A method for fast detection and decoding of specific 2d barcodes

Ovidiu Pârvu, Andrei G. Bălan

Abstract — **This paper addresses the problem of barcode detection and decoding on devices with limited resources. We propose a fast and robust method specifically designed for 2D Data Matrix type barcode scanning. Our approach uses a contour tracing technique for the identification of key corners and segments, followed by one or two fitting steps. The method is well suited for implementation on mobile hand-held devices or on other systems where a real-time solution is required.**

Keywords — **2D barcodes, barcode detection , Data Matrix, image processing, mobile devices.**

I. Introduction

Nowadays, when almost everyone is connected via hand-held mobile devices in a "technological synesthesia" and important informational paradigms merge, the general use of decoding barcodes becomes widely available. Barcode technology is accessible to anyone who possesses a simple camera featured device. Although specific industrial barcode reading hardware has already been developed and used for many years, cheap camera-enabled gadgets allow small businesses to take advantage of visually encoded data at almost no cost with minimal hardware investments.

Modern barcode representation migrates towards 2-dimensional data scattering. Different encoding symbologies have been designed, such as: QRcode, Data Matrix (semacode), colorcodes (High Capacity Color Barcode), maxicode, shotcode, etc. some of them heavily used in diverse applications. Despite the ever increasing rate of hardware capabilities, there is still a need to keep processing power and resource consumption under strict control. In the following sections we will strictly cover the detection and decoding of barcodes considering specific Data Matrix [1] format attributes.

II. 2D barcodes decoding

Data Matrix "Fig. 1.a" is one of the most widely spread 2D symbologies alongside the QR Code. Elements that best describe the main characteristics of a Data Matrix symbol are closely related to its finder pattern "Fig. 1.b". An L-shaped boundary formed by two adjacent sides helps in finding the location of possible symbols. It also provides the orientation and code size estimations on the source image. On the complementary sides, two alternating synchronization patterns lie. Encoded bits are distributed across a regularly spaced cell grid with its size given by the synchronization pattern. Further defining the symbol, the outer elements are accompanied by a "quiet zone", where no other graphic elements should be found. White on black barcodes are common with colored [2] variations also possible due to the decoding algorithm's focus on high contrast zones and grayscale conversions.

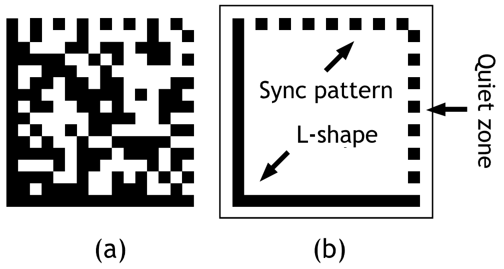

Fig. 1. (a) Data Matrix Code (b) Finder patern

Different approaches [3]-[5] were proposed in order to detect and decode barcodes with common hardware. Some solutions are general as they are for the most part applicable to a wide range of symbologies. Others focus on a specific barcode type. These methods are somewhat "faster" than their "general" counterparts as they tend to exploit characteristics of the symbology they try to decode.

Many methods rely on edge detection passes. These, in turn, point to different corner isolation techniques. Linear features used in object candidates retrieval are more than often highlighted through Hough space transformations. It is not an ideal practice for all mobile gismos since the method is memory hungry and pretty expensive computationally wise. Other solutions revolve around the identification of common gradient feature regions and growing / linking algorithms. Several texture analysis and segmentation techniques can also be used to detect candidate regions.

III. A fast barcode detection method

The guiding ideas implied by the development of our fast, computationally inexpensive solution were: reduction of large array memory accesses (process as few individual pixels as possible), the restriction to integer operations due to hardware limitations and possible speed optimizations. Our method follows classical image processing steps with a binarization / object candidates segmentation approach since we don't stray a lot from a basic scan-line, row by row, image analysis algorithm.

Independent components are identified with a contour tracing algorithm. Without disregarding noise and other

Pârvu Ovidiu is with the "Politehnica" University of Timisoara, Romania, Faculty of Automatic Control and Computer Science (e-mail: parvu@cs.upt.ro).

