAxisLength(principal\_bar)$

After locating a pair of a principal bar and a secondary bar, we calculate the actual orientations of the principal bar and the secondary bar, *PrincipalOrientation* and *SecondaryOrientation* which range between 0 and 2π as in Fig. 5. *PrincipalOrientation* is the angel that the principal bar is rotated around the bar's down-most pixel on marker. *SecondaryOrientation* is the angle that the secondary bar is rotated around the bar's right-most pixel on marker. We also add the pair of bars into a set of validated pairs of bars.

Computing Southeast Corners

The southeast corner is the projected position of (11,11) in the marker coordinate on the image plane. It is the intersection of the principal bar *i* and the secondary bar *j* and the position *(x,y)* is computed as

$$x_1 = Centroid\ (i).x$$
$$y_1 = Centroid\ (i).y$$
$$x_2 = Centroid\ (j).x$$
$$y_2 = Centroid\ (j).y$$
$$\alpha = PrincipalOrientation\ (i)$$
$$\beta = SecondaryOrientation\ (j)$$
$$y = \frac{x_2 - x_1 + y_2 \times \tan\beta + y_1 \cot\alpha}{\tan\beta + \cot\alpha}$$

If $\tan\beta < 100$

$$x = -y \times \tan\beta + x_2 + \tan\beta \times y_2$$

else

$$x = y \times \cot\alpha + x_1 - \cot\alpha \times y_1$$

It has different ways to compute *x* to avoid computation error when $\tan\beta$ is close to infinity.

Locating Northeast Cornerstones

We choose a region to be a cornerstone candidate if its ratio of major axis and minor axis lengths is smaller than a value. Because of low resolution and focus problem of cell phones, some small cornerstones have small minor axis length so the major axis length would be relatively large. Therefore, we use several levels of rules to compromise this problem. We say a region *i* is a cornerstone candidate if

$AxisRatio(i) \le 5$ when $MinorAxisLength(i) \le 2$
$AxisRatio(i) \le 3$ when $MinorAxisLength(i) \le 5$
$AxisRatio(i) \le 2$ when $MinorAxisLength(i) \le 15$
$AxisRatio(i) \le 1.5$ otherwise

Fig. 3 illustrates the results of cornerstone candidate detection. For each pair of a principal bar, *i*, and a secondary bar, *j*, we estimate the position of northeast cornerstone as

$$est\_x = Centriod\ (i).x - est\_width \times 5 \times \cos(PrincipalOrientation(i))$$
$$est\_y = Centriod\ (i).y - est\_width \times 5 \times \sin(PrincipalOrientation(i))$$
$$est\_width = MajorAxisLength(i)/7$$

Since the estimation is based on the principal bar, the *est_width* used here should also be estimated based on the length of the principal bar.

If there is a cornerstone near the estimated position, we verify that its size is about the size of one grid of the marker on image plane. The size of one grid of the marker on image plane is estimated by the weighted average of the principal bar area, *area(i)*, and the secondary bar, *area(j)*.

$$est\_grid\_area = \frac{area(i)/7 + area(j)/5}{2}$$

The cornerstone we found is valid for the pair of bars if its area is smaller than twice *est_grid_area*. If we cannot find a

valid cornerstone for a pair of bars, we discard the pair from the validated pairs of bars.

Locating Southwest Cornerstones

The position of the southwest cornerstone of a pair of a principal bar *i* and a secondary bar *j* is estimated as

$$est\_x = Centroid(j).x + est\_width \times 8 \times \sin(SecondaryOrientation(j))$$
$$est\_y = Centroid(j).y - est\_width \times 8 \times \cos(SecondaryOrientation(j))$$
$$est\_width = MajorAxisLength(j) / 5.0$$

Since the estimation is based on the secondary bar, the *est_width* used here is also estimated based on the length of the secondary bar. Also, a valid southwest cornerstone should be similar to the size of the estimated grid area and the size of the northeast cornerstone we found. Therefore, with the same *est_grid_area* in locating northeast cornerstones, a valid southwest cornerstone is smaller than twice *est_grid_area* and bigger than 0.2 × NE_cornerstone_ area.

