A GRAYSCALE READER FOR CAMERA IMAGES OF XEROX DATAGLYPHS

K.L.C. Moravec
Xerox Research Centre Europe 6 chemin de Maupertuis
38240 Meylan, France
kimberly.moravec@xrce.xerox.com

Abstract

Xerox DataGlyph technology is a 2-D matrix barcode encoding which combines relatively high data density with an error correction scheme and an unobtrusive format. Some of the more exciting applications proposed for this technology involve the decoding of unconstrained camera images of DataGlyphs. This paper details a set of grayscale methods including steered gradient filters and the Radon transform which can decode many unconstrained camera images. This method has been successfully tested on nearly 200 camera images with very good results.

1 Introduction

Machine readable tags such as barcodes have been essential for inventory management, package tracking and process control in many industries such as manufacturing and retail for decades. Recently, researchers in the area of ubiquitous computing have used hand-held barcode readers as an inexpensive and robust way to bridge the gap between virtual and real worlds [9, 11, 14]. Hand-held digital cameras and webcams coupled with barcodes are also used in augmented reality desktops [10, 16]. These intriguing new applications for barcodes motivated the development of a camera-based reader for DataGlyphs.

The DataGlyph, a type of 2-D matrix barcode, was developed at Xerox's Palo Alto Research Center [6, 7]. It can encode more data than a 1-D barcode and has an extremely high data density for a 2-D barcode. It appears to a human as a grey halftone that is easily integrated into a document's design. It has been used in products such as Xerox Flowport™ and is an active area of development [1]. This paper describes robust methods of decoding DataGlyphs from images taken by a hand-held camera or webcam, extending the range of applications for which DataGlyphs can be used.

This introduction explains in detail the DataGlyph format and the unique challenges of decoding camera images of DataGlyphs. The paper then describes the proposed decoder and the methods used in it. Results of the new decoder are compared with a binarization and correlation-based method, followed by the paper's conclusions.

[1]For current information, see www.dataglyphs.com.

BMVC 2002 doi:10.5244/C.16.68

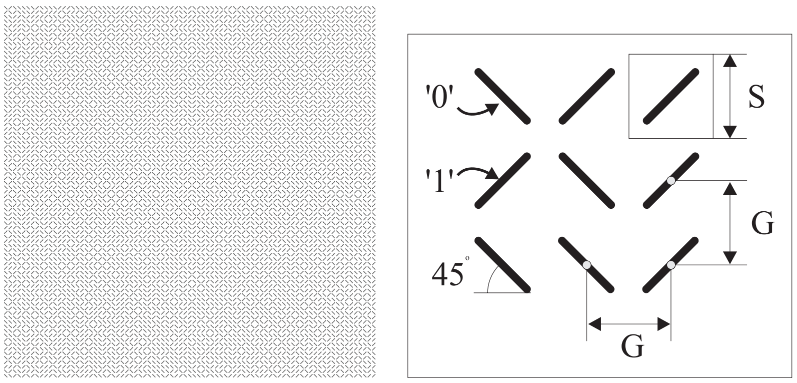


Figure 1: A 300dpi DataGlyph containing the text of the abstract (left) and a diagram showing the detailed structure of the DataGlyph (right). The glyph consists of individual 45° diagonal marks, forward slashes represent a binary '1' and reverse slashed represent a binary '0'. S is the size of the glyph mark, and G is the size of the inter-mark spacing.

1.1 The DataGlyph Format

A DataGlyph represents binary data on paper as a grid of small $\pm 45°$ lines known as *glyph marks* (shown in Figure 1). By printing at higher resolutions, the grid, known as a *glyph block*, appears to the human eye as an evenly textured gray rectangle, much like a halftone photograph. An example of this is shown in the left half of Figure 1.

To encode the data in a robust manner, the data is first converted using a Reed-Solomon code with two CCIR checkbits. It is then interlaced (to deal with burst errors, resulting from damage such as a paper fold or a mark on a glyph) and finally it is enmeshed in a *synchronization frame*.

