Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 515296, 13 pages
doi:10.1155/2012/515296

Research Article

Data Matrix Code Location Based on Finder Pattern Detection and Bar Code Border Fitting

Qiang Huang,[1] Wen-Sheng Chen,[1, 2] Xiao-Yan Huang,[1] and Ying-Ying Zhu[1]

[1] *College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China*
[2] *College of Mathematics and Computational Science, Shenzhen University, Shenzhen 518060, China*

Correspondence should be addressed to Ying-Ying Zhu, zhuyy@szu.edu.cn

Received 25 November 2011; Accepted 7 January 2012

Academic Editor: Bin Fang

The 2-D bar code possesses large capacity of data, strong ability for error correction, and high safety, which boosts the 2-D bar code recognition technology being widely used and developed fast. This paper presents a novel algorithm for locating data matrix code based on finder pattern detection and bar code border fitting. The proposed method mainly involves three stages. It first extracts candidate regions that may contain a data matrix code by morphological processing and then locates the data matrix code roughly by detecting "L" finder pattern and the dashed border on the candidate regions. Finally, the lines fitted with the border points are used as the borders of data matrix code. A number of data matrix code images with complexity background are selected for evaluations. Experimental results show that the proposed algorithm exhibits better performance under complex background and other undesirable conditions.

1. Introduction

2-D bar code consists of a certain white and black geometric modules that alternately arrange in the vertical and horizontal directions according to certain rules (see Figure 1), and it is a symbol with large capacity for storing information. As the 2-D bar code with smallest size in the world, data matrix code is widely applied to electronic product components. 2-D bar code recognition technology shows great commercial value, and at present, most COTS (commercial of the shell) recognition algorithms are proprietary and protected by patents, so the 2-D bar code recognition technology is in a great demand for researching.

How to locate a 2-D bar code quickly and precisely in an image with complex background, poor illumination or other undesirable condition is crucial to the recognition process. For data matrix code location, many kinds of location algorithms have been

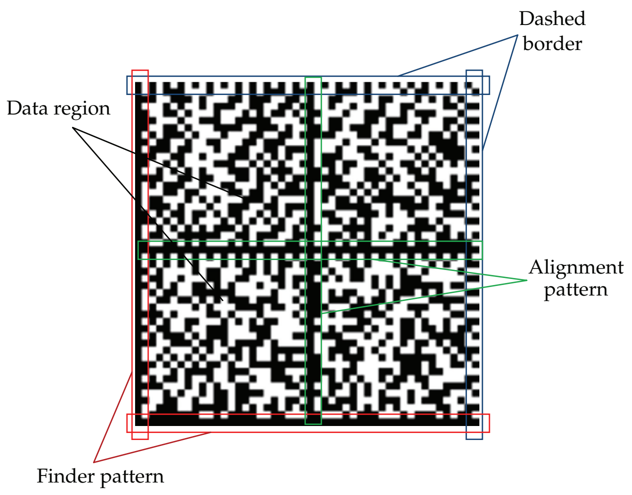

Figure 1: Data matrix code symbol.

proposed. Donghong et al. [1] proposed an algorithm based on Radon transform, which mainly locates data matrix code by the "L" finder pattern and dashed border detection. This algorithm has high precision and works well for the data matrix code within high density but is very time consuming and is not suitable to be applied to real-time application. Chenguang et al. [2] proposed the locating algorithm based on Hough transform. This algorithm is very time consuming and space consuming though it can reduce the consumption by a second Hough transformation. What's worse, this algorithm has low precision for the complex background. Wenting and Zhi [3] discussed the method of locating data matrix code based on convex algorithm, which determines the 3 vertexes of the "L" finder pattern according to the convex of the edge points of the bar code. This algorithm is simple and fast but requires that background is clean and the bar code gets no stained and complete. There are other locating algorithms [4, 5] and are solely appropriate for specific situation, such as simple background situation, good illumination condition, and low density. In reality, the bar code images are always accompanied with complex background, and furthermore the images might get stained, incomplete, or printed in high density. Under these undesirable conditions, most of the algorithms mentioned above do not work effectively or are not demanded for higher processing power and more storage space, which cannot satisfy most real-time application. The location problem of 2-D barcode involves nonlinear systems [6, 7].

In this paper, we propose a data matrix code location algorithm based on finder pattern detection and bar code border fitting, which is proved by extensive experiments to be effectively and fast. In this algorithm, the finder pattern is detected mainly based on line segment detection and combination. Some work has been done for finding "L" finder pattern by segment detection, such as reference [8–12]. About line segment detection, Grompone Von Gioi et al. [13] proposed a linear-time algorithm called line segment detector (LSD), which requires no parameter tuning and gives accurate results. The LSD algorithm has improved the line segment finder proposed by Burns et al. [14] and combined with a validation criterion inspired from Desolneux et al. [15, 16]. In this paper, the LSD algorithm is utilized to detect the "L" finder pattern, and an introduction of LSD is given in the related work section. About border fitting, the most important step is straight line fitting. An effective straight line fitting solution is proposed by Fischler and Bolles called RANSAC [17] algorithm, which will be used to fit the bar code borders.