Locating Northwest CornerStones

The position of the northwest corner is the most difficult one to estimate. To simplify the estimation, we assume that the angel between the image plane and the marker plane is not large so that the projected marker on the image plane is approximately a parallelogram. Therefore, we can estimate the position of the northwest cornerstone *(est_x,est_y)* with the position of the southeast corner, (x_1,y_1), the northeast cornerstone, (x_2,y_2), and the southwest cornerstone (x_3,y_3).

$$est\_x = x_2 + x_3 - x_1$$
$$est\_y = y_2 + y_3 - y_1$$

The size of a valid northwest cornerstone should also be about 1 and about the same as the sizes of the northeast and the southwest cornerstones. Therefore, given a pair of a principal bar *i* and a secondary bar *j*, a valid northeast cornerstone c_{ne}, and a valid southwest cornerstone c_{sw}, a cornerstone candidate *c* is a valid northwest cornerstone for the marker if it is near the estimated position and it satisfies all of the following criteria:

1. The area of *c* is smaller than 1.5 $\cdot\pi\cdot(dist(c,c_{ne})/20)^2$
2. The area of *c* is smaller than 1.5 $\cdot\pi\cdot(dist(c,c_{sw})/20)^2$
3. The area of *c* is smaller than 4 times the area of c_{ne} and 4 times the area of c_{sw}
4. The area of *c* is larger than 0.25 time the area of c_{ne} and 0.25 times the area of c_{sw}

   where $dist(c_1,c_2)$ is the distance between the centroids of c_1 and c_2.

Criteria 1 and 2 are used to eliminate the cornerstone candidates that are too close to the northeast cornerstone or the southwest cornerstone. The distance between the northwest cornerstone should be large enough to fit in 10 grids of the marker. Criteria 3 and 4 are used to eliminate the cornerstone candidates whose sizes are not similar to the sizes of northeast and southwest cornerstones that we have already found.

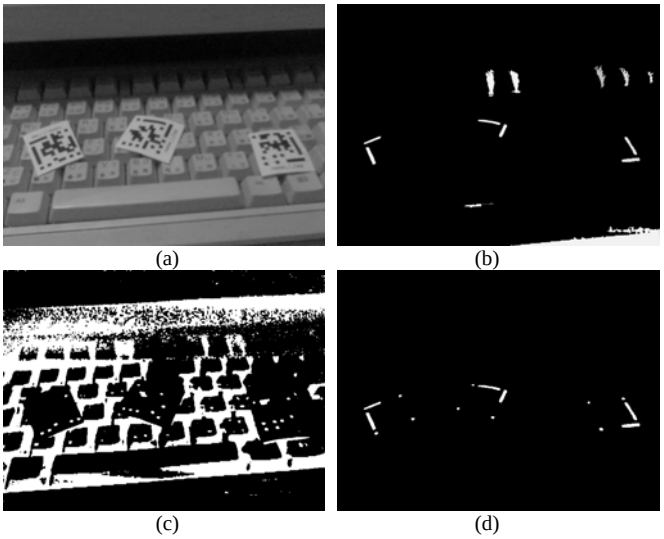


Fig. 6. Result of marker detection.
(a) original image. (b) bar candidates. (c) cornerstone candidates. (d) three detected markers.

After locating a correspondent set of a principal guide bar, a secondary guide bar, a northeast cornerstone, a southwest cornerstone, and a northwest cornerstone, we have a found valid marker. Next, our task is to read the code on the marker. Figure 6 shows the results of marker detection.

IV. CODE READING

The transformation from the marker coordinate to the image coordinate is a perspective projection[2]. Given the position *(u,v)* of a point *p* on the marker coordinate, the position *(x,y)* of the projected point *p'* on the image coordinate, and the transformation **A** from the marker coordinate to the image coordinate, we have the relationship $p' = p\cdot\mathbf{A}$. Thus,

$$(x', y', w) = (u', v', q)\begin{pmatrix} a & d & g \\ b & e & h \\ c & f & i \end{pmatrix}$$
$$(x, y) = (x'/w, y'/w)$$
$$(u, v) = (u'/q, v'/q)$$

We convert two-dimensional Cartesian coordinate (x,y) into three-dimensional homogeneous coordinate for perspective projection[3]. A perspective projection has eight degree of freedom so we can solve the perspective projection by letting $i = 1$ and solving the system[2]:

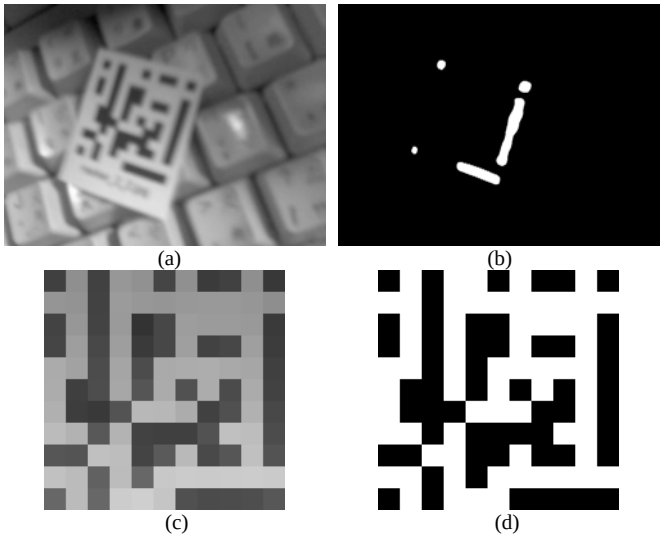


Fig. 7. Code reading
(a) original image. (b) detected marker. (c) raw marker. (d) binarized marker

$$\begin{pmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1x_1 & -v_1x_1 \\ u_2 & v_2 & 1 & 0 & 0 & 0 & -u_2x_2 & -v_2x_2 \\ u_3 & v_3 & 1 & 0 & 0 & 0 & -u_3x_3 & -v_3x_3 \\ u_4 & v_4 & 1 & 0 & 0 & 0 & -u_4x_4 & -v_4x_4 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -u_1y_1 & -v_1y_1 \\ 0 & 0 & 0 & u_2 & v_2 & 1 & -u_2y_2 & -v_2y_2 \\ 0 & 0 & 0 & u_3 & v_3 & 1 & -u_3y_3 & -v_3y_3 \\ 0 & 0 & 0 & u_4 & v_4 & 1 & -u_4y_4 & -v_4y_4 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$$

To solve A for the marker, we let $(u_1,v_2) = (11,11),(u_2,v_2) = (1,11)$, $(u_3,v_3) = (11,1)$, and $(u_4,v_4) = (1,1)$. The correspondent (x_1,y_1), (x_2,y_2), (x_3,y_3), and (x_4,y_4) are the position of the southeast corner and the centroids of the northeast, the southwest, and the northwest cornerstones respectively.

After solving the perspective projection transformation **A**, we calculate the projected positions of all the positions on the marker coordinate and read the values of those pixels on grayscale image. Thus, we have an 11×11 grayscale image representing the marker and then binary the grayscale image with MATLAB's *im2bw* and *graythresh* functions. Finally, we can extract the 81-bit information from the binarized image. Fig. 7 shows the result of reading a marker code.

V. Results Combining

We discussed that the adaptive thresholding had two different directions and a parameter *p* to decide the resulting binarized image. The images captured by the cell phones are usually blurred. Depending on the blur and the background, we need different *p* to perform adaptive thresholding well. As in Fig. 4, when *p* is too small, the adaptive thresholding may reject too much black pixels and result in that no marker is detected. On the other hand, when *p* is too large, the adaptive thresholding may accept too much black pixels and resulting in that the bars are connected to their neighbor elements.