Typical decoding of a DataGlyph consists of an image processing stage and a decoding stage. First, the skew and the scale of the glyph are determined using profiles at different angles (an accepted technique for finding skew in Document images [2, 13]). The decoder then binarizes the image using a locally adaptive threshold and performs correlation on the marks using a template generated from the scale and angle information. Building the block begins by finding the highest correlation mark and then 'walking' outwards, in mark-sized steps, to locate its neighbors. Applied recursively, this step eventually locates most of the marks in the block. Uncertainties are passed on to the decoder.

The decoding stage takes the binary data matrix found by the image processing stage and attempts to decode it by finding the synchronization frame, de-interlacing the data and then decoding the Reed-Solomon code. Because of the error coding, an incorrectly decoded message is almost impossible. If the image processing has failed to find a certain percentage of the marks correctly, the error coding spots this and returns a failure. For faxed and scanned images, this rarely happens: the combined steps are extremely successful and would be difficult to improve upon.

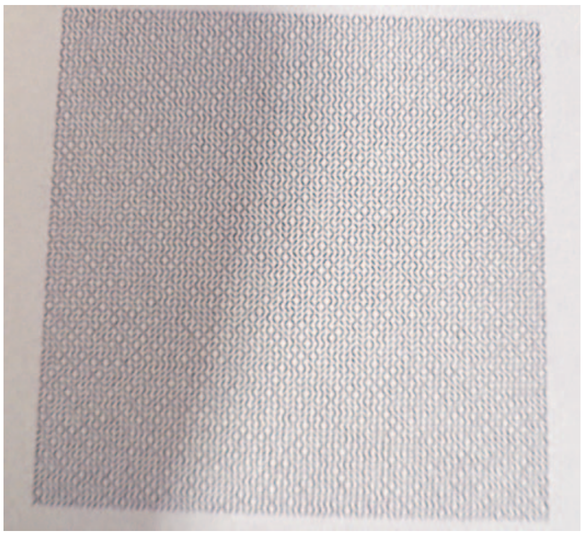


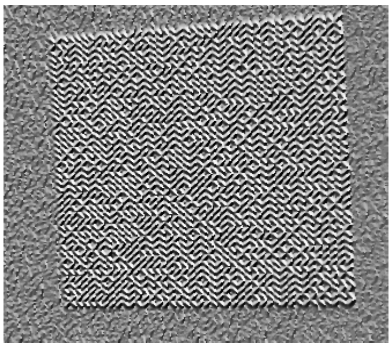
Figure 2: A camera image of the DataGlyph in Figure 1. The image contains lighting variations, blur and perspective warping.

1.2 Camera Images

When hand-held cameras or hand-positioned webcams are used to take images of DataGlyphs, the resulting images are different from scanned or faxed images of DataGlyphs, as illustrated in Figure 2. In scanned or faxed images, where the paper is a fixed distance from the CCD, it can be assumed that the glyph marks of a glyph printed at a particular resolution (say 600 or 300 dpi) will all have the same scale in pixels, whereas for camera images, the scale of the marks can vary over the image depending on the distance of the camera from the glyph. There are also often other features of camera images which make them difficult to decode: there is usually some camera noise, some blur, and there is likely to be some skew and perspective warping in the image. In addition, any decoding method must be able to decode images with uneven background lighting, a phenomenon which often occurs because of the shadow of the camera or a directed light source.

2 Decoding Methods

The camera decoder proposed in this paper uses methods which are fairly robust to unknown scale, lighting variations, noise, blur, skew and perspective. There is no binarization step; the proposed decoder uses the full grayscale information to perform its processing and produce a labelled glyph matrix in a specified format, which can then be decoded using standard algorithms. The proposed decoder consists of four main subroutines: *Mark Estimation*, where the overall orientation of the glyph marks is determined; *Block Estimation*, where the locations of marks in the glyph grid are found; *Integration*, which labels each mark location based on local angle information and *Decoding*, which uses standard algorithms to decode the glyph. A detailed description of each method is given below.