The remainder of this paper will be organized as followings. Section 2 is the introduction of the related algorithm about line segment detection. Section 3 will give details of the proposed data matrix code location algorithm. Section 4 comments on the experimental results and Section 5 concludes the paper.

2. Related Work

LSD is a linear-time line segment detector, which draws and improves the idea of Burns et al. (see [14]) that defines a line segment as a region which only concerning the gradient information, and combines the validation criterion inspired from Desolneux et al. (see [15, 16]). This algorithm is implemented in 4 steps.

Step 1 (finding the line-support regions). This method defines a line segment as a region called line-support region, which is a cluster of points in a connected region that sharing roughly the same gradient orientation angle and whose gradient magnitude is greater than a threshold. To get the line-support region, a region-grow algorithm is applied. Firstly, a pseudoordering is done. The pixels whose gradient magnitude value is larger than threshold ρ are classified into a finite number of bins according to their gradient magnitude value, then the pixels from higher bins are visited first and pixels in lower bins later; secondly, each region starts with one pixel and initializes the line-support region angle with the first pixel's gradient angle, if an adjacent pixel is under the condition as inequality (2.1), add it to the line-support region and update the line-support region angle as formula (2.2). Repeat the steps until there is no pixel that can be added to the line-support region.

$$\text{abs}\left(\text{angle}(\overline{\text{p}}) - \theta_{\text{region}}\right) < \tau, \tag{2.1}$$

where $\overline{\text{p}}$ is the adjacent pixel, $\tau = 22.5°$ is the threshold of the difference between adjacent pixel's angle and line-support region angle.

$$\theta_{\text{region}} = \arctan\left(\frac{\sum_i \sin(\text{ang}_i)}{\sum_i \cos(\text{ang}_i)}\right). \tag{2.2}$$

Step 2 (finding the rectangular approximation of every line-support regions). A line segment that associated with a line-support region is defined as a rectangle which is the rectangular approximation of the line-support region. Parameters, such as center, orientation angle, length, and width, can be used to describe a line segment, see Figure 2. In LSD algorithm, instead of using the mean level-line angle as the line angle, which may lead to erroneous line angle estimation, the first inertia axis orientation is used to be the line segment orientation. The centroid of mass of the rectangular approximate is selected as the center, when the gradient magnitude is used as pixel's mass. The length and width are chosen in the way that covers the line-support region.

Step 3 (validation of each potential line segment). After getting the rectangular approximations of line-support region, it is necessary to validate each approximation a line segment

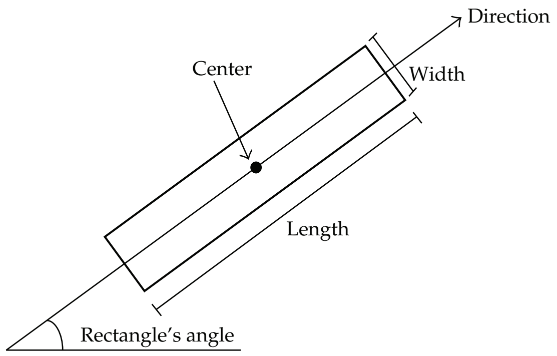

Figure 2: Line segments are characterized by a rectangle determined by its center point, angle, length, and width.

or not according to the number of aligned point and the total number of pixels of each approximation. Aligned point is defined as the pixel whose gradient angle is the same to the line segment angle up to the tolerance τ.

Suppose that image background has the Gaussian white noise model H_0, more formally, an image X under the background model H_0 is a random image (defined on the grid $\Gamma = [1, N] \times [1, M] \subset Z^2$) such that:

(1) for all $m \in \Gamma$, $\text{angle}(\nabla X(m))$ is uniformly distributed over $[0, 2\pi]$.

(2) The family $\{\text{angle}(\nabla X(m))\}_{m\in\Gamma}$ is composed of independent random variables.

Under the model H_0, image is isotropic flat zones, while straight edges are exactly the opposite: highly anisotropic zones. Thus, in practice, a set of pixels will not be accepted as a line segment if it could have been formed by an isotropic process. This algorithm defines the Number of False Alarms of a rectangle $r \in R$ in an image x, as

$$\text{NFA}(r, x) = \#R \cdot IP_{H_0}[k(r, X) \geq k(r, x)]. \tag{2.3}$$

In the formula (2.3), X is a random image under model H_0, $\#R$ is the total number of potential rectangle (line segment) in image X, $k(r, X)$, and $k(r, x)$ represent the number of aligned points in rectangle in image X and image x, respectively, $IP_{H_0}[k(r, X) \geq k(r, x)]$ is the probability that $k(r, X)$ greater than or equal to $k(r, x)$.