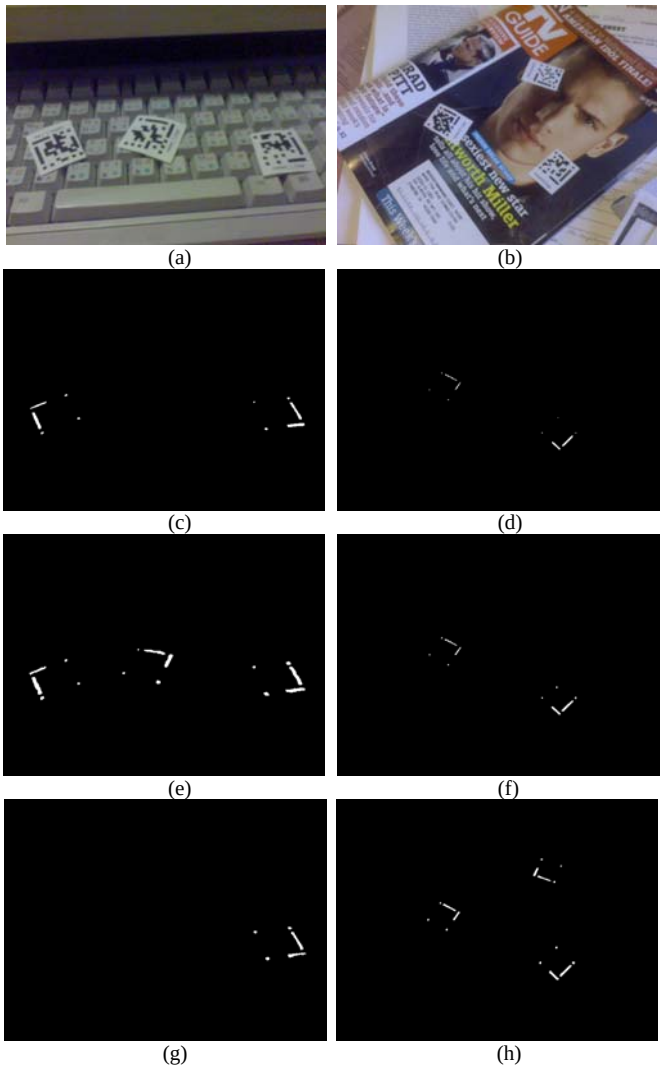


Fig. 8. Combining the results of different parameters.
(a)(b) two original images. (c)(d) markers detected with adaptive thresholding in alternate directions and p = 0.8. (e)(f) markers detected with adaptive thresholding in alternate direction and p = 0.9. (g)(h) markers detected with adaptive thresholding in left-to-right direction and p = 0.9.

Therefore, we need to find a good *p* for each marker. Our solution is to run the detection in alternate directions with different *p*'s and then combine all the results.

Moreover, when the marker is small, the alternate-direction adaptive thresholding does not work well since the parameter s which is 1/8 of the image width is relatively too large. Thus, to detect those markers, we run another detection with left-to-right direction and s = 1/128 of the image width. Fig. 8 shows the results with different parameters.

VI. Results

After experimenting on different combinations of

TABLE I
FOUR SET OF PARAMETERS USED IN OUR METHOD

p	*S*	Direction
0.6	1/8 of image width	Alternate directions
0.75	1/8 of image width	Alternate directions
0.9	1/8 of image width	Alternate directions
1.1	1/8 of image width	Alternate directions
0.9	1/128 of image width	Left-to-right direction

TABLE II
RESULTS OF TRAINING IMAGES

Image	Score	Execution Time*	Correct Bits	#False Alarms	#Repeats
1	83	4.89	83	0	0
2	166	4.20	166	0	0
3	249	4.14	249	0	0
4	83	4.50	83	0	0
5	249	4.69	249	0	0
6	83	3.97	83	0	0
7	166	4.58	166	0	0
8	83	4.14	83	0	0
9	249	4.89	249	0	0
10	249	5.14	249	0	0
11	83	4.23	83	0	0
12	166	5.12	166	0	0

*run with MATLAB 7.0 on a PC with Pentium4 2.6GHz and 768MB RAM

parameters on test images, we found five sets of parameters that should work well with the training images and test images we generated. Our method uses these five different sets of parameters and gets five detection results. Then, we eliminate repeated detections from the five detection results and get a final detection. The five different sets of parameters we use are listed in Table I. As we discussed in Section II, a large *p* accept more dark pixels and a small *p* reject more dark pixels. We choose a large p for the markers where the contrast is low and choose a small p for the markers where the contrast is high. Parameter s works well in most cases when s is 1/8 of the image width. However, it may work poorly when the marker is too small so the background can affect the thresholding. To detect those markers, we need a smaller s and process in left-to-right direction.

Table II shows the results of running our method on the training images. Our method performs well in detection the ma rkers. Fig. 10 shows the result of detected markers on the training image.

Our method has good performance on most of the images we use to test. However, if only some part of marker is covered by sharp shadow, it may lead to bad thresholding and thus the marker may not be detected. An example is showed in Fig. 9.

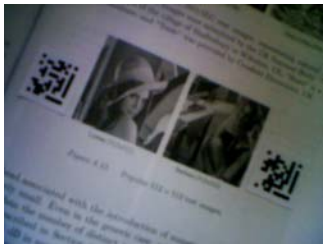
Fig. 9 Bad thresholding.
The right marker is partly covered by sharp shadow. Thus, part of the marker is likely to be binarized to black and results in bad thresholding.

REFERENCES

[1] M. Rohs, "Real-World Interaction with Camera-Phones," *2nd International Symposium on Ubiquitous Computing Systems* (UCS 2004), pp. 39-48, Tokyo, Japan, November 2004

[2] P. D. Wellner, "Adaptive Thresholding for the DigitalDesk," *Technical Report EPC-93-100*, Rank Xerox Research Centre, Cambridge, UK, 1993

[3] P. S. Heckbert, "Fundamentals of Texture Mapping and Image Warping," *Master's Thesis*, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 1989




WASH COUPONS
CAR WASH
$16 OFF
$10.99
CAR WASH
$15 OFF
$11.99
$18 OFF


PRIDE & PREJUDICE

Fig. 10. Results of marker detection on training images.