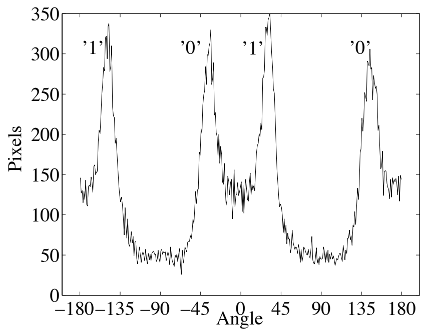


Figure 3: The angles of the glyph image (left) and the angle histogram (right) showing four peaks, two pairs each corresponding to '0' and '1' marks.

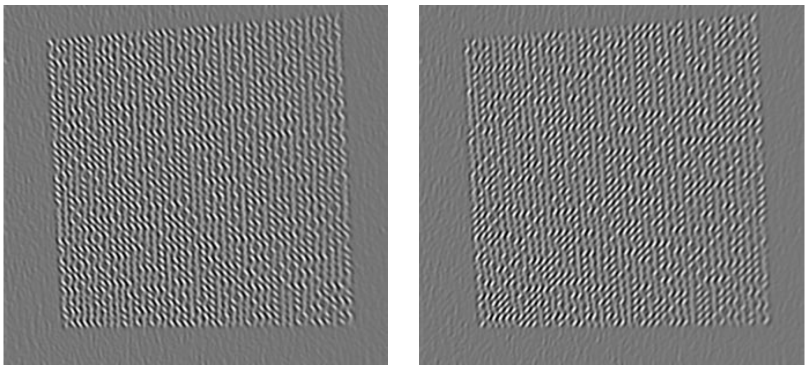


Figure 4: The final glyph images filtered for '0' marks (left) and '1' marks (right).

2.1 Mark Estimation

An orientation filter in the proposed decoder finds the rough orientation of the glyph for the Radon step and finds the glyph mark centers without the need to make assumptions about the scale or overall orientation of the image. For our purposes, a fast calculation of the gradient was desired. The filters chosen for the proposed decoder are 4×4 gradient operators [1]. The filters, f_V and f_H, find the gradient in the vertical and horizontal directions respectively. These particular gradient operators give a consistent response over all angles.

Given I_{f_V} and I_{f_H}, the images filtered by f_V and f_H, the orientation of each pixel can be derived by finding the maximum response of the filters [5] using the following equation:

$$I_\theta = \tan^{-1} \frac{I_{f_V}}{I_{f_H}} \tag{1}$$

$$|I| = \sqrt{I_{f_V}^2 + I_{f_H}^2} \tag{2}$$

$|I|$ is shown in the left half of Figure 3. From these images, a histogram of angles can be derived (as shown in the right half of Figure 3), consisting of two pairs of peaks separated by π radians. One pair corresponds to '0' marks at angle θ_0 and the other corresponds to '1' marks at angle θ_1. Steering the filtered images to each angle produces images with strong responses at that angle:

$$I_{\theta_0} = \cos\theta_0 I_{f_V} + \sin\theta_0 I_{f_H} \quad (3)$$
$$I_{\theta_1} = \cos\theta_1 I_{f_V} + \sin\theta_1 I_{f_H} \quad (4)$$

The original image is then filtered a second time at each angle to produce two image with peaks at the centers of '0' marks and '1' marks (shown in Figure 4). This corresponds to taking a directional second derivative which has local maxima at the centers of the glyph marks. The local maxima of both filters are combined into a single image (shown on the left of Figure 6). This image of the glyph centers is passed on to the next stage, Block Estimation.

2.2 Block Estimation

The proposed decoder assumes that the image of the glyph centers is a noisy grid under perspective. A grid under perspective can be modelled as the intersections of two pencils of lines [15] which can easily be found using the Radon or Hough transform [8]. The Radon transform, R, is given by

$$R(m, c) = \sum_x I(x, mx + c) \quad (5)$$

where $I(x, mx + c)$ is a pixel of the image I at column x along a line characterized by the slope m and y-intercept c. Here the transform is parameterized by slope rather than angle [4].

