A COMMON IMAGE PROCESSING FRAMEWORK FOR 2D BARCODE READING

E.Ottaviani, A.Pavan [1], M.Bottazzi, E.Brunelli, F.Caselli, M.Guerrero [2]

(1) Elsag S.p.A. Italy - (2) Datalogic S.p.A. Italy

The present work describes an image processing system able to locate, segment and decode the most common 2D symbol used in applications. The different symbol are treated exploiting their similarities, in order to achieve an unified computational structure.

INTRODUCTION

Until today the most important optical machine readable symbols have been the 1D barcodes. This is a well-known and established technology but it shows some limitations in terms of quantity of information stored in a symbol and error correction capability. Also the intrinsic robustness of the bar-code symbols is not very high, because many codes do not have any check capability and just some of them make use of a check digit that reduces but does not eliminate the risk of misreading.
To overcome these limitations many 2D symbols have recently appeared, in which the optical readable information is stored using the whole area occupied by the symbol. They offer many advantages:

- it is possible to encode up to several thousand characters of machine readable data so they can be viewed as a fully portable data file containing all information required for automatic identification;
- if only a short information is required it can be stored in a very small area in which a 1D barcode is not suitable (e.g. silicon wafer, electronic components, small pharmaceutical items, ...);
- they have an error detection and correction structure increasing the reliability and robustness of the identification.

Many different 2D symbols have been developed, and they are all competing to gain a dominant relevance in different applications. The international standardization process is in a really early phase and today it is not possible to foresee which symbols will be successful. Though each 2D symbol has its own morphological structure, different symbols may be grouped in three main classes:

- multi-row (or stacked) codes, in which a set of linear bar-codes are stacked together in a single, multi-row symbols. A typical example is the PDF417 [1];
- 2D codes with a locating target, in which a special pattern is used to locate the symbols against a complex or unknown background. Two typical examples are the Maxicode [3] and the QR-code [4];
- 2D codes without a locating target, in which the locating must exploit the internal structure of the code itself. A typical example is the Data Matrix. [2].

The Fig.1 shows code patterns of the four examples cited above. Many other code types are currently in use, but these ones seem the most important for each class.

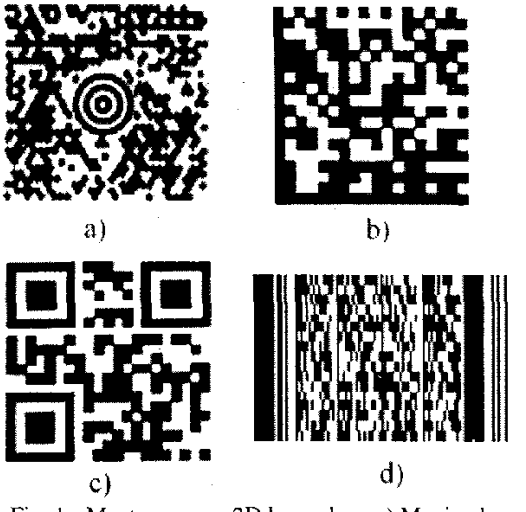

Fig. 1 - Most common 2D barcodes - a) Maxicode, b) Datamatrix, c) QR-code, d) PDF417

THE GENERAL READING PROBLEM

The problem addressed in this paper is the real-time decoding of these 2D symbols. Real time really means different things depending on the system: in the case of an attended system, in which a human operator is using the reader, the term real-time means to have a response time satisfying the feeling of the operator, e.g. a decoding time of about 500 msec is acceptable.
The process of decoding a 2D barcode is a very complex task requiring sophisticated image processing algorithms. The typical steps required to decode a symbol are:

- to acquire a gray levels image containing the symbol: this can be made by means of 2D CCD camera or with a linear CCD in the case of moving targets.
- to locate the symbol inside the image: this is often the slower, time consuming, processing activity because of the large size of the image and the fact that the type, number and resolution of the symbols present in the image and the context of the image

itself are typically unknown. Its goal is to find all the possible region of interest that could contain a symbol. Location must be robust against: noise, distortions, interference, scaling and rotation.

- to segment the symbols from the region of interest found in the previous step. (e.g. to extract the boundaries of the code in order to evaluate the gray intensity and the co-ordinates of each point of the mesh composing the symbol).
- to decode the symbol, classifying the features extracted using the syntactic rules of the symbol and translate the information in an ASCII string. This step often include an error detection and correction technique (e.g. Reed Solomon Algorithms are often used) that requires a computation time rapidly increasing with the correction capability.

The difficulties to reach high reading performance can be summarized as follows:

- the variable resolution of the symbols inside the image, due to the variable distance between the target label and the reader machine and to the different printing resolution, symbols may appear in the image with very different resolution (a magnification factor of 8 should be considered).
- the geometrical distortions under which the symbols may be found on the image, due to the lack of orthogonality in the image acquisition phase.