The smaller the NFA value is, the more significant the rectangle r is. Consider one pixel accuracy, there are N^4 potential line segments in a $N \times N$ image for the start point and end point both have N^2 possible, but in practice, line segment's width can be at most N pixels, thus $\#R = N^5$. Under the model H_0 and a tolerance $\tau = \pi p$ for the difference between aligned point gradient angle and line segment angle, $p = \tau/\pi$ is the probability of that a given point is an aligned point. Each pixel's gradient is independent in rectangle, so the number of aligned point $k(r)$ obey binomial distribution, thus,

$$IP_{H_0}[k(r, X) \geq k(r, x)] = b(n(r), k(r), p), \tag{2.4}$$

where $b(n,k,p) = \sum_{i=k}^{n}(n/i)p^{i}(1-p)^{n-i}$, finally, NFA is easy to be calculated:

$$\mathrm{NFA}(r) = N^{5} \cdot b\big(n(r), k(r), p\big). \tag{2.5}$$

The NFA is the key to validate the rectangle as a line segment or not, if NFA is less than a threshold ε, then the rectangle is accepted as a ε-meaningful line segment, vice, the rectangle is not a line segment. This method is almost independent of ε (actually logarithmic). Thus, this algorithm let $\varepsilon = 1$ thoroughly as advised by Fuchao (see [17]).

Step 4 (improved the approximations of line-support region and validate them). In Step 3, the best rectangular approximation of a line-support region is the one that gives the smallest NFA value, in order to get a better NFA, the LSD algorithm tries to adjust the width of the approximation and the probability p. Five dyadic precision steps are considered before adjusting the rectangle width and again five dyadic precision steps afterward. Repeat Steps 2 and 3 and keep the rectangular approximation with best NFA value as the final line segment.

3. The Proposed Location Algorithm

Observed the data matrix code, the “L” finder pattern and the dashed border make data matrix code distinct from other 2-D bar codes or objects, so the first idea that comes to mind is to locate the data matrix code by detecting the “L” finder pattern and dashed border, but this procedure may lead to an imprecise location when there exists a perspective distortion in the bar code image or the image get stained or obscured (these situations happen frequently). Therefore, it needs to locate the bar code more precisely. In this paper, we utilize some edge points of data matrix code to fit 4 straight lines as the 4 borders of the bar code and finally achieve accurate positioning with their intersections (4 vertexes).

The location steps discussed above are the main steps of the location algorithm, which consume most of processing time, and the computation time is closely related to the size of the image. Normally, the bar code region is only a small part of the image, it is necessary to extract the bar code from the image to facilitate the follow-up location. In this way, it not only reduces the cost of computation but also enhances the antijamming capability of the algorithm for the extraction of bar code removes most of the background. After the extraction of candidate regions (may include a data matrix code), the location algorithm is applied on them to complete the whole location. The whole data matrix code location procedure is shown in Figure 3.

3.1. Extraction of Data Matrix Code Candidate Region

In general, 2-D barcodes consist of staggered white and black modules and have vast closely spaced edges, while other objects or background in the image has few sparse edges. This characteristic is utilized to extract candidate regions in 3 steps:

Step 1 (edge detection, remove most of the background by canny edge detection). The proposed algorithm chooses canny operator to do edge detection because it can get more complete edges by restraining the nonmaximum value and connecting the inconsecutive edges with mathematical morphology. Canny operator gets a good tradeoff between noise suppression and edge detection. In the standard canny edge detection, the first step is the

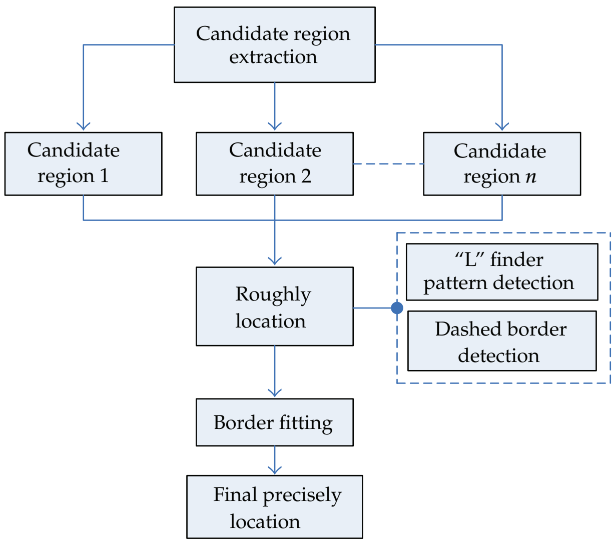

Figure 3: Data matrix code location algorithm flowchart.