A Radon transform in the x-direction gives a sharp point response for each row in the image. Since the image of the glyph block, $I(x, y)$, is a projective transform of the original block image, $I(u, v)$, the two are related by the parameters p, q and r as follows [12]:

$$x = \frac{p_u u + p_v v + p}{r_u u + r_v v + 1} \quad (6)$$
$$y = \frac{q_u u + q_v v + q}{r_u u + r_v v + 1}. \quad (7)$$

Noting that the original glyph rows can be related to the image glyph rows by

$$y(u, v) = m(v)x(u, v) + c(v) \quad (8)$$

it is shown in [3] that

$$q_u = p_u m(v) + r_u c(v) \quad (9)$$

which is a linear relation between m and c for all v. Therefore, the responses for each row in the x-pencil will always be colinear in the Radon transform of the glyph centers image, as shown in the top left of Figure 5. A second Radon transform determines this line (shown in the top right of Figure 5), and a profile of the line (Figure 5, bottom) is retrieved for further analysis.

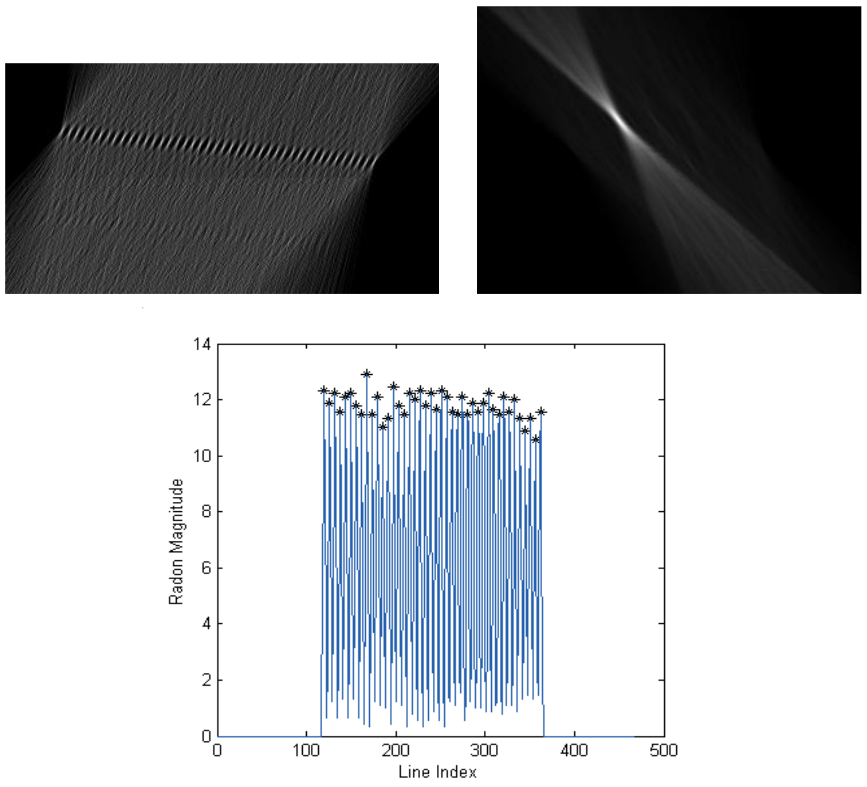


Figure 5: Block Estimation. The results of the first Radon transform (top right), showing the pencil of glyph marks as a line of peaks down the center of the image. The results of the second Radon transform (top left), showing the single peak corresponding to the line in the top right image. The peaks of the profile along that line (bottom), each peak corresponding to a row of glyph marks in the image.

The local maxima along the line correspond to rows of the glyph, but this line is often noisy due to stray marks and poorly located glyph centers. The valid local maxima are found by modelling the inter–glyph mark spacing by a Poisson distribution and mark magnitude by a Gaussian distribution; outliers are deleted.

