EE368 Project: Visual Code Marker Detection through Geometric Feature Recognition

George Yu, Perry Wang, Hattie Dong
Department of Electrical Engineering
Stanford University, Stanford, CA 94305
{georgeyu, peilien, dongh}@stanford.edu

I. Introduction

Visual code markers are developed as a human-machine interactive tool [1] [2]. In this class project, a code recognition program is developed to read code markers. The requirements of the code recognition program are high accuracy, speed and illumination insensitivity. There can be more than one visual code in the picture and the orientations of the visual codes are not restricted. The code markers are taken with cell-phone cameras.

Images from cell phones are poor due to several reasons. The images typically have a soft focus, caused by low quality lens, camera shake, and color sensor interpolation. Motion blur is exacerbated by the slow lens' poor light gathering ability. A low signal to noise ratio is the result of cheap sensors, analog circuitry, and lack of in-camera noise canceling hardware/software yielding significant Gaussian-like additive noise. Inadequate metering of the mobile phone camera results in poorly exposed photos. In addition, the camera sensor has low dynamic range preventing post-processing algorithms to recover details, resulting in low contrast in images. Hence, cell phone photos are more challenging for the software to recognize. The most significant challenge might be the view-angle induced distortion.

At first glace, the visual code detection problem resembles a license plate recognition problem. Various techniques have been presented for recognizing license plates based on edge detection and Hough transform [3] [4] [5] [6] [7]. Others have proposed hybrid methods combining color segmentation and statistical techniques for more accurate segmentation [8] [9]. Hough transform is a popular solution to image segmentation, as license plates typically appear square and upright in the pictures. Further, there is usually no more than one license plate per picture. Most algorithms extract vertical edges and find the most prominent vertical lines to isolate the license plates. However, visual code detection is not suitable for this technique. Visual code markers do not have closed and clear boarders, and they could have long edges in the data field as well. Furthermore, the visual code markers typically appear in highly complex environments such that the markers' perimeters are not prominent compared to its surroundings.

In this report, we present a visual code detection algorithm that combines edge detection, geometric feature recognition, and an error rejection metric to improve reliability. It does not include the Hough transform.

II. Algorithm Description

Two marker detection algorithms were considered. The first is template matching. The advantage of template matching is its relative insensitivity to noise and blur. However, a distortion-, orientation-, and size-invariant template-matching algorithm is complex and requires significant computing power. An algorithm combining edge detection, geometric feature recognition, and error rejection is more realistic.

Our algorithm is divided into the following subsections: preprocessing, edge detection, marker search, marker sampling, and error correction.

A. Preprocessing

In this step, the image is prepared for edge detection. Ample edge detection can be accomplished

without color information. Using color information helps with filtering out the unwanted edges, but the added performance is not significant enough to warrant its usage.

Sharpening is a double-edged sword. On one hand it de-blurs the images to a certain degree. On the other hand it causes edges to be less smooth and amplifies noise left over from the noise reduction. Sharpening is turned off because our edge detection algorithm does better without it, except for excessively small markers.

Contrast adjustment is the most important step because most thresholds become unreliable if the images are not normalized. We tried histogram equalization, adaptive histogram equalization, and *imadjust*, which remaps intensity values such that 1% of data points are saturated at high and low intensities. It is found that *imadjust* works the best. For uneven lighting, our contrast adjustment mechanism is quite robust such that uneven lightening needs not be corrected. For noise reduction, we use Wiener filtering from Matlab.

B. Edge Detection

Visual code marker edges are then detected. We use the Sobel operator to calculate the 2D gradient magnitude of the image. We then apply *imadjust* to the gradient magnitude image before converting it to a binary image. This is effective in removing the weaker edges and reduces runtime compared to applying Matlab edge detection functions directly to the grayscale image. The resulting binary image contains thick edges. We use a combination of morphological erosion and thinning to refine edges. This result is labeled and passed on to the marker detection logic. An example of the preprocessing and edge detection steps is shown in Figure 1.

C. Marker Detection

Refer to Figure 2 for the component layout of the visual code marker. The connected regions, or objects of the edge binary image are tagged using the bwlabel command. The properties of each object are then stored for later use. The gist of the marker search technique is successively locating components of the visual code through restraints on object properties.

