Embedding a Standard within a Standard using Mobile Progressive Barcodes

Marie Vans[1], Matthew Gaubatz[2] and Steven Simske[1]
HP, Inc., [1]Fort Collins, CO, USA, [2]Seattle, WA, USA

Abstract

Progressive barcodes have been proposed as a mechanism to employ Data Matrix barcodes on a package or label to enhance security services for products in a supply chain by layering additional data into color channels. While the standards-based component of the code remains readable by the same equipment, the extra (color) layer requires additional processing to extract relevant information, communicate with appropriate database modules, etc. This paper examines the problem of extending the scope of applications addressable by this technique to a wider range of designs intended for mobile consumption. This challenge involves developing a scheme that can be naturally used by a greater number of mobile devices, as well as increasing the ease of adoption. The extension of progressive barcodes to the more ubiquitous QR code via embedding a compact Data Matrix code into the design is an approach for improving functionality of print and document workflows. The two barcodes can be used simultaneously for their most familiar purposes. Mobile processing allows us to leverage many more possible object-driven workflow touchpoints.

I. Introduction

Data Matrix barcodes embed dense information into small printed objects. This barcoding scheme serves as an example on-ramp for a wide-range of object-based workflows. One type of variant, designed to help secure an object throughout its lifecycle in a supply chain is based on progressive barcodes, by strategically layering additional information into the marking via color channels [1, 2]. The progression of colors within the code supports an *inference model*, which, due to the structure of the allowable progressions, aids in efforts to mitigate illicit uses of the marked objects by, for instance, localizing points of diversion, providing additional layers of workflow directives, or simply increasing coded data capacity.

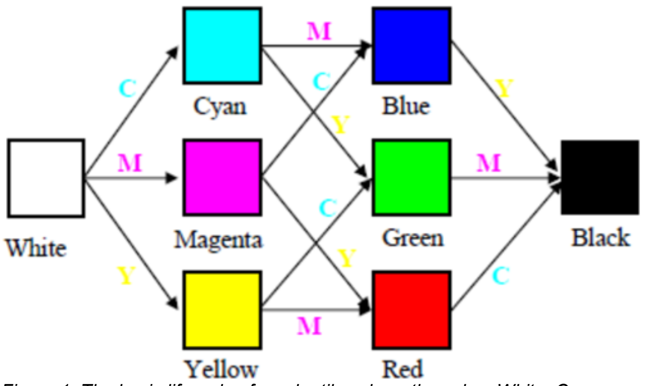

Figure 1. The basic lifecycle of a color tile, where the colors White, Cyan, Magenta, Yellow, Blue, Green, Red and Black are shorthanded as W, C, M, Y, B, G, R and K, respectively. Note that the tile may be written to three times in its lifecycle, and contain one of 8 states (3 bits).

Previously, progressions, as shown in Figure 1, were applied to modules in Data Matrix codes [3]. At the first stage, the W tiles can be overprinted with C, M or Y to create a C, M, or Y colored tiled, respectively. At the second stage, the C tile can be overprinted with M or Y to create a B or G tile, respectively; the M tile can be overprinted with C or Y to create a B or R tile, respectively; and the Y tile can be overprinted with C or M to create a G or R tile, respectively. This paper examines the problem of extending the scope of applications addressable by this technique by incorporating these progressive structures into new barcoding schemes, i.e., QR Codes [4]. The solution should be compact, flexible, and straightforward to incorporate into existing object-driven workflows.

Embedding standard barcodes into printed marks for object-based workflows was originally investigated for security purposes, using Data Matrix codes inside of color tile markings. Figure 2, taken from [5], shows an example.

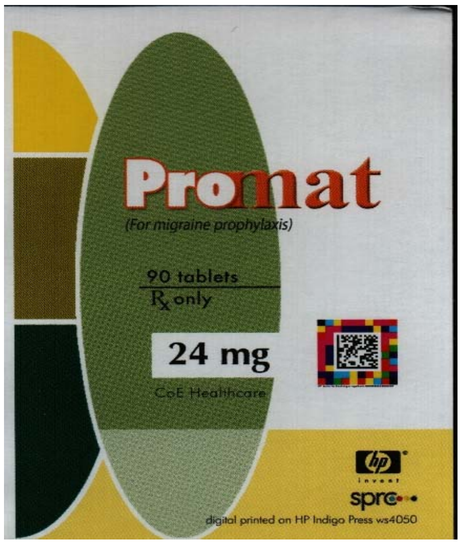
Promat
(For migraine prophylaxis)
90 tablets
℞ only
24 mg
CoE Healthcare
invent
digital printed on HP Indigo Press ws4050