Andrei G. Bălan is with the "Politehnica" University of Timisoara, Romania (e-mail: andrei@ufodesign.ro).

errors, the longest two segments (linear or curved) are determined by contour corner analysis. If they pass certain criteria, the two segments could form the support for the L-shape marker. The code's corners are determined through two consecutive fitting and re-fitting steps starting from the maximal allowed fourth corner deviation relative to the already found segments. Next, we will introduce the general detection process followed by a somehow detailed discussion of its key algorithms:

1. *Pre-process source image (thresholding step).*
2. *If any remaining unprocessed regions exist then continue to step 3, otherwise go to last step.*
3. *Select region and trace region contour.*
4. *If contour acceptance test failed go to step 2, otherwise continue to step 5.*
5. *Select best two candidate segments corresponding to the current region.*
6. *If the two segments can't form a marker (L shape part of a barcode) go to step 2, otherwise continue to step 7.*
7. *Approximate the fourth corner (worst case).*
8. *Arc-circle fitting method.*
9. *If fitting failed then go to step 2, otherwise continue to step 10.*
10. *Optionally refit region / the four quadrilateral segments.*
11. *Find synchronization pattern. Approximate synchronization midpoints coordinates.*
12. *If synchronization pattern doesn't exist go to step 2, otherwise continue to step 13.*
13. *Extract binary pattern information based on midpoint positions under perspective.*
14. *Decode barcode region information and add to result queue. Go to step 2 (searching for another barcode).*
15. *Process result queue. End.*

A. Image pre-processing preserving relevant markers

This step prepares the original image “Fig. 2.a”, making it suitable for further processing. In order to eliminate unwanted background and accentuate important code features, a process of image segmentation is implemented in a binarization / thresholding step. This is achieved by applying different methods: histogram based (Otsu) [6] “Fig. 2.b”, adaptive, knowledge based, etc.

In our context of fiducial highlighting, adaptive thresholding produces very good results, being almost impervious to different degrees of illumination. A simple and efficient algorithm was proposed by Wellner [7], with the purpose of obtaining local threshold values by the mean of the last K linear pixels “Fig. 2.c”.

Our approach “Fig. 2.d” uses only a slightly modified box-blur adaptive threshold. We pass thru the image with a box window of K pixels around the current one (2*K+1 by 2*K+1), calculate its mean value and compare it to the pixel’s intensity. If we scan the rows horizontally, the only difference from the last box window are two vertical columns of 2*K+1 size at the left and the right of the new window. Then, if we retain the value of the first column, we only have to read one more column. Now, looking from a vertical perspective, the same observation applies for a pixel segment of 2*K+1. Thus for each pixel we only need to read its corresponding value and pre-calculate only one column value (two pixel reads, one column read, one column write) independent of the window size. The selected box window size is crucial to the detection step, even though the size doesn’t affect computation time, due to the algorithm used.

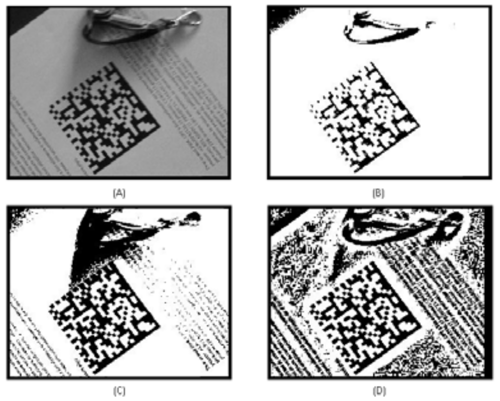

Fig. 2. (a) Original grayscale image (b) Histogram thresholding (c) Scan-line adaptive (d) Local adaptive

Although this method produces a lot more “noise” than the ones mentioned before, the object of interest is not affected as it can be very well distinguished from the background. The algorithm we used is very robust but it is still rarely susceptible of far then ideal results under certain illumination circumstances.

$$G(x,y)=\frac{1}{(2k+1)^2}\sum_{i=x-k,\,j=y-k}^{x+k,\,y+k}P(i,j)\cdot\theta \qquad (1)$$

As an improvement to this step, we could combine it with a global thresholding method, creating a hybrid solution. In our algorithm the global variance will be implemented as a theta (1) coefficient multiplied with the adaptive threshold, where G is the computed thresvalue for the [x,y] pixel under global variance for a window of size 2*k+1. By employing a hybrid solution, we have to consider whether the computation time in the code detection procedure is reduced or if it just introduces the inherent thresholding overhead.