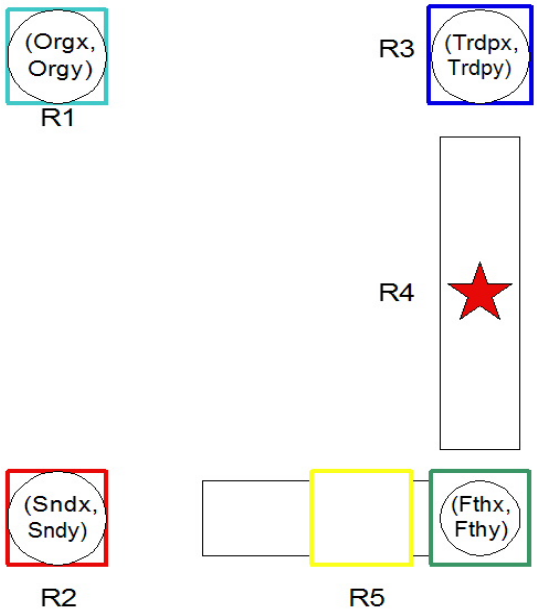


Fig. 2. Component layout of visual code marker

The first step is to locate object R4. The constraints used are aspect ratio and object area. The aspect ratio is defined as the ratio of major axis length to minor axis length. Both axes lengths and object area are stored in the regions' properties. We found through testing that an aspect ratio between 3.1 and 6 will work for all the images in the training set. Further, the object area must be greater than 50. Once R4 is found, we look for the two possible locations of object R3 at the two ends of R4. The two possible sets of coordinates are calculated using an estimation of the distance between R3 and R4. The distance estimation is based on test markers generated from the provided Matlab file. The two points p1 and p2 have coordinates (p1x, p1y) and (p2x, p2y). Program halts if R4 cannot be found.

To locate R3, we loop through all the objects a second time. If an object graindata(y) satisfies the following constraints, it is R3. The constraints on graindata(y) are: 1) the distance between its center and either p1 or p2 should be less than one fifth of R3's major axis length, 2) its equivalent diameter is less than one fifth of R3's major axis length, 3) its aspect ratio is less than 2, and 4) its area is greater than 10. Equivalent diameter is defined as the diameter of the circle whose area is the same as the object under study. These four conditions are used to ensure that R3 is in the correct location