Figure 2. Pharmaceutical packaging featuring an embedded data matrix barcode inside the color tile mark. [5]

The color tiles are organized around the two-dimensional data matrix code as shown in the label. The tiles are arranged on a 10x10 grid, so that the deterrent can support 100 tiles. In the default configuration, there are 64 colored tiles containing red, blue green, cyan, magenta, yellow, or white. These deterrents were investigated towards increasing the *payload density*, i.e., the amount of data that can be carried within a small footprint. While color-tile-based scheme certainly carries a high capacity payload, it requires special tooling to leverage and use the layer of data represented by the color tiles anywhere in a workflow it is to be used.

QR codes enable a wider range of touch points in an object-directed workflow, as these are easily interrogable by a plethora of devices. For a progressive barcode solution to be integrated into a QR code-based workflow, a standards-compliant method of implementing a progression is required. This topic is discussed in section II. In addition, the technique must be easy to apply on a broad range of devices, and an easily deployable mobile solution to this problem is discussed in section III. Results obtained using variations on the proposed approach are articulated in section IV.

II. Data Matrix Inside QR Elements

It is well known that QR codes can be rendered in exotic, even quixotic, ways while remaining readable [6, 7]. It is also known that finder pattern size correlates strongly with user experiences with 2D barcodes [8]. Thus, with respect to image processing concerns, a QR code is a strong candidate for implementing a progression to drive a workflow. Figure 3 illustrates an instance of a progression using a Data Matrix code, and Figure 4 gives a similar example using a QR code. While a "pure-QR" approach is a strong candidate solution, it is important to promote flexibility and ease-of-use. A hybrid solution, i.e., one where a Data Matrix code is embedded within a QR code creates a marking applicable to a greater number of workflows, and yields the ability to do so with more numerous examples of existing hardware and software tooling.

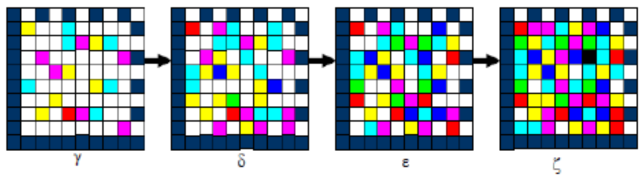

Figure 3. Example progression of codes in a "pure" Data Matrix framework.

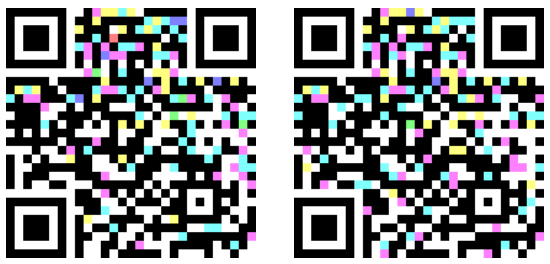

Figure 4. Example progression of codes in a QR code-compliant framework.

An example of such a design (non-progressive) is illustrated in Figure 5. One of the biggest challenges in extending progressions to this hybrid design is in creating augmentations that still enable both QR code and Data Matrix elements to be robustly readable; that is, the error-correcting code (ECC) of the enclosing barcode is not parasitically exhausted by the included barcode. Both do have calibration marks, or finder patterns, and both can use forms of Reed-Solomon ECC. But differences in the finder patterns lead to different degrees of signal detectability, which in turn effect decoding performance. As a result, algorithms designed to read codes of one kind may not be tuned to achieve the same performance on codes of the other, under equivalent spatial (or error-correction) density, once progressions are implemented. In other words, creating a set of progressions that have equivalent effects, or rather, are equally robustly interpreted when integrated into both standards, is not straightforward.

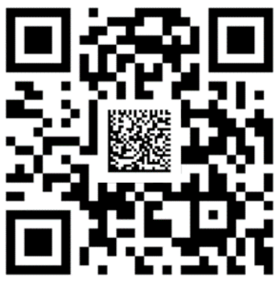

Figure 5. Example hybrid code: a Data Matrix code contained within a QR code. By scanning the design at different distances, both codes are readable with standard tools.