Gaussian filtering which is used to remove noise, but in this application, we ignore this step to save time bringing little impact on subsequent processing.

Step 2 (morphological processing, highlight the bar code region by dilate operation and open operation). The dilate operation fills bar code but expands bar code boundary at the same time, which may lead to connection of bar code and other objects (such as text). So open operation is used to separate these small adhesions. The result is susceptible to the shape and size of the structure element. Actually, 2-D barcode module always has the size of 3 to 8 pixels in rectangle shape, so the structure elements are defined as follows.

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

Dilate structure element:
4 × 4 rectangle

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Open structure element:
5 × 5 rectangle

(3.1)

Step 3 (contour analysis, filter candidate regions with contour perimeter and area). Mark every connected region: $\{R_1, R_2, \ldots, R_n\}$ and extract the contour: $\{C_1, C_2, \ldots, C_n\}$, then filter the connected regions with their perimeter and area:

$$\text{candidate\_regions} = \{R_i \mid \text{Perimeter}(C_i) > \tau_1 \ \& \ \& \ \text{Area}(R_i) > \tau_2\}. \tag{3.2}$$

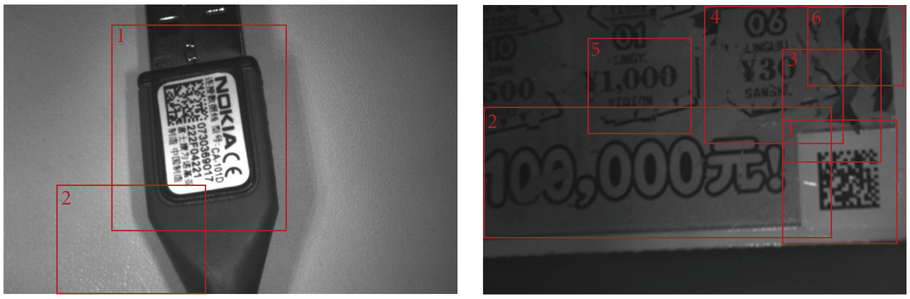
(a) (b)

Figure 4: Extraction of bar code candidate region.

To make sure that the whole bar code is included in the candidate region, the bounding box of the candidate is calculated and expanded. The experimental results are shown in Figure 4.

3.2. *Preliminary Location Based on Finder Pattern*

Based on the data matrix code feature analysis mentioned above, it can quickly determine whether there is a data matrix code or not in a candidate region and obtain the approximate location of data matrix code by detecting the “L” finder pattern. Detection of dashed border helps determine the top and right boundary position of data matrix code. In this paper, “L” finder pattern and the dashed border are detected to preliminarily locate data matrix.

3.2.1. “L” Finder Pattern Detection

L-detection is relatively complex, but an “L” finder pattern can be regards as two segments, and there are many mature line segment detection algorithms. The LSD algorithm (introduced in Section 2) proposed by Grompone Von Gioi et al. [13] is utilized to detect line segment. Consider that Step 4 of LSD is not very meaningful for large and well-contrasted line segments (the “L” finder pattern of data matrix code is always well contrasted) and time consuming within repeat of Steps 2 and 3, our algorithm ignores this step to save time. The detection result is shown in Figure 5.

An “L” finder pattern can be detected by combining the appropriate line segments. Assume that the line segments obtained in a candidate region are: $\{l_1, l_2, \ldots, l_n\}$, a line segment is described with its two endpoints as: $l_i = \{p_{i1}, p_{i2}\}$. Normally, if two line segments are belong to the same “L” finder pattern, then $\text{angle}(l_i, l_j) = 90^\circ$, $i, j = 1, 2, \ldots n$. But this assumption no longer holds when the image has perspective deformation. Extensive experiments have demonstrated that the angle usually ranges from 60° to 120°. A constrain about the segments’ length ratio that the length of the long line segment can not exceed 5 times the length of the short one is added. Therefore, combination of line segments can be implemented as in Algorithm 1.

Inevitably, some pseudo-L will be detected, so an “L” finder pattern will be abandoned if the postprocessing cannot locate a data matrix code. The approximate location of 3 vertexes (two endpoints and an intersection of “L”) can be obtained from the “L” finder pattern to locate data matrix code roughly just as Figure 5(c).