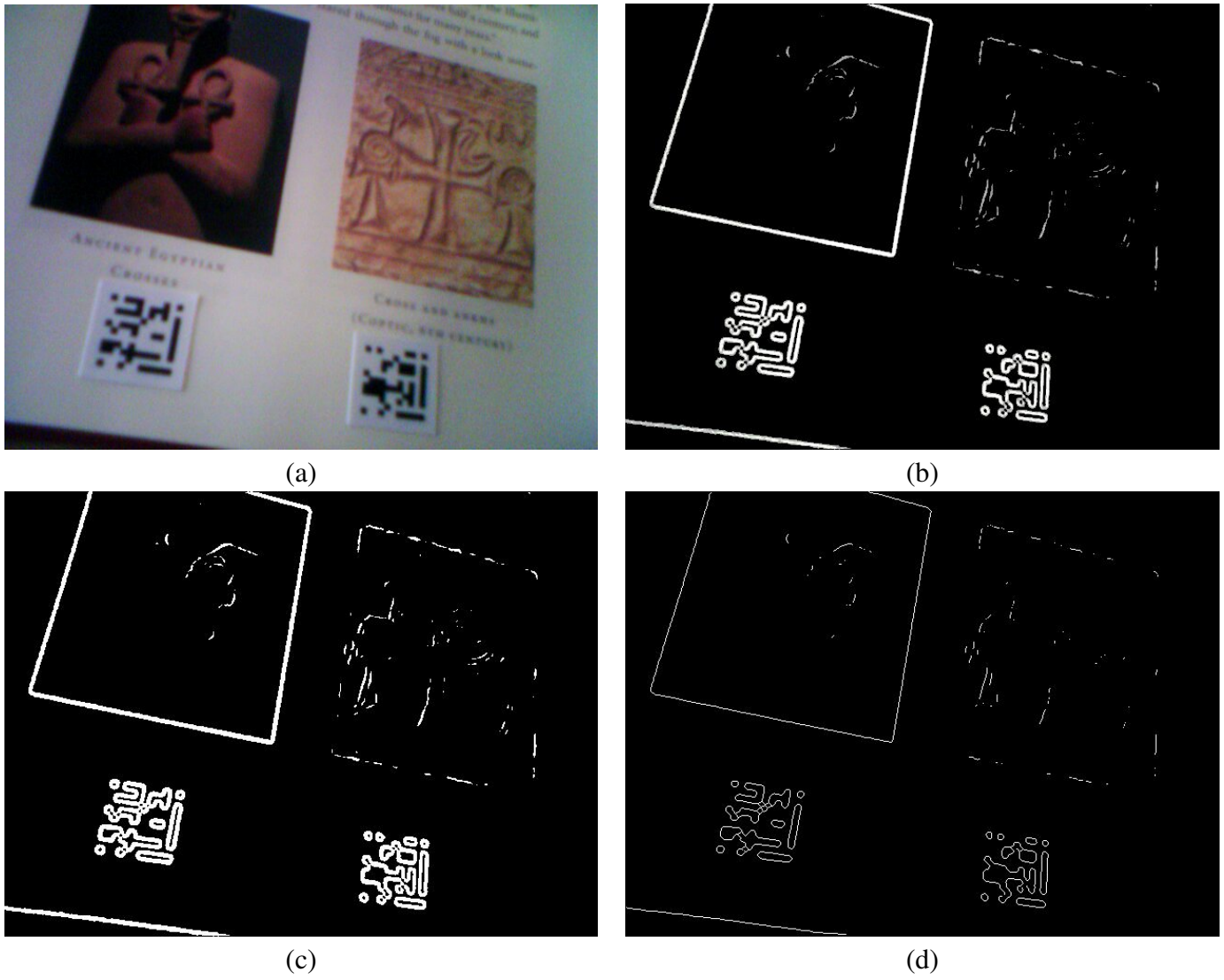


Fig. 1. Overlapping FOVs: a) Original image, b) Gradient magnitude image after imadjust and Sobel operation on a), c) Binary image after performing *imadjust* and binarization on b), d) Edge image after multiple morphological thinning on c)

and of the correct shape. Once R3 is located, R4's center is denoted by a red star, and R3's center is denoted by a blue square. R3's center is saved as (Trdpx, Trdpy). Trdp stands for the third point. The coordinates of the mirror image of R3 with respect to R4's center are then calculated to assist with locating R5. They are (otherx, othery). Program halts if R3 cannot be found.

To find R5, we again loop through all the objects. Once an object graindata(z)'s properties satisfy constraints on aspect ratio, location and angle with respect to R4, its orientation is checked. We must ensure that R4 and R5 form an "L" mirrored in its vertical bar. Using coordinate transformation, R4 defines a new x-axis x_r. From Figure 3, we can see that the center of R5 in the new coordinate must lie below y_r. Further, the absolute value of R5's new x coordinate must be less than that of R4, so that we don't accidentally pick up a faraway object resembling an R5. Once all these constraints are satisfied, we denote its center using a yellow square. We proceed to estimate the location of the "L"s bend. The two possible locations are (t1x, t1y) and (t2x, t2y). If either of these points comes within a threshold from (otherx, othery), we can be sure that graindata(z) is indeed an R5. Once R5 is found, "L"'s bend, the point Fth is denoted using a green square. The location of R2 is estimated through (t3x,

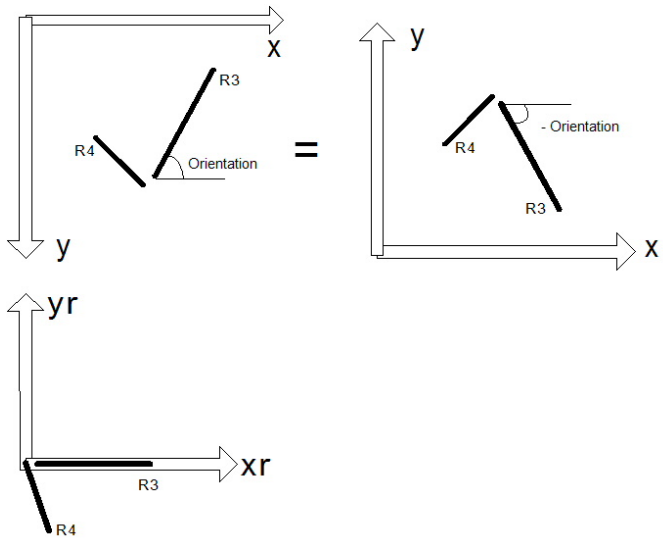


Fig. 3. Coordinate transformation to ensure correct orientation of visual code marker

t3y) and denoted using a green star. Program halts if R5 cannot be found.