Though barcoding systems modified to include a progressive layer need not use a set of progressions that are tuned in the same manner, since ease of adoption is a key goal, it is usually convenient if componentry used to interpret progressive layers can be reused. These criteria help to promote robust performance on as wide a set of reading devices as possible; the cost incurred for this advantage is a reduction in the capacity of the progressive layer. Still, the presence of the QR finder patterns essentially removes the requirement that the embedded Data Matrix code elements be robust to the signal detection and orientation process. In other words, the error correction applied to the Data Matrix code need only be enough to compensate for general imaging distortions, and the chroma changes implementing the progression. Because Data Matrix codes are designed to be dense, there are many applications where this capacity constraint is not limiting.

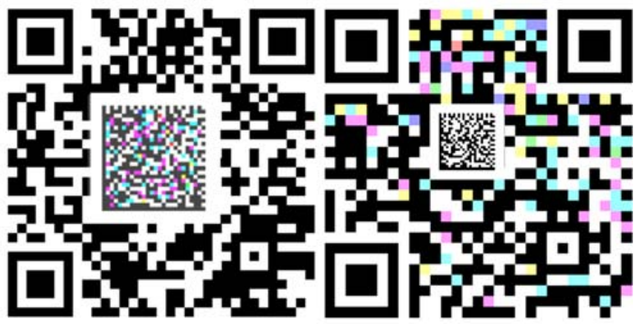

Figure 6. Comparison of different hybrid QR/progressive Data Matrix codes; the progressive portion is the same in both examples. Note that it is possible to represent nearly arbitrary graphical content within a QR code without significantly disrupting the quality of the payload [6]. Left side contains 3 channels of information: 1-black & white QR code, 2-black & white Data Matrix, 3-color channel in the Data Matrix. Right side also contains 3 channels of information, however, in this case the color channel is associated with the QR code rather than the Data Matrix barcode.

Figure 6 illustrates different examples of hybrid progressive QR/Data Matrix codes. Note that there is no real need for the modules in either code to be the same size, although there are visual benefits to making that choice. Furthermore, it is straightforward to process the central portion of the QR code as if it were, in fact, part of a Data Matrix code (see Figure 7; in essence, the decoder would need to synthesize the Data Matrix finder patterns). Whereas this mechanism is somewhat inelegant, it reuses known rendering and progression routines that perform well with progressive Data Matrix codes. It also improves the achievable capacity of the progressive layer simply because four additional rows and columns of modules (checkerboard edges plus to single-module quite zones) can have a pronounced effect on the amount of data that can be robustly embedded, especially in smaller designs.

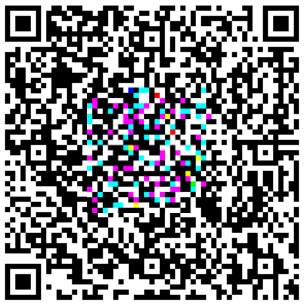

Figure 7. Example QR-Data Matrix hybridization where quiet zones and finder patterns have been removed entirely. In this case, the static data associated with the black & white Data Matrix is also no longer readable by standard barcode readers, but would be if the quiet zones and finder patterns were superimposed over the QR code around the edges of the color region.

III. Mobile Decoding Solution

There are many barcode reading and analysis tools from which an implementation of a reader for a progressive QR code reader can be constructed. In fact, QR code reading is a native feature on several different platforms. Nonetheless, the progressive layer requires strategic process of the *color* information associated with the barcode, and hence involves an image capture operation. Thus, it is important to be able to access an image of a marking obtained when or after a QR code has been detected and decoded. Furthermore, the solution should ideally be usable on as many platforms as possible. An obvious choice in that regard could be to rewrite and/or modify existing barcode readers to perform the required hybrid (composite) reading operation.

A growing trend in mobile development is to expose more native capabilities via JavaScript APIs that allow web programs to implement highly functional applications with minimal overhead. Complementing this trend is a set of tools designed to help authoring such code, one of which is capable of translating C++ (Clang) compiled bitcode into a JavaScript library [9]. The basis for the proposed solution herein uses a barcode reading library [10] that was created from a C++ implementation of a well-used barcode library. The main appeal of this approach is straightforward: no installation is required for use on a mobile device. A caveat is that the APIs needed to enable video rate camera frame access are not implemented in every available mobile browser. There is growing support for Android and PC devices in a variety of browsers [11], and this capable should be extended to mobile Apple products in an upcoming operating system release [12].