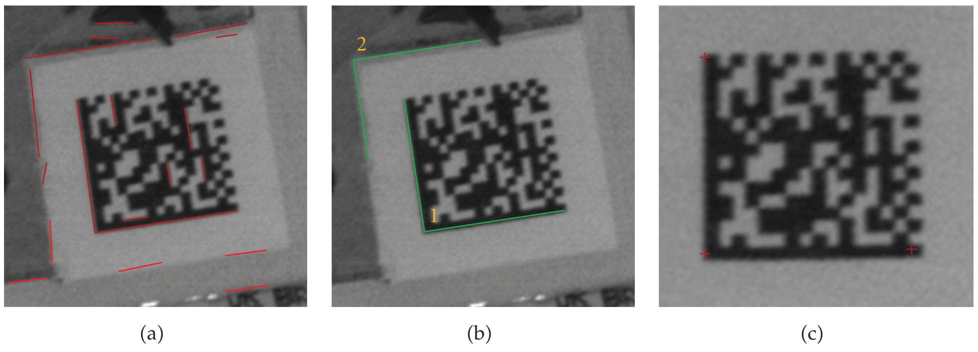

Figure 5: "L" finder pattern detection (a) shows the result of line segment detection; (b) combine the appropriate segments to an "L"; (c) roughly location using the 3 vertexes.

```
For each $l_i$ $(i = 1 : n)$
    If   $l_i$ is not marked
        Search in $D_{p_{i1}}$ and $D_{p_{i2}}$ to find $l_j$; // $D_{p_{i1}}$ and $D_{p_{i2}}$ are Neighborhoods of $p_{i1}$ and $p_{i2}$.
        If   $l_j$ is not marked &&   $60^\circ < \text{angle}(l_i, l_j) < 120^\circ$
          && $\max\{\text{len}(l_i), \text{len}(l_j)\} / \min\{\text{len}(l_i), \text{len}(l_j)\} < 5$ // $\text{len}(l_i)$ and $\text{len}(l_j)$ are the lengths of $l_i$ and $l_j$
            Combine $l_i$ and $l_j$ into an "L";
            Mark $l_i$ and $l_j$;
        End
    End
End
```

Algorithm 1

3.2.2. *Dashed Border Detection*

Composed by alternating black modules and white modules, the dashed border has a lot of edges. On the other hand, the dashed border is roughly parallel to the "L" finder pattern. Detecting a dashed border by scanning edge point in the direction paralleling to "L" in two steps.

Step 1 (determine a detecting region). A quadruple: $\{x, y, \text{width}, \text{height}\}$ is used to defined the detecting region, where x and y are coordinates of the start point of the scanning, width and height are scanning arranged paralleling to "L" (see Figure 6). The 3 vertexes of the "L" are $p_1(x_1, y_1)$, $p_2(x_2, y_2)$, and $p_3(x_3, y_3)$, and the lengths of the two segments of "L" are len_1 and len_2. $\text{len}_1 > \text{len}_2$, then the detection regions are

$$\begin{aligned}\text{detect\_reg}_{\text{upper}} &= \{x_1 - \tau, y_1 - \tau,\ \text{len}_1 + 2\tau,\ \text{len}_1 - \text{len}_2 + 2\tau\},\\ \text{detect\_reg}_{\text{right}} &= \{x_3 + \tau, y_3 + \tau,\ \text{len}_1 - \text{len}_2 + 2\tau, \text{len}_1 + 2\tau\},\end{aligned} \tag{3.3}$$

where τ is an adjustable parameter.

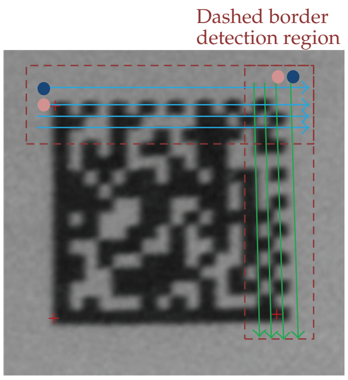

Figure 6: Dashed border detection.

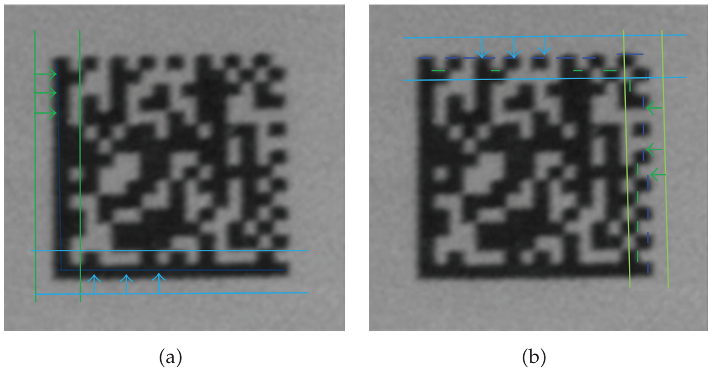

Figure 7: Scan border points.

Step 2. Progressively scan edge points in the direction paralleling to the horizontal line segment of "L" in detect_reg$_{\text{upper}}$ and count the number of edge points. The row with the most edge points is kept as the horizontal dashed border. The row with the least edge points is kept as static region. The vertical dashed border is detected in the same way.