To find R2, we loop through all the objects once more. If an object graindata(yy) satisfies constraints on location, diameter, aspect ratio and filled area, it is R2. We denote it using a red square. R2's centroid is stored in (Sndx, Sndy). Once R2 is found, we use the points (Sndx, Sndy), (Trdpx, Trdpy) and (Fthx, Fthy) to estimate the position of the top left hand corner of the visual code marker, (t4x, t4y). (t4x, t4y) is a reflection of the point Fth in the line formed by the two points Snd and Fth. Program halts if R2 cannot be found.

To find R1, we loop through all the objects using a set of constraints. The constraint of note is that the distance threshold between (t4x, t4y) and an object graindata(yyy) representing a possible R1 must be adjusted to accommodate distortion caused by photography. T4 is the fourth corner of the parallelogram defined by the three points Trd, Fth and Snd. The actual center of R1 has much more flexibility in terms of location. However, if the threshold distance is set too high, the program will pick up R1-like objects from markers close by. Hence the threshold distance presents a tradeoff between accounting for distortion versus the ability to separate cluttered markers. Our program favors close markers more so than distorted markers, because we think that case is of higher probability. Once R4 is located, its center is denoted by (Orgx, Orgy) and drawn onto the plot using a cyan square. Program exits if R1 cannot be found.

D. Sampling

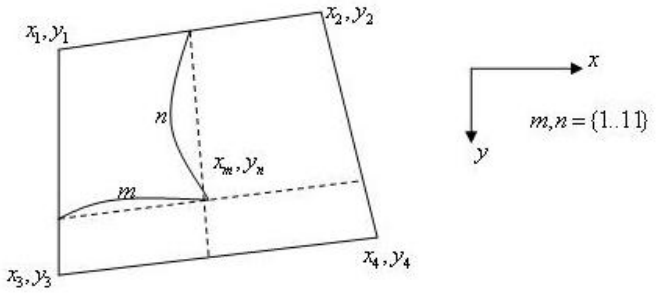
After segmentation, the four corner points are located. A bilinear sampling method, illustrated in Figure 4, is implemented to determine the coordinates of the target points in the 11 by 11 visual code marker. This method is insensitive to the distortion caused by different view angles. However, this method cannot compensate for barrel distortion caused by the lens. Five pixels, forming a cross, are sampled for each desired point and the median value of them is recorded. Choosing the median value is a means of filtering out unwanted noise. After sampling, 121 values are recorded and form an 11 by 11 picture. This picture has to be binarized before we can extract 83 data bits. Since the lighting condition is highly dynamic and noisy, we developed a dynamic thresholding method to choose the correct threshold value. First, a "scale-to-extreme" method is used to balance the gray value of the 11 by 11 picture. The equation that caculates the new gray values is

$$gray_{new} = \frac{gray(x,y) - min(gray_{x=1:m,y=1:n})}{max(gray_{x=1:m,y=1:n}) - min(gray_{x=1:m,y=1:n})}$$

Figure 4 shows the scale-to-extreme method and the histogram. After gray level balancing, the histogram is constructed and used to determine the threshold value. The threshold value is set at the center of the "valley" between the two peaks. The thresholding accuracy is satisfying for the training set provided for the project. This method achieved 100% accuracy for all 23 pictures. The final output data is a realigned set of the wanted 83 data bits. The sampling and thresholding method is illustrated in Figure 5.

E. Error Rejection

We added a routine to reject bogus visual code matches. This is done at two places. First, after a marker is detected, we go through the non-data samples and calculate the error rate of those samples. If the error is larger than a threshold, we reject the marker. Second, we use a route to



$$x_m = \{[(x_4 - x_3)\cdot(m-1)/10 + x_3] - [(x_2 - x_1)\cdot(n-1)/10 + x_1]\}\cdot(n-1)/10 + [(x_2 - x_1)\cdot(n-1)/10 + x_1]$$

$$y_n = \{[(y_4 - y_3)\cdot(m-1)/10 + y_3] - [(y_2 - y_1)\cdot(n-1)/10 + y_1]\}\cdot(n-1)/10 + [(y_2 - y_1)\cdot(n-1)/10 + y_1]$$

Fig. 4. Bilinear determination of sampling point coordinates

detect repeats. Two markers are considered repeats if the corner diagonally across from the origin is the same. Amongst the repeats, only the marker with the lowest error rate is kept. This algorithm helps removing the repeats created by distractions around the origin.

III. Results

Initially, there were several major issues with the program. First, non-dynamic thresholding caused many error bits after marker detection. Because each picture was taken under a different lighting condition, detected markers would have very different gray level histograms. Dynamic thresholding is in essence a gray-level histogram normalization procedure.