In this paper we propose a quite general image processing approach suitable to locate, segment an decode the most common true 2D symbols. That means we focus our attention on codes like Maxicode and Datamatrix.
They share some geometrical properties like:

- they can be surrounded by a 4-side polygon (and a sufficiently large quiet zone);
- they are characterized by the presence of a stable pattern composed by simple geometrical primitives (segments and arcs).

THE IMAGE PROCESSING APPROACH

The image processing approach is composed of four main steps:

1. Regions of interest (ROI) detection
2. Code location
3. Code segmentation
4. Decoding

The first step selects sub-images which are likely to contain the codes. The second one locate the distinctive feature of each code (i.e. the "target"), then the third performs an accurate detection of the code boundary. Finally, the fourth one allows the computation of the equivalent ASCII string.

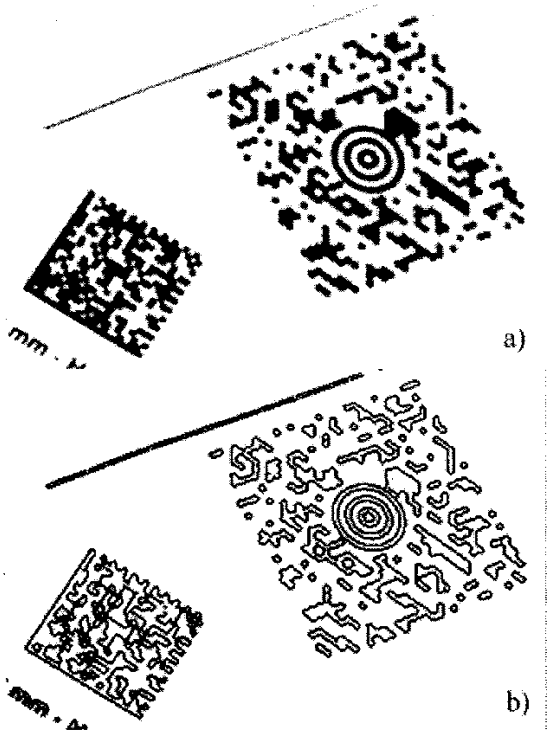

Fig.2 - Typical code image: a) original b) edges

ROI detection

The approach is based on a common image representation in term of magnitude and phase of the image gradient. By this representation, we can select a few region of interest (ROI) in which the gradient shows some directional characteristics. In order to speed up the computation, magnitude and phase of the gradient G are evaluated with 3x3 Sobel masks for G_x and G_y and suitable look-up tables for square root and inverse tangent operators. In order to select a proper threshold for the magnitude, its RMS value G_m is also computed.
The ROIs are generated by an image tassellation with a fixed block size L. Such size must be small enough to ensure a good selection of ROIs and large enough to have statistical meaning. With typical images in which code size is from 100 to 400 pixels, a block size of 32 seems adequate.
In each block the points with gradient magnitude greater than G_m are selected and their directional histogram is evaluated. A strongly modal histogram suggests the presence of 1D barcode or PDF, while the occurrence of two orthogonal modes suggests the presence of a Datamatrix of QR-code. Otherwise, the Maxicode is suggested when histogram is nearly uniform.
The ROIs are then computed by grouping connected blocks sharing the same directional properties. Then the groups are labeled as candidate 1D or 2D codes, but only the true 2D ones are the subject of the following steps.

Code location

Code location is performed by a standard edge detection and linking approach, in which edge points are grouped in chains (see Fig.2). In order to get exactly 1D chains, non-maxima suppression and morphological thinning operators are applied to the thresholded magnitude gradient image [5]. Then the chains are piecewise fitted by segments or elliptic arcs. The list of segments and arcs found in a given region of interest are the base for the locating process, which selects segments or arcs configurations according to a given code. Here are the geometrical rules used to located the different codes:

- Maxicode: find a point which is centre of several concentric arcs;
- QR-code: find three points which are intersection of the axis of several parallel or orthogonal segments;
- Datamatrix: find a point which is the intersection of two nearly orthogonal long segments.

This last stage of the location process must be necessary dependent on the code to be searched in order to ensure high performances, exploiting all type of constraints imposed by both code specifications and imaging geometry.
Finally, a set of candidates points is produced and each one is evaluated with a confidence factor taking into account the number and the length of primitives voting for each point. It is assumed that each ROI contains only one 2D code of each type, so only the best candidates for each type are selected.