3.3. *Border Fitting*

The 3 vertexes obtained from the "L" finder pattern can only roughly locate data matrix code, even, the location is wrong when the "L" gets stained or partly covered and the detection of the dashed borders is not precise. So it is necessary to get more information for further location. In this paper, the 4 borders will be fitted to finally locate data matrix code in 3 steps.

Step 1 (scan border points). Border point is defined as the first edge point from outside to inside in the direction perpendicular to bar code's border. The scanning range is around the bar code borders with an offset (such as 5 pixels). An example has been demonstrated in Figure 7, 4 sets of border points can be obtained: border_point$_{\text{left}}$, border_point$_{\text{right}}$, border_point$_{\text{upper}}$, and border_point$_{\text{lower}}$.

Step 2 (fit borders). In mathematics, it is usually to describe a straight line by equality [18]: $y = kx + b$, but it cannot work when the straight line is perpendicular to the x axis, the slope k tends to be infinity, thus we use Hesse paradigm to describe a straight line: $ax + by + c = 0$, in fact, it is a excessive parametric expression, 2 points can determine a straight line while there are 3 parameters in the equality, so a constraint is added: $a^2 + b^2 = 1$.

In order to fit a straight line in a set of point: $\{p_1(x_1, y_1), p_2(x_2, y_2), \ldots, p_n(x_n, y_n)\}$, the sum of the distance from each point to the straight line: $e^2 = \sum_i^n (a_i + by_i + c)^2$ should be minimized. Therefore, the straight line fitting can be converted into distance minimization problem. But if $a = b = c = 0$, the equality above will get a zero measurement error. To avoid this problem, the constraint condition $a^2 + b^2 = 1$ would be added as a Lagrangian multiplier:

$$e^2 = \sum_i^n (ax_i + by_i + c)^2 + \lambda\left(a^2 + b^2 - 1\right)n. \tag{3.4}$$

The minimization problem can be expressed as:

$$(a, b, c) = \arg\left(\min\left(e^2\right)\right). \tag{3.5}$$

Solve (3.5):

$$\frac{a}{b} = -\frac{s_{xy}}{s_{xx}} = -\frac{s_{yy}}{s_{xy}}, \qquad c = -(a\overline{x} + b\overline{y}) \tag{3.6}$$

to avoid the situation that $a/b = -0/0$, accept the proposition that with greater value:

$$\begin{gathered}\text{if } s_{yy} \geq s_{xx}, \quad \frac{a}{b} = -\frac{s_{yy}}{s_{xy}}, \quad \text{then } a = -\frac{s_{yy}}{\sqrt{s_{yy}^2 + s_{xy}^2}}, \quad b = -\frac{s_{xy}}{\sqrt{s_{yy}^2 + s_{xy}^2}}; \\ \text{if } s_{yy} < s_{xx}, \quad \frac{a}{b} = -\frac{s_{xy}}{s_{xx}}, \quad \text{then } a = -\frac{s_{xy}}{\sqrt{s_{xx}^2 + s_{xy}^2}}, \quad b = -\frac{s_{xx}}{\sqrt{s_{xx}^2 + s_{xy}^2}}.\end{gathered} \tag{3.7}$$

In the formulas above:

$$\begin{gathered}\overline{x} = \frac{1}{n}\sum_i^n x_i, \qquad \overline{y} = \frac{1}{n}\sum_i^n y_i, \\ s_{xx} = \sum_i^n (x_i - \overline{x})^2, \qquad s_{yy} = \sum_i^n (y_i - \overline{y})^2, \qquad s_{xy} = \sum_i^n (x_i - \overline{x})(y_i - \overline{y}).\end{gathered} \tag{3.8}$$

Once the values of parameter a, b, c are determined, a straight line is fitted. Let us consider the points in the set, some outliers must deviate the straight line which fitted by minimizing the distance from points to it. An effective solution is the mature and typical RANSAC algorithms. This algorithm selects minimum quality of points (such as 2 points)

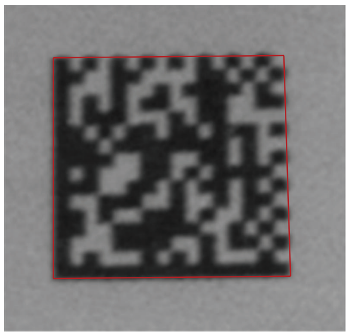

Figure 8: Precisely location.

randomly to fit a straight line, and then check the proposition of outliers. Repeat this procedure until the proposition of outliers reaches level less than a threshold such as 1%, and the straight line with minimum proposition of outliers is kept, finally used these certain 2 points and their inliers to fit the final straight line. The robustness RANSAC is used in this paper to fit 4 borders with the 4 set of border points obtained in Step 1.