If a visual code maker has more than one plausible origin, more than one marker would be detected. To reject erroneous repeats, a routine was implemented that checks the bit error rates of all markers sharing three vertices and picks the one with the lowest error rate.

The markers have a high probability of being distorted away from their original square shape. The possible shapes are square, parallelogram, and trapezoid. The program was improved to account for all shapes.

The ”L” part of the visual code is of fixed orientation. Initially, the orientation was not taken into account, causing the program to pick up nearby objects, forming erroneous visual markers. Adding an additional constraint routine solved this problem.

The finalized algorithm described above was applied to the 23 pictures of the training set given.

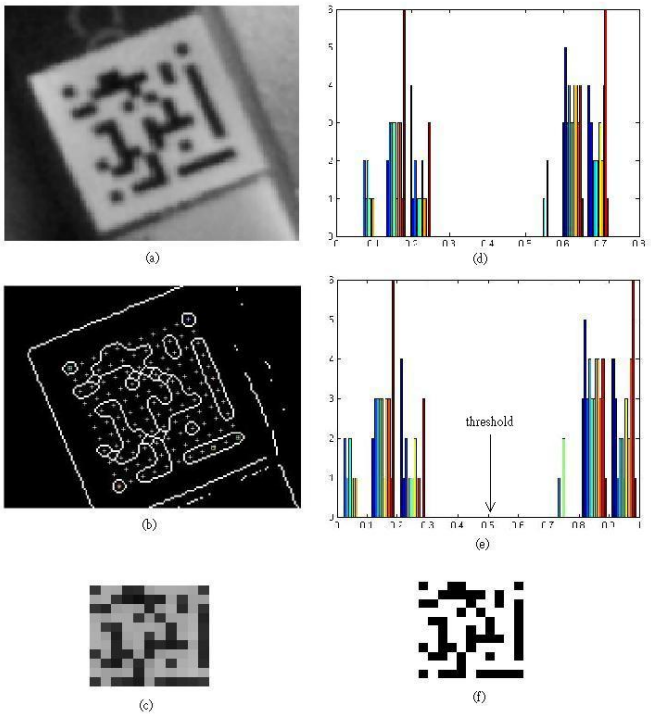


Fig. 5. (a) Original gray level picture, (b) Picture after segmentation, (c) Small picture composed of 11x11 sampled values, (d) Histogram of the small picture, (e) Histogram of the small picture after scale-to-extreme gray level balancing, (f) Restored binary picture of the 11x11 visual code marker

The result achieved is a perfect score of 1909. Two examples are shown in Figure 6.

Further, the algorithm was applied to pictures taken with our own cell phones. We tried to cover as many possibilities as possible with regards to shape, size and lighting. The algorithm was able to detect all reasonable markers and decipher most data bits correctly. Several examples are shown in Figures 7 and 8. Figure 8 shows that the algorithm can detect a distorted marker that is trapezoidal and curved. In both Figures, if a visual code marker is too small, the algorithm cannot detect it. A possible solution is to first enlarge the original picture before running the algorithm. This is however at the expense of computing power.

IV. Work Log

The work log of each group member is shown in table I.

	HD	GY	PW
literature survey	5	5	5
brain storming	5	5	5
preproccessing experimentation	3	3	8
marker detection	6	6	2
marker sampling	7	10	6
error rejection	1	1	4
create extra training set	1	2	2
debug	10	10	10
report	8	4	4

TABLE I

GROUP WORK LOG IN HOURS

V. CONCLUSIONS

In this report, we presented a visual code detection algorithm that combines edge detection, geometric feature recognition, and an error rejection metric to improve reliability. Edge detection was based on gray level image adjustment, gradient generation and morphological operations. This edge detection process was more robust than applying standard techniques directly to the original image. To locate markers, the algorithm successively located components of the visual code through restraints on object properties. Bit sampling was achieved by first using bilinear interpolation to calculate the coordinates of the desired samples, followed by dynamic thresholding to binarize the detected marker. Finally, erroneous markers were rejected by comparing bit error rates of repeats.

Applying the proposed algorithm to the given training set achieved a perfect score of 1909. When the algorithm was used on mobile phone pictures made by the authors designed to cover extreme cases, most markers were correctly detected and deciphered. Only markers very small in size compared to the overall picture were not successfully detected.

This algorithm can be used in commercial applications where visual code markers are of various size, orientation, and brightness, and are embedded in pictures of low image quality.