B. Shape extraction based on contour tracing segmentation

Limited resource use was the criteria we based our detection solution on. Shape extraction and component isolation had to be done in a fast, reliable way [8]-[11]. We started from the algorithm suggested by F. Chang and C.J. Chen [12] as it was designed for fast row by row scanning with the associated contour deviations. The algorithm deals with the connected component labeling problem but it also generates contour information.

The algorithm scans each binary image row for unlabeled critical pixels. If a boundary unprocessed pixel is found the method diverges from the scan-line, sequential approach by tracing and marking the whole new-found contour. Processing time is directly dependent on the number of traced contours and indirectly on the implemented binarization strategy. Since we're not interested in component labeling we opt to mark only the contour and its adjacent pixels. Inward component pixels are not of our concern. In doing so, we save image memory writes as there is no need to label these pixels. Our basic detection method doesn't actively use interior contours hence gaining important speed-ups.

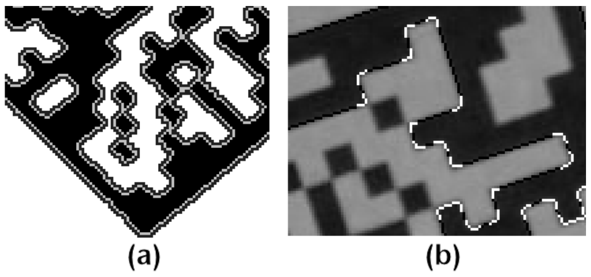

Fig. 3. (a) Contour tracing (b) Corner neighborhood

The boundary is determined on a consecutive pixel tracing basis. Once the first contour pixel is found, the next one is detected by analyzing the current pixel's neighbors in a clockwise manner and so on... The neighboring pixels are numbered similarly to a Freeman chain-code element [13]. Tracing direction is established starting from the last pixel's corresponding number. Shape outline "Fig. 3.a" is extracted in a continuous way, so that the data is already suitable for further processing.

When a contour is closed, some simple contour evaluation tests are taken with regard to its maximum allowed length, bounding box size / ratio limits and bounding box position. We found out that such a trivial approach is highly efficient to the removal of most of the unwanted noise and image objects.

C. Detection of corner neighborhood points

Once a specific object boundary is found, its contour pixels are covered in a continuous way. Notice that the main characteristics of an "L-shaped contour" are defined by its longest two line segments. Different contour corner techniques were suggested [14]-[16] but we stopped on the method proposed by D. M. Tsai et al.[17] which is based on the eigenvalues of the covariance matrix of data points. For each pixel in a sequence of boundary points two proper eigenvalues associated to the minor and major axis of an ellipse could be calculated. A symmetric region of support is analyzed for every contour point. Those values are then compared to an estimated threshold. A pixel could be considered as a contour corner by verifying the eigenvalue criteria. Corner neighborhood "Fig. 3.b" pixels are also selected when proper threshold values are applied.

The method is very fast as it and can be easily integrated into the contour tracing algorithm. New unprocessed contour pixel coordinates are added to the region of support in a sliding window manner so that we actually calculate the eigenvalues for an already traced contour pixel. Sliding window size has no influence whatsoever on the performance of the corner detection method.

The two L-marker segments are identified by selecting the longest two continuous contour sections without corner or near-corner points. For the detection of severely damaged codes a simple segment merging routine is implemented. Another object rejection phase is carried on right away. Corner-based rejection tests could be carried out while "walking" the boundary, which consequently leads to the exclusion of further corner detection.

D. Fourth corner approximation and fitting

In the last carried step, preliminary L-shape coordinates were found. The strongest L-shape corner location can be closely approximated starting from the intersection of the two recently extracted segments. Next, the remaining three points are classified and labeled with A_1,B_1,C_1 "Fig. 4." in order to obtain the code's orientation. A_1 and C_1 are located "inside" the AB1 and CB_1 segments, where A and C are accurate barcode extremities. The corner detection method ensures us that A_1 and C_1 are extremely close to A and C points. We define A_2 and C_2 as two pixel locations which are certainly found on the opposite barcode segments but also on the L-shape boundary. As A_2 and C_2 could be determined in a simple way starting from A_1 and C_1, they remain in the close neighborhood of A and C points. Note that A and C are not yet determined.