Because the solution is inherently web-based, it is straightforward to read a barcode, recover the finder pattern coordinates needed to infer the projective transform portion of the distortion induced by a capture process, and send a recovered image to a server/the cloud for further processing. An alternate solution interprets the layering information directly in JavaScript on the device. Whereas this approach is efficient, a server-based decoding system is more convenient for continuing and administering object-based workflows that connect the application to privileged information or other sensitive material regulated via role-based access control. A system diagram for the proposed solution is given in Figure 8. An additional advantage of such an approach is that if there is enough flexibility in controlling the payload for the QR portion of the design, it can be set up to include both a URL and variable data, as a query string, such that the utility itself used to interpret the progressive layer information is automatically associated with any object bearing the progressive marking.

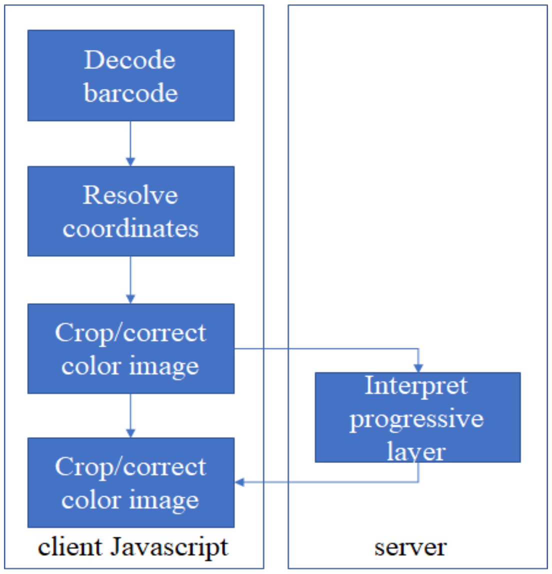

Figure 8. Client-server system diagram for web-based progressive layer interpretation engine.

IV. Results and Discussion

The proposed approach was implemented using a combination of C# and JavaScript for rendering, capture and interpretation. It was tested on an Intel-based PC, a Linux environment and several Android devices. The focus was on testing of mobile platforms, but extensibility to different imaging form factors was considered as well. In every example, the system was tested using Google's Chrome browser. It was found that the tested approach worked at several different resolutions, including 1920x1080; performance at this highest resolution was reasonably fluid. A weakness of the proposed approach is that it does require an internet connection, but in many cases, the resources being controlled or accessed need to be accessed via the web anyway. The barcode reading operation proper was not quite as robust as when implemented via compiled and installed native code, but this effect did not hinder the overall functionality; as cameras and processing power in mobile devices improve, this different will only decrease. Figure 9 illustrates the web-based mobile decoding solution in action on a Samsung Galaxy S6. The approach was also tested with older devices (including an S5 and an S4). The top image reflects a progressive QR code as viewed through a traditional barcode reader, and the bottom image illustrates the process at the point where a code is detected, and an image is being sent to a server to process the progressive layer.

While the examples in Figure 6 are somewhat contrived, they illustrate the ability to embed different channels of information into a single mark. The left side of Figure 6 contains a standard QR code and a Data Matrix, each encoded with a different email address. The right side of Figure 6 again contains a text statement and an email address encoded in the standard black & white channels of each. The color channels in these examples, whether associated with the QR code or the Data Matrix barcode can be read with proprietary software accessible through applications available on various platforms such as desktop and mobile. To give an example of a workflow application, suppose the QR code encodes a URL to a website. Once at the website, the black and white channel of the Data Matrix contains a code for use within the website, for example, to unlock a coupon. The color channel can then be used to verify the authenticity of the coupon. Another includes packaging: a (possibly static) QR code with a product ID could be carried throughout a supply chain, while the progressions in the code reflect subsequent transactions throughout this ecosystem involving the product with that ID. Inclusion of a Data Matrix code would enable the product to be scannable by a wider range of devices, i.e., consumer cameras and industrial scanning machines.