REFERENCES

[1] Rohs, M., Gfeller, B, "Using Camera-Equipped Mobile Phones for Interacting with Real-World Objects," *Advances in Pervasive Computing Proceedings*, Vienna, Austria, Austrian Computer Society (OCG), pp.265-271, 2004.

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

[3] Bai Hongliang, Liu Changping, "A hybrid license plate extraction method based on edge statistics and morphology," *Pattern Recognition*, Volume 2, Page(s):831 - 834, 2004.

[4] Kamat, V., Ganesan, S., "An efficient implementation of the Hough transform for detecting vehicle license plates using DSP'S," *Real-Time Technology and Applications Symposium Proceedings*, pp.58 - 59, 1995.

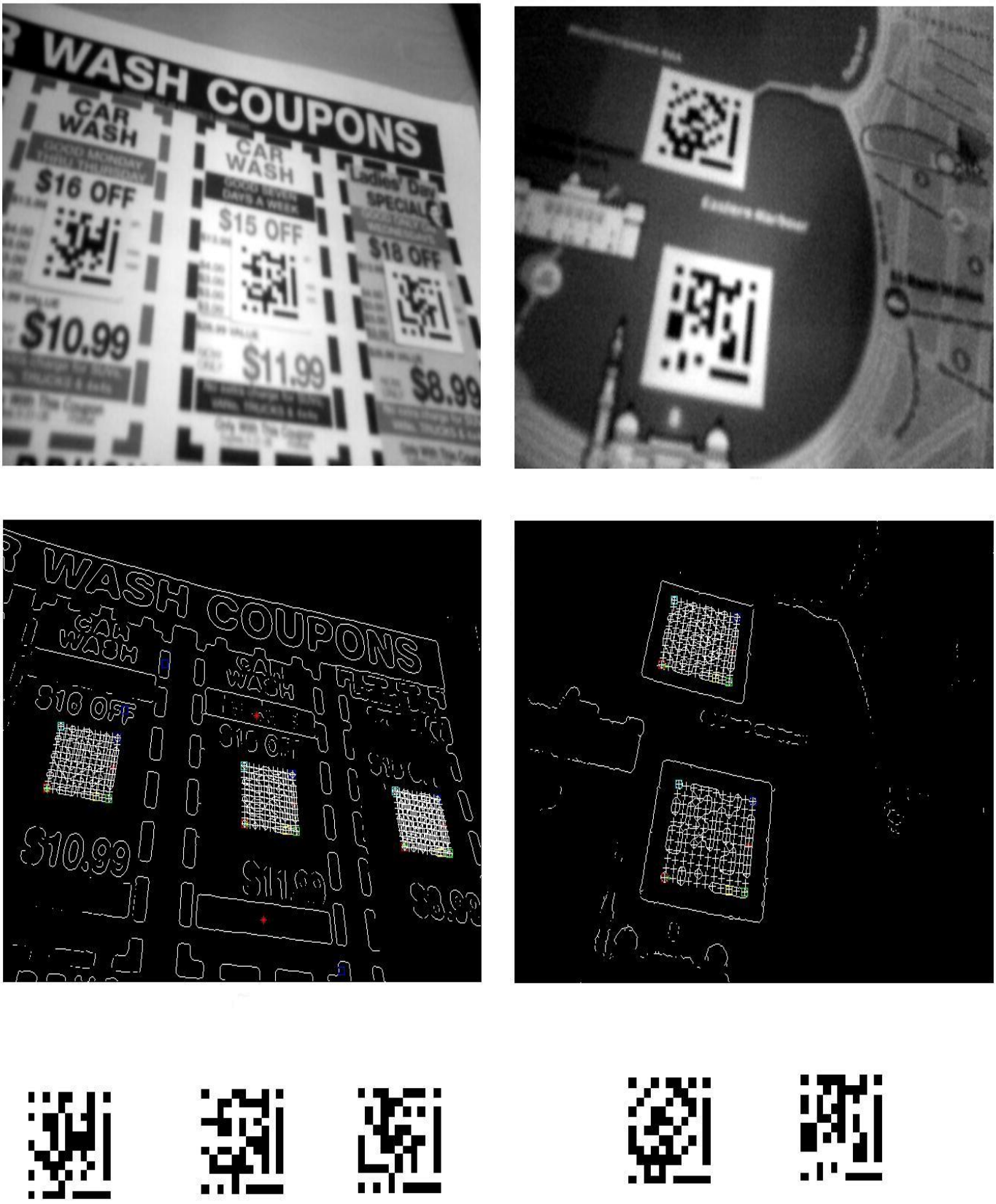
[5] Yanamura, Y., Goto, M., Nishiyama, D., Soga, M., Nakatani, H., Saji, H., "Extraction and tracking of the license plate using Hough transform and voted block matching," *Intelligent Vehicles Symposium Proceedings*, pp.243 - 246, 2003.