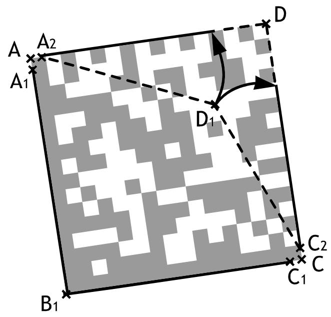

Fig. 4. Arc-circle fitting

The most difficult problem that remains to be solved is finding the coordinates of the fourth corner (D). Our first naive attempt was to apply an inverse perspective transformation under weak-perspective assumptions. The position of the fourth corner is one of the solutions to a specific quartic equation. Finding these solutions analytically is not quite trivial. This endeavour proved to be less than appealing as the exact coordinates of A and C weren't precisely determined. We finally opted for a barcode quadrilateral fitting step and introduced the D_1 point as a bad approximation to the fourth corner. As a mandatory requirement, D_1 has to be located inside the barcode area. We estimated the location of D_1 based upon the median of the A_1C_1 segment and the bisector of the $A_1B_1C_1$ angle. The A_2D_1 and C_2D_1 segments are the sources of two imaginary circles with the centers in A_2 and C_2. For each pixel found on circumference its corresponding previous or next circle neighbors could be rapidly computed with a fast "midpoint arc-circle" drawing routine. Starting from D_1, the arc-circle drawing algorithm moves towards the exterior of the barcode region. As long as black pixels are found on the "outside" of the two oriented segments the next arc-circle pixel is considered and so on. Searching stops when the arc-circle reaches the quiet zone. A small tolerance value has to be introduced in order to ignore possible noise, small distortion and image binarization errors. The method facilitates fast convergence towards the outside boundary so that correct estimation of D_1's position doesn't play a crucial role in it. Once this process is completed, two new boundary points are found. The intersection of the two new found segments gives us a decent approximation of the D point. A and C are calculated by intersecting segments A_2D with A_1B_1 and C_2D with C_1B_1, respectively.

The last calculated point positions might be used with satisfactory results in the Data Matrix cell sampling steps. Since there is still room for improvements, we use a second, optional, segment fast re-fitting step. The vicinities of the already known segments are sampled by "climbing" (relative to the analyzed segment) when black pixels are encountered and "descending" when white pixels are found. Black pixel peaks are used to generate new line equations with a linear regression approach.

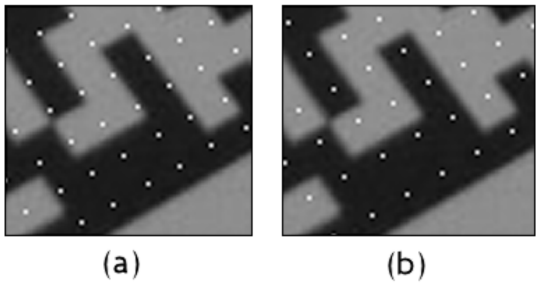

Fig. 5. (a) Bad case of perspective sampling cell centers (b) Synchronization midpoint with perspective sampling

Sampling this grid only by perspective projection means gives moderate results "Fig. 5.a". Our tested implementation takes into account the fact that information about synchronization pattern midpoints is calculated in a previous pass "Fig. 5.b". Midpoint information, although incorrect perspective-wise, alongside correctly transformed coordinates makes for a better solution.

IV. Variations and Improvements of the Proposed Method (Future Work)

As a speed optimization, our algorithm processes only the exterior object boundary for barcode information. This does not favor the inverse contrast codes. To face this drawback, the same method may be used by taking into account the inner contour. In the case of extremely damaged barcodes (highly disconnected) another step for component merging can be applied after contour tracing. The use of curve/arc equations instead of lines would also be a significant upgrade to the decoding algorithm, easily correcting the errors induced by camera lens distortions.