The same method for finding the rows of the glyph block is used to find the columns of the glyph block. The intersections of the row and column pencils are the new, estimated centers of the glyph marks. The results of block estimation are shown in Figure 6.

Finding the glyph mark centers is an essential step of the proposed decoder . Although the Radon transform can be applied to the original grayscale image, the most distinct sets of linear features in these original images are the spaces between the marks and the marks themselves. The rows and columns of the mark centers are only apparent when the stronger line structure in the images has been eliminated

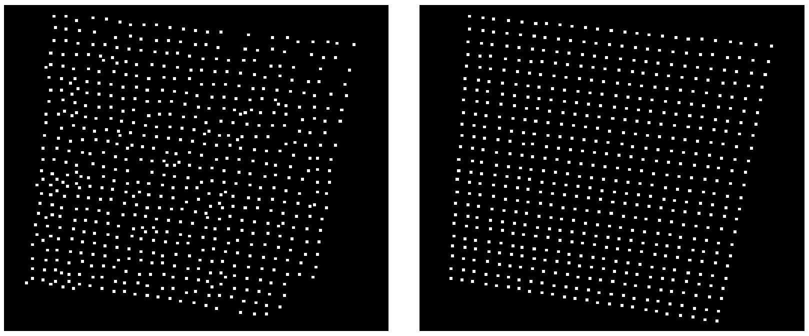


Figure 6: Estimated glyph mark centers before and after Block Estimation (left and right respectively).

2.3 Integration and Decoding

Given the corrected glyph mark centers, a matrix of the correct dimensions is formed. The major orientation of the steered images at the corrected centers becomes the label of the matrix. In cases where there is ambiguous information from the steered images, the matrix entry is labelled ‘2’, corresponding to the DataGlyph format label for uncertainty. The labelled matrix block of ‘0’s, ‘1’s and ‘2’s is decoded as per standard DataGlyph decoding.

3 Results

The proposed camera-image decoder described in Section 2 was tested against the current DataGlyph decoder described in Section 1.1. Although other camera-based decoders have been written in the past, the current DataGlyph decoder is the only method for which documentation or code is still available.

The two methods were tested on a database of 191 DataGlyph camera images. Ten glyphs of URLs were created for the database. A specialized stand was made which was at a height such that both the current and proposed decoders decoded glyphs reliably. For images with extreme noise and blur, twelve images of each glyph were taken and combined into an average image to remove camera noise. The blur images were then created from these noiseless images by filtering with a Gaussian kernel. Additive Gaussian noise was added to the noiseless images to create the noisy image database. For the resolution and perspective images, the stand was altered by placing 5mm thick shims under the stand base at the sides and corners.

Results are summarised in Tables 1 and 2. All figures are for decodes of ten images, with the exception of skew, where figures are for three images. Significant improvement to the decode rate for the current decoding methods can be seen when the proposed methods are employed. Within the parameters that it operates, the proposed camera decoder operates reliably, while the current decoder has variable performance even for small image degradations.

An unoptimized C prototype, running on a 733MHz Pentium III processor, has a processing time of 1.7 seconds for the largest glyph tested(67×67 marks).

Synthetic Degradation: no. decoded out of 10 imgs.					
	Blur			**Noise**	
σ	**Current Decoder**	**New Decoder**	σ	**Current Decoder**	**New Decoder**
0.5	6	10	**5**	9	10
1	4	10	**10**	9	10
1.5	0	8	**15**	6	10

Perspective Warping: no. decoded out of 10 imgs.				
	X Pencil		**Y Pencil**	
# **shims**	**Current Decoder**	**New Decoder**	**Current Decoder**	**New Decoder**
1	6	10	10	10
2	5	10	4	10
3	1	9	2	10

Resolution: no. decoded out of 10 imgs.		
# **shims**	**Current Decoder**	**New Decoder**
0	10	10
1	4	10
2	5	9
3	4	10
4	2	10

Skew: no. decoded out of 3 imgs.		
$\angle$	**Current Decoder**	**New Decoder**
0	3	3
5	3	3
10	2	3
15	3	3
20	1	3
25	0	3
30	0	3

Table 1: Tables showing the number of images decoded using the new decoder proposed in this paper versus the current decoder generally in use. Results are given for camera images corrupted by synthetic degradation, perspective warping, resolution and skew.