[6] Mei Yu, Yong Deak Kim, "An approach to Korean license plate recognition based on vertical edge matching," *IEEE International Conference on Systems, Man, and Cybernetics Proceedings*, pp.2975 - 2980 vol.4, 2000.

[7] Tran Duc Duan, Duong Anh Duc, Tran Le Hong Du, "Combining Hough transform and contour algorithm for detecting vehicles' license-plates," *2004 International Symposium on Intelligent Multimedia, Video and Speech Processing Proceedings*, pp.747 - 750, 2004.

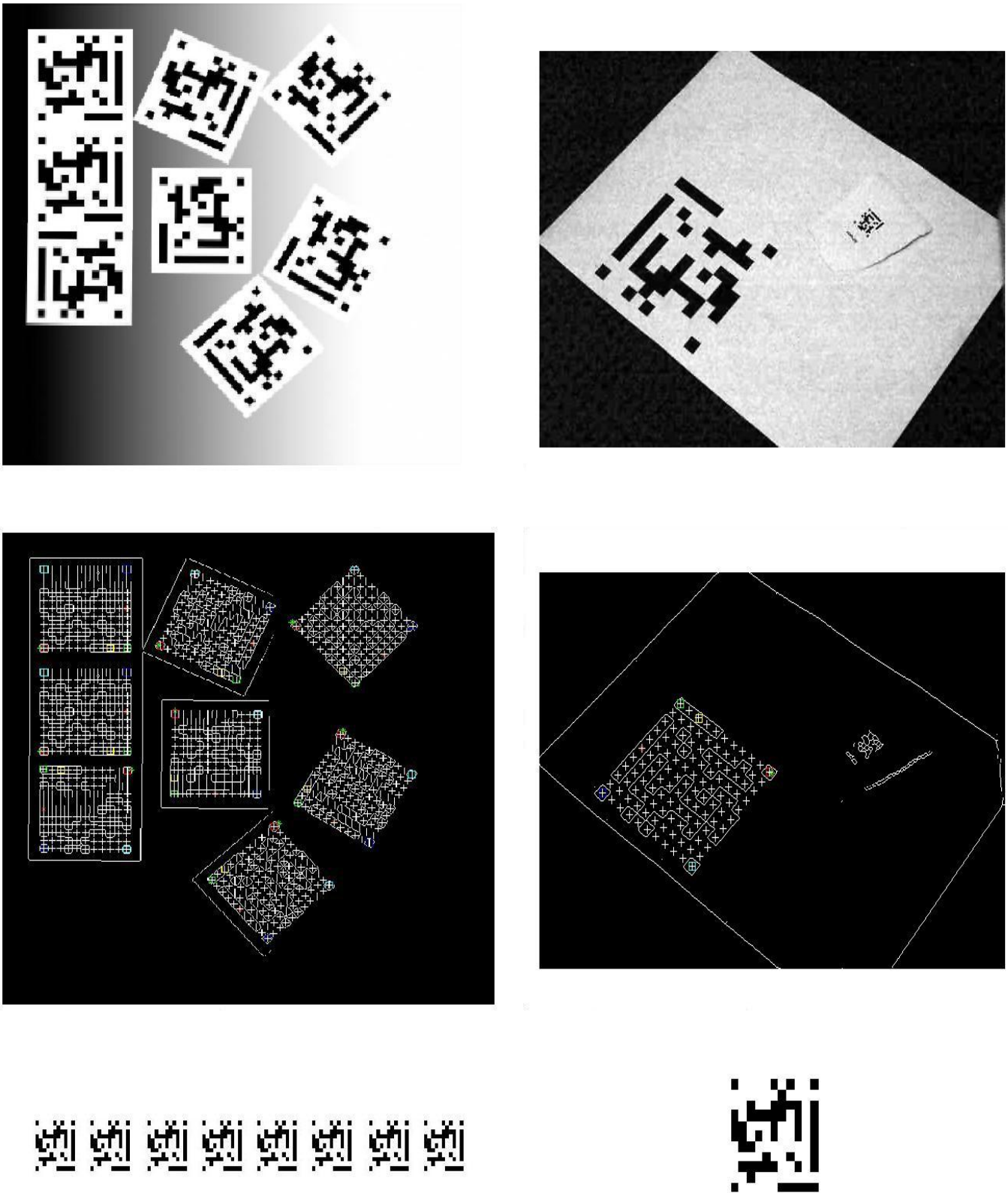
[8] Jian-Feng Xu, Shao-Fa Li, Mian-Shui Yu, "Car license plate extraction using color and edge information," *2004 International Conference on Machine Learning and Cybernetics Proceedings*, pp.3904 - 3907 vol.6, 2004.