Step 3 (obtain 4 vertexes of data matrix code precisely). After the Step 2, 4 straight lines l_{left}, l_{upper}, l_{right}, and l_{lower}, have been fitted and then calculate their intersections to obtain data matrix code position. The Figure 8 shows the final precise location.

4. Experimental Results

In order to verify the performance of the proposed algorithm, we conduct experiments on the images under different conditions (such as complex background, perspective distortion) and give four representative experimental results. All the experiments run on the ARM 11 hardware platform, and all test images have the resolution of 752×480. Figures 9(a)~9(d) are the four original test images. Figure 9(a) is the mobile phone battery image with a data matrix code. Around the data matrix code, there are some texts and other objects closed to the bar code, which bring great interference to bar code location. Figure 9(b) is an USB data cable connector image with a data matrix code. The size of the USB connector is small and the size of data matrix code is even smaller. Also, there are lots of texts very closed to the data matrix code. Figure 9(c) is an award ticket image with a data matrix code; the image is dim with bad illumination. Figure 9(d) is an image of the printed data matrix code, some perspective distortions exist in the image because of the tilt angle when taking photo.

Figures 9(e)~9(h) are the experimental results of data matrix code location. It can be seen that the proposed location method works effectively and precisely. Figure 9(e) shows that the proposed algorithm has good robustness to interference brought by complex background. Figure 9(f) demonstrates that the algorithm has high precise for the small size of data matrix code. Figures 9(g) and 9(h) reveal that the proposed algorithm can achieve good performance even under bad illumination or distortion conditions.

Moreover, the proposed algorithm costs no more than 100 ms in these experiments, which can completely satisfy the demand of real-time application, and it is especially suitable for embedded device.

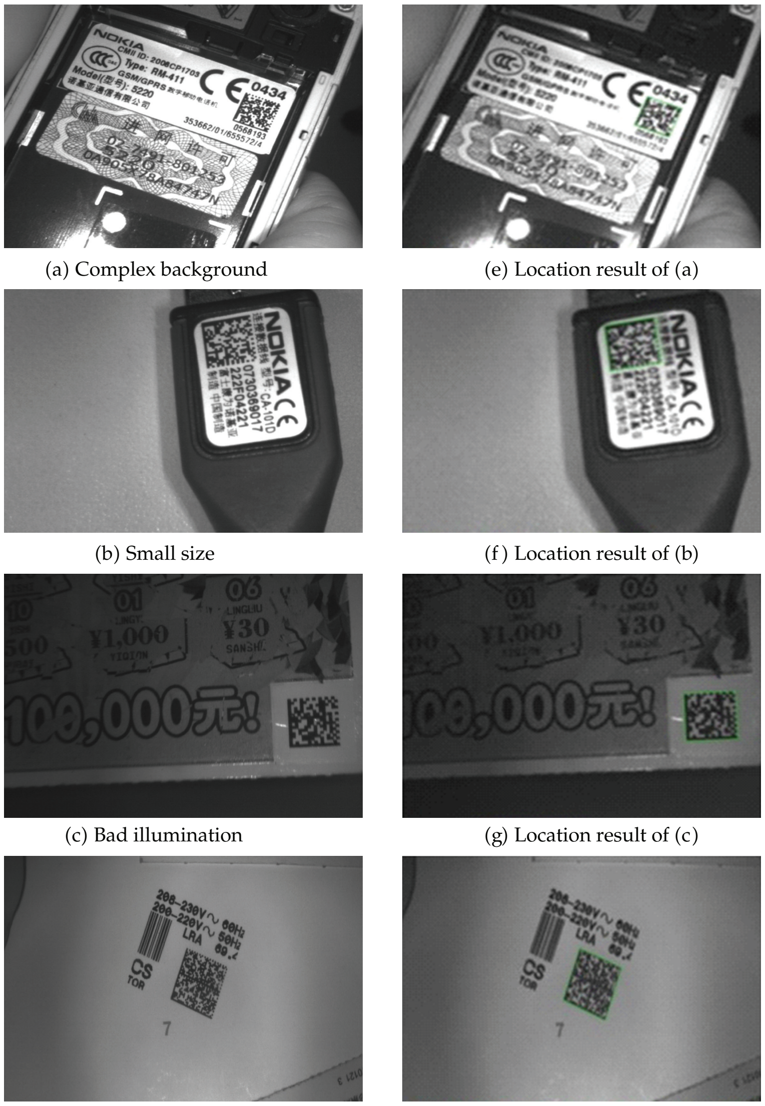
(a) Complex background (e) Location result of (a)
(b) Small size (f) Location result of (b)
(c) Bad illumination (g) Location result of (c)
(d) Distortion (h) Location result of (d)

Figure 9: Data Matrix code precise locations.