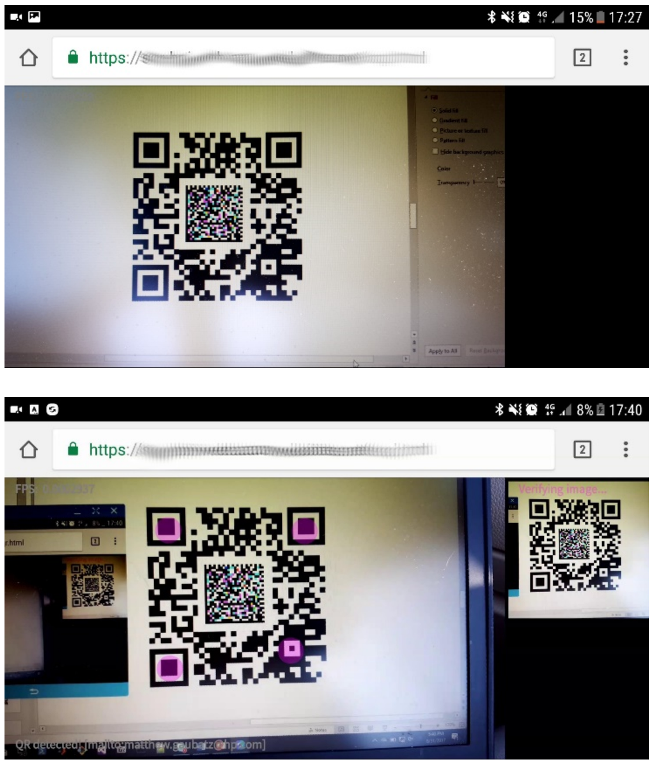

Figure 9. Example mobile decoding solution in action on a mobile phone (Samsung Galaxy S6). The top image shows the default view within the decoder, which resembles many existing barcode readers. When a QR code is detected and decoded, magenta dots appear on top of the detected finder patterns. Then, based on their positions, the coordinates of the finder patterns are resolved such that the image can be rectified, and the marking is cropped and sent to a server for further processing of the progressive-layered (chroma) data. Note that this same application can also be used on any computer if it is running Chrome or any browser supporting the appropriate APIs, and it is connected to a camera.

References

[1] S. J. Simske and A. M. Vans. Applications for Progressive Barcodes. J. Imagining Science and Technology. 2014, 58, 4, pp. 40404-1-40404-9.

[2] A.M. Vans, S J. Simske, and B .Loucks. Progressive Barcodes, Proceedings of the Digital Fabrication and Digital Printing Conference, NIP28, Quebec City, Quebec, Canada, 2012. pp. 368-370.

[3] International Standard ISO/IEC 16022:2006(E), Second edition 2006-09-15, “Information technology – Automatic identification and data capture techniques – Data Matrix bar code symbology specification,” 142 pp., 2006.

[4] International Standard ISO/IEC 18004:2015, Third edition. “Information technology – Automatic identification and data capture techniques – QR Code bar code symbology specification,” 117 pp., 2015.

[5] S.J. Simske. Meta-algorithmics: patterns for robust, low cost, high quality systems. John Wiley & Sons, 2013. Pp. 87.

[6] K.-T. Lay, Y.-J. Chen, H.-C. Hsueh and S. G. Karungaru, “Visually comprehensible QR codes via embedding of big logos,” Proceedings of the IEEE International Conference on Signal and Image Processing (ICSIP), Aug. 2016, pp. 225-230.

[7] G. J. Garateguy, G. R. Arce, D. L. Lau and O. P. Villarreal, “QR Images: Optimized Image Embedding in QR Codes,” IEEE Transactions on Image Processing, vol.23, no. 7, 2014, pp. 2842-2853.

[8] K. T. Tan and D. Chai, “A New Perspective on First Read Rate of 2D Barcodes in Mobile Applications,” Proceedings of the IEEE Conference on Wireless, Mobile and Ubiquitous Technologies in Education (WMUTE), April 2010, pp. 192-194.

[9] A. Zakai, Emscripten: An LLVM to JavaScript Compiler, https://github.com/kripken/emscripten, Dec. 23, 2016.

[10] D. Schmich, HTML5 QR code scanner using your webcam, https://github.com/schmich/instascan, April 3, 2017.

[11] MediaDevices.getUserMedia() – Web APIs | MDN, https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia, August 28, 2017.

[12] Safari technology preview release notes | release 38, https://developer.apple.com/safari/technology-preview/release-notes/.

Author Biography

Marie Vans is currently a Research Scientist with Hewlett-Packard Labs in Fort Collins, Colorado. Her main interests are security printing and document analytics. She has a Ph.D. in computer Science from Colorado State University. She also received a Masters of Library and Information Science the Department of Information from San José State University in 2016, where she is focused on technologies for distance education.