[9] Syed, Y.A., Sarfraz, M, "Color edge enhancement based fuzzy segmentation of license plates," *Ninth International Conference on Information Visualisation Proceedings*, pp.227 - 232, 2005.



(a) (b)

Fig. 6. Training pictures given by TA. Top: original picture. Middle: edge image. Bottom: restored markers. a) Training_5.jpg, b) Training_12.jpg



(a) (b)

Fig. 7. Pictures made by authors. Top: original picture. Middle: edge image. Bottom: restored markers.

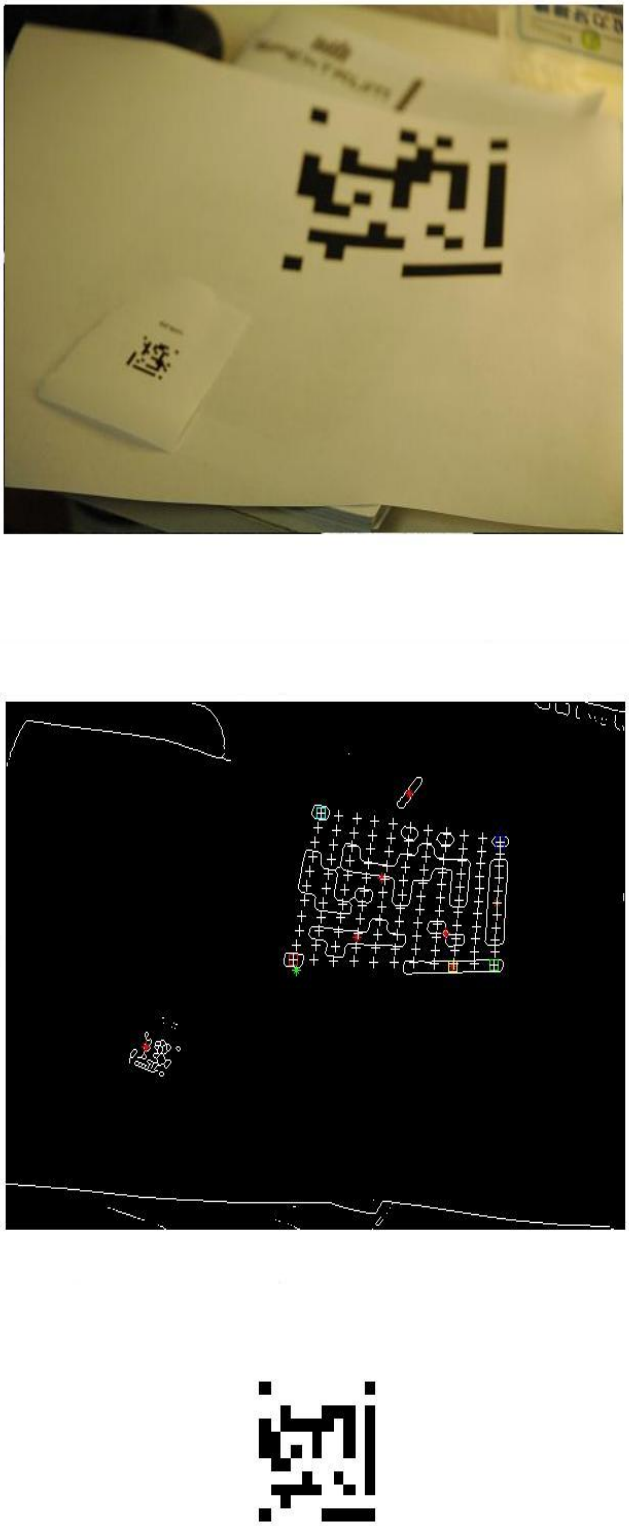


Fig. 8. Picture made by authors with trapezoidal and curved visual code marker. Top: original picture. Middle: edge image. Bottom: restored markers.