Code segmentation

Starting from each candidate point, the segmentation process try to extract the code boundary. Let us note that the code boundary is a typical instance of subjective contour, because it does not correspond to any contour in the image.
Code segmentation can be carried out in a general way. The basic idea is to start form a small area (seed) that must be inside the code and to modify its boundary in order to include all code points. The shape of the seed area depends on the code to be found (e.g. it is a circle for the Maxicode and a triangle for the Datamatrix).
The segmentation task is performed by an alternate sequence of region growing and convex hull evaluation (see Fig.3).
Region growing modifies the boundary of the region in order to include connected pixel with grey level lower than an adaptive threshold T, evaluated over a small area centred around that pixel. Adaptive thresholding is necessary in order to deal with variable lighting conditions. We adopted a modified version of the Niblack algorithm [6], in which the averaged grey level of a neighbourhood of a given point is used to set a threshold for that point. This approach works well because all codes are balanced in terms of black/white ratio, and the only deviations come from lighting conditions and code printing quality.

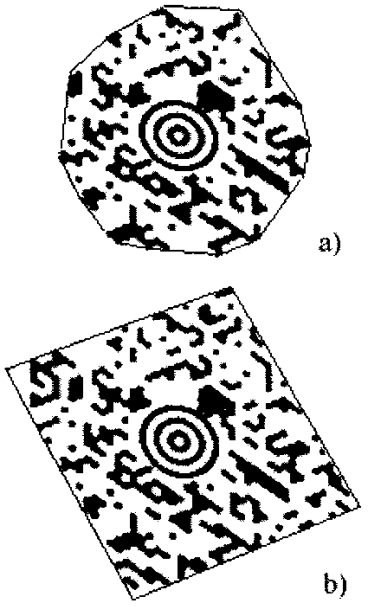

Fig. 3 - Convex expansion: a) intermediate region and convex hull, b) final region and 4-side polygon

The convex hull computation [7] ensures the segmentation of the whole code pattern even if it is not a connected one. This is specially important for codes composed by sparse dots (e.g. Maxicode). The convex hull allows the detection of all "black" points inside the boundary which are not connected to the code region. They are the seeds for the following growing step.
The process stops when no more points inside the convex boundary can be selected or if the convex hull does not change after the iteration. Usually two iterations are sufficient for compact codes (e.g. Datamatrix), while up to four iterations are needed for the sparse ones.
Finally the smallest 4-sided polygon surrounding the segmented code is evaluated. This is a special case of convex hull evaluation, in which the hull is constrained to have exactly four sides. The output consists in a list of four points which are the vertices of a general polygon surrounding the code.

Decoding

The area inside the polygon is then meshed, resampled and binarized in order to compute the bit pattern of the code. The mesh is evaluated by exploiting code characteristics and keeping in account that codes may be rotated, scaled and distorted with respect to a given prototype. Affine group transformations are used in order to define the allowed deformations.
Finally the nodes of the mesh are evaluated in order to define their logical value. The resulting bit pattern is checked in order to find the equivalent ASCII string, exploiting the error correcting mechanisms embedded into the code specifications.

RESULTS AND COMMENTS

The process ensures real-time decoding of 2D symbols even with bad illumination conditions and with strong perspective deformations. Tests on real images show that codes are still readable even at 45 degree from the vertical axis. Image resolution is a key issue. At least 3 pixel for the smallest element of the code (the "modulus") are needed in order to get an answer.
Software implementation does not require any special device but can be done on low cost CPUs (e.g. Pentium 233 MHz). Processing time on typical VGA images (640x480 pixels) ranges from 200 to 400 msec, depending on the size and the type of the code. Let us note that such result is already compatible with the typical user requirement for a handheld device. In fact, an automatic reading device must have a response time comparable with the reaction time of a human operator.
At least one half of the processing time is due to image pre-processing (gradient computation). Dealing with larger images, a multiresolution approach may be adopted, in order to find ROIs at a lower resolution and to process them at a higher one.
The approach is suitable to be extended to many more 2D symbols sharing similar geometrical properties. That means it is possible to decode new symbols only adding proper functions for spatial reasoning and decoding.

CONCLUSIONS

The paper presents a general image processing architecture in order to locate and decode 2D symbols like Maxicode, QR-code and Datamatrix. The approach ensures high reading performances even under bad lighting conditions and strong perspective deformations. Results show the process is already suitable of real time implementation.
The work has been supported by the EEC funded project BBC (Bidimensional Bar Codes), developed under the ESPRIT Technology Transfer Program and coordinated by the Technological Transfer Node NOTSOMAD. Achieved results have been jointly patented by Elsag and Datalogic.

REFERENCES

1. AIM, 1994,"Uniform Symbology Specification: PDF417"
2. AIM,1997,"International Symbology Specification: Data Matrix"
3. AIM,1996,"International Symbology Specification: Maxicode"
4. AIM,1997,"International Symbology Specification: QR-Code"
5. R.Jain, and R.Kasturi, 1995, "Machine Vision", Mc-Graw Hill, New York
6. J.Sauvola et al. , 1997, "Adaptive document binarization", Proc. IEEE, pp.147-152
7. F.Preparata, and M.Shamos, 1993, "Computational Geometry", Springer Verlag, New York