5. Conclusions

A data matrix code location algorithm is proposed in this paper, which utilizes the obvious features of "L" finder pattern and dashed border of data matrix code. This algorithm provides 3 advantages compared to those algorithms mentioned. (1) First: robustness, it locates data matrix code on candidate regions excluding most of interference from background. The two most important algorithms, namely, LSD and RANSAC, are used to achieve high robustness under complexity background conditions. (2) Second: it has high accuracy, preliminarily location by finder pattern, and then accurate location by fitting border lines. (3) It is suitable for real-time application. Extraction of candidate region greatly reduces the operating area of location algorithm, which saves a lot of time. Instead of using the time-consuming Hough transformation algorithm to detect line segment, the linear-time LSD algorithm is used. The proposed method is evaluated on four images with complex background or distortion. The experimental results show that our proposed algorithm gives good performance.

Acknowledgments

This work was partially supported by Natural Science Foundation of China under Grant no. (60501026, 60873168), foundation for combination of industry and academy in Shenzhen under Grant no. SY200806270120A and Science & Technology Planning Project of Shenzhen City (JC200903130300A), and the Opening Project of Guangdong Province Key Laboratory of Computational Science of Sun Yat-Sen University (201106002).

References

[1] H. Donghong, T. Hui, and C. Xinmeng, "Radon transformation applied in two dimensional barcode image recognition," *Journal of Wuhan University*, vol. 5, pp. 584–588, 2005.

[2] Z. Chenguang, Y. Na, and H. Rukun, "study of two dimensional barcode identification technology based on HOUGH transform," *Journal of Changchun Normal University*, vol. 4, pp. 94–98, 2007.

[3] C. Wenting and L. Zhi, "Two dimensional barcode localization algorithm based on convex," *Journal of Zhejiang University*, vol. 46, pp. 669–672, 2008.

[4] M. Li and W. Zhao, "Visiting power laws in cyber-physical networking systems," *Mathematical Problems in Engineering*, vol. 2012, Article ID 302786, 13 pages, 2012.

[5] M. Li, C. Cattani, and S. Y. Chen, "Viewing sea level by a one-dimensional random function with long memory," *Mathematical Problems in Engineering*, vol. 2011, Article ID 654284, 13 pages, 2011.

[6] M. Li, "Fractal time series—a tutorial review," *Mathematical Problems in Engineering*, vol. 2010, Article ID 157264, 26 pages, 2010.

[7] Z. Liao, S. Hu, D. Sun, and W. Chen, "Enclosed Laplacian operator of nonlinear anisotropic diffusion to preserve singularities and delete isolated points in image smoothing," *Mathematical Problems in Engineering*, vol. 2011, Article ID 749456, 15 pages, 2011.

[8] X. You and Y. Y. Tang, "Wavelet-based approach to character skeleton," *IEEE Transactions on Image Processing*, vol. 16, no. 5, pp. 1220–1231, 2007.

[9] W. Zhang, Q. M. J. Wu, G. Wang, X. You, and Y. Wang, "Image matching using enclosed region detector," *Journal of Visual Communication and Image Representation*, vol. 21, no. 4, pp. 271–282, 2010.

[10] J. Huang, X. You, Y. Y. Tang, L. Du, and Y. Yuan, "A novel iris segmentation using radial-suppression edge detection," *Signal Processing*, vol. 89, no. 12, pp. 2630–2643, 2009.

[11] P. L. Palmer, J. Kittler, and M. Petrou, "An optimizing line finder using a hough transform algorithm," *Computer Vision and Image Understanding*, vol. 67, no. 1, pp. 1–23, 1997.

[12] H. Kato, K. T. Tan, and D. Chai, "Development of a novel finder pattern for effective color 2D-barcode detection," in *Proceedings of the International Symposium on Parallel and Distributed Processing with Applications (ISPA '08)*, pp. 1006–1013, December 2008.

[13] R. Grompone Von Gioi, J. Jakubowicz, J. M. Morel, and G. Randall, "LSD: a fast line segment detector with a false detection control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, Article ID 4731268, pp. 722–732, 2010.

[14] J. B. Burns, A. R. Hanson, and E. M. Riseman, "Extracting straight lines," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 4, pp. 425–455, 1986.

[15] A. Desolneux, L. Moisan, and J. M. Morel, "Meaningful alignments," *International Journal of Computer Vision*, vol. 40, no. 1, pp. 7–23, 2000.

[16] A. Desolneux, L. Moisan, and J. M. Morel, "Computational gestalts and perception thresholds," *Journal of Physiology Paris*, vol. 97, no. 2-3, pp. 311–324, 2003.

[17] W. Fuchao, *Mathematical Method in Computer Vision*, Science Press, Beijing, China, 2008.

[18] C. Steger and M. Ulrich, *Machine Vision Algorithms and Applications*, Tsinghua University Press, 2008.