Taking into account the chosen application environment (mobile devices) we implemented our algorithm by using only integer operations. Decoding precision is improved by using floating point operations. Generally, this accounts for intersection calculus, which is mainly reflected in sampling point coordinates. However this usually claims a great deal of resources, especially on devices with limited specifications. Following the pre-processing steps we only need two bits to store the information for each pixel. In the 2 bit per pixel approach, memory use is greatly reduced (but this could also induce unnecessary computations).

In case of a real-time oriented implementation, another algorithm could be merged into our proposed method by tracking the barcode region in a continuous image stream. Tracking would improve the detection and decoding part of our method, since it would process a higher number of input images for better region sampling. However, this is not mandatory considering that our method could already be implemented on a real-time system.

V. Conclusion

In this paper, we proposed a fast and robust solution for the detection and decoding of 2D barcodes, specially suited for operation on devices with limited resources. The featured method was tested and provided a satisfactory output, containing a high decoding accuracy mixed with a fast response time. Based upon the optimizations discussed earlier, we will further develop this method increasing its precision and making it more "time-critical".

References

[1] "*Data Matrix bar code symbology specification*" ISO/IEC 16022:2006, 2006
[2] D. Parikh, G. Jancke, *"Localization and segmentation of a 2D high capacity color barcode"*, Proceedings of the 2008 IEEE Workshop on Applications of Computer Vision, 2008, pp. 1-6
[3] E. Ohbuchi, H. Hanaizumi, L. Ah Hock, *"Barcode Readers using the Camera Device in Mobile Phones"*, Proceedings of the 2004 International Conference on Cyberworlds (CW'04)
[4] H. Wang, Y. Zou, *"Camera readable 2D bar codes design and decoding for mobile phones"* , ICIP 2006, pp. 469-472
[5] E.Ottaviani, A.Pava, M.Bottazi, E.Brunclli, F.Casclli, M.Guerreo, *"A common image processing framework for 2D barcode reading",* 7th International Conference on Image Processing and its Applications, pp. .652–655
[6] N. Otsu, *"A threshold selection method from gray-level histograms"*,IEEE Transactions on Systems, Man, and Cybernetics,pp. 62-66, 1993
[7] P. D. Wellner, *"Adaptive Thresholding for the DigitalDesk"*, Xerox Technical Report vol. EPC-1993-110, 1993
[8] I. Debled-Rennesson, S. Tabbone, L. Wendling, *"Multiorder polygonal approximation of digital curves"*, Electronic Letters on Computer Vision and Image Analysis, 2005,pp. 98-110
[9] A. Mikheev, L. Vincent, V. Faber, *"High-quality polygonal contour approximation based on relaxation",* Proc. International Conference on Document Analysis and Recognition, 2001
[10] D.G. Bailey, C.T. Johnston, *"Single pass connected components analysis"* , Proceedings of Image and Vision Computing New Zealand, 2007, pp. 282–287
[11] S. Kiranyaz, H. Liu, M. Ferreira, M. Gabbouj, "A*n efficient approach for boundary based corner detection by maximizing bending ratio and curvature",* Signal Processing and Its Applications, ISSPA 2007, pp. 1-4
[12] F. Chang, C. J. Chen, *"A Component-Labeling Algorithm Using Contour Tracing Technique"*, Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)
[13] F.C.A. Groen, P.W. Verbeek, *"Freeman-Code Probabilities of Object Boundry Quantized Contours"*, Computer Graphics and Image Processing 7, pp. 391-402 (1978)
[14] E. Rosten, T. Drummond, *"Machine learning for high-speed corner detection",* Computer Vision – ECCV, 2006, pp. 430-443
[15] Z. Hao, S. Lejun, *"A fast corner detection algorithm based on area deviation",* Proceedings of *IAPR* Workshop on machine vision applications, 1994
[16] F. Arrebola, F. Sandoval, *"Corner detection and curve segmentation by multiresolution chain-code linking",* Pattern Recognition, No. 10, 2005, pp. 1596-1614
[17] *D. M. Tsai, H. T. Hou, H. J. Su, "Boundary-based corner detection using eigenvalues of covariance matrices",* Pattern. Recogn. 20, 1998,pp. 31-40