4 Conclusions

The novel aspects of the proposed decoder lie in the choice of methods for decoding camera images. The guiding principle was to choose methods which were invariant to camera image degradations as opposed to calculating those degradations explicitly and correcting for them. By using an orientation filtering approach, marks can be located and classified without the need for binarization, or the need to specify a scale or template explicitly. Using a steerable approach avoids the need to know skew *a priori*. Using this perspecitve grid-modelling method allows hand-held and hand-positioned cameras to reliably decode DataGlyphs, increasing the range of applications for which Dataglyph technology can be used.

5 Acknowledgements

The author would like to thank Jeff Breidenbach and Jindong Chen of Palo Alto Research Center for the code they provided, their support and helpful comments. The author would also like to thank Christopher Dance for his extensive help and suggestions.

Database	Total No. of Images	Range	% Decoded	
			Current Decoder	New Decoder
Blur	30	0.5–1.5σ	33	93
Noise	30	5–15σ	80	100
X & Y Pencil	60	1–3 shims	47	98
Skew	21	0–30°	52	100
Resolution	50	0–4 shims	50	98
Overall	**191**	–	**52**	**98**

Table 2: Table showing the overall performance of the new proposed decoder versus the current decoder for camera images.

References

[1] S. Ando. Consistent gradient operators. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(3):252–265, March 2000.

[2] H.S. Baird. The skew angle of printed documents. *Society of Scientific Photographic Engineering*, 40:21–24, 1987.

[3] C.R. Dance. Perspective estimation for document images. *Proceedings SPIE Document Recognition*, Volume 4670, 2001.

[4] R.D. Duda and P.E. Hart. Use of the Hough transform to detect lines and curves in pictures. *Communications of the ACM*, 15:11–15, 1972.

[5] W.T. Freeman and E.H. Adelson. The design and use of steerable filters. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(9):891–906, September 1991.

[6] D.L. Hecht. Embedded Data Glyph technology for hardcopy digital documents. In *Proceedings Color Imaging: Devide-Independent Color, Color Hardcopy and Graphic Arts III*, volume 2171, pages 341–352, 1994.

[7] D.L. Hecht. Printed embedded data graphical user interfaces. *IEEE Computer*, pages 47–55, March 2001.

[8] J. Illingworth and J. Kittler. A survey of the Hough transform. *Computer Vision, Graphics and Image Processing*, 44:87–116, 1988.

[9] T. Kindberg. Implementing physical hyperlinks using ubiquitous identifier resolution. Technical Report HPL-2001-95, HP Laboratories Palo Alto, March 2002.

[10] M. Kobayishi and H. Koike. EnhancedDesk: integrating paper documents and digital documents. In *3rd Asian Pacific Conference on Computer Human Interaction*, pages 57–62, 1998.

[11] P Ljungstrand and L.E. Holmquist. Webstickers: Using physical objects as WWW bookmarks. In *Extended Abstracts of the CHI 99*, pages 332–333, 1999.

[12] J. Mundy and A. Zisserman. *Geometrical Invariance in Computer Vision.* MIT Press, 1992.

[13] W. Postl. Detection of linear oblique structures and skew scan in digitized documents. In *Proceedings International Conference on Pattern Recognition*, pages 687–689, 1986.

[14] M. Spasojevic and T. Kindberg. A study of an augmented museum experience. Technical Report HPL-2001-178, HP Laboratories Palo Alto, July 2001.

[15] T. Tuytelaars, M Proesmans, and L. Van Gool. The cascaded Hough transform. In *International Conference on Image Processing*, page II:736, 1997.

[16] P. Wellner. Interacting with paper on the Digital Desk. *Communications of the ACM*, 36(7):86–89, 1993.