

Acode: a Registration-free Printed Media Watermark with Positioning Feedback

Eric Métois, Ph.D.

Escher Group Ltd
101 Main Street
Cambridge, MA 02142 - USA
(1) 617.721.7440

metois@eschergroup.com

ABSTRACT

This paper describes a novel technique for encoding data into and decoding data from a coded pattern using an angular symbology in which data symbols are represented by angular orientations of data modulation patterns relative to a reference modulation pattern. The modulation patterns are selected to have components that are localized in a Fourier transform domain and the technique may convey arbitrary digital data within 2D images. This symbology's targeted regime of operation would typically involve D/A and A/D stages: a coded image is printed on some physical medium, and subsequently optically re-digitized prior to the retrieval of the embedded digital data. The technique was explicitly designed to minimize any registration issue, which typically occurs for this type of technology. It was also designed to be graphically versatile and sensitive to the appearance of the resulting coded images. Additionally, it exhibits the unique ability to convey positioning information throughout the image that carries it. Not only can the data payload be retrieved from any section of the coded printout, but the position of the imager with respect to the printout can also be estimated.

General Terms

Algorithms, Measurement, Performance, Design, Verification.

Keywords

Symbology, Watermark, Data embedding.

1. INTRODUCTION

There are numerous applications in which machine-readable data must be encoded on a paper document or other substrate. These may range from product identification and tracking to document security and authentication. Typically such data is encoded using a printable symbology, either as a form of barcode [1], as characters optimized for optical character recognition, or as some form of digital watermark [2] [4].

Within such symbologies, stringent registration requirements are typical hurdles. Indeed the retrieval of the embedded data often requires a precise alignment of some sort. In the case of a 2D symbology, this means that any amounts of translation, rotation and scaling of the captured image must be estimated and compensated for. This can be a computationally intensive task.

Acode was explicitly designed to minimize any registration issue, conveying digital data within the coded image in both a translation and rotation independent fashion.

Additionally, Acode's design was sensitive to the appearance of the resulting coded images. The raw coded image will typically look like a fine texture, which can be combined with a pre-existing image. The present technology does not literally qualify as a digital watermark because it does not aim to absolute transparency. Rather, it aims to keep visual disturbances at a reasonable level within its intended application (i.e. printed material).

Finally, Acode offers the unique ability to convey positioning information throughout the image that carries it. As a section of the encoded and printed image is optically scanned, this feature permits the recovery of the center position and the orientation of the section with respect to the original image's dimension.

In what follows, we will discuss the underpinnings of our approach before describing the symbology itself and its full encoding and decoding stages. We will then discuss Acode's performance and describe a few sample applications.

2. UNDERPINNINGS

This section introduces the underpinnings that constitute the basis of Acode's design.

2.1 Basic elementary components

Within our discussion, a real spatial frequency component will refer to a sinusoidal linear pattern. Its Fourier transform is a complex image consisting of two complex conjugated values. The distance between these non-zero complex values and the origin of the Fourier plane corresponds to the spatial frequency of the pattern. The orientation of these two non-zero complex values matches the orientation of the linear pattern. While translating such linear pattern will modulate the phase of these two complex values in the Fourier domain, their magnitude and positions will remain unchanged.



Figure 1 – (From left to right): an elementary component; application of a 2D Hamming window; FFT-based estimation its Fourier magnitude domain.

We recall that a Fourier transform is an abstract mathematical tool that assumes an infinite spatial support for the original image. In the real world however all images have a finite support and resolution. An FFT will treat an image as an elementary “tile” of an infinite support image. The computed Fourier series will therefore reflect this arbitrary tiling pattern. The most common way to reduce such boundary artifacts is to apply a smooth window to the original image prior to its FFT analysis. Figure 1 illustrates the application of a 2D Hamming window to the original image.

2.2 Elementary radial components

Within our discussion, a real elementary radial component will be a sinusoidal pattern that exhibits a radial symmetry around a location (x,y) within an image. Its Fourier transform is a complex image consisting of zero values everywhere except along a circle, which radius corresponds to the frequency of the component.

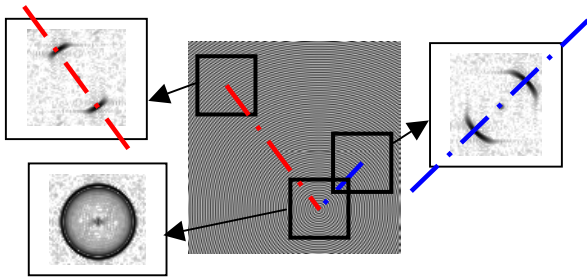


Figure 2 – Radial component and Fourier magnitude estimations for three sub-sections.

As a sub-section of the image is analyzed through an FFT magnitude plot, the energy along this circle will not be evenly distributed unless the sub-section happens to be centered on the center (x,y) of the component. Recalling that the Fourier magnitude plane will be symmetrical around its center, the location of the maximum energy value along that circle will determine a direction (i.e. an orientation modulo 180 degrees). Back in the spatial domain, this direction corresponds to the line that joins the center of the current sub-section of the image, and the center of the printed radial component. This phenomenon is illustrated in Figure 2.

2.3 Angular measurements in the Fourier domain

We recall from above that the orientation of a linear is clearly exhibited in the magnitude Fourier transform domain. Figure 3 shows the example of a pattern that was created by adding three frequency components with different spatial frequencies and different orientations.

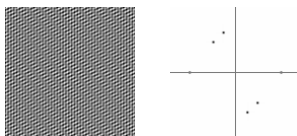


Figure 3 – Composite of three elementary components with different spatial frequencies and orientations.

In order to measure the orientation of a component, for which the spatial frequency is known, an effective approach is to look for peak positions of the magnitude Fourier transform along a circle of appropriate radius. In the context of elementary radial components, the exact same process maybe applied in order to find the direction linking the center of the scanned sub section and the center of the radial component.

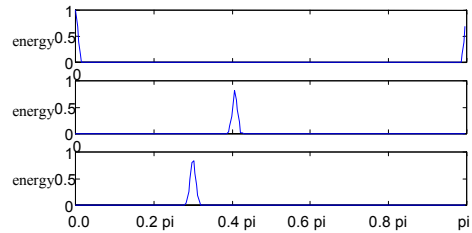


Figure 4 – $[0,\pi]$ Polar sweep of the Fourier domain for three spatial frequencies.

Recalling that the estimated magnitude Fourier domain has a limited resolution, such polar lookup will typically require interpolation. Figure 4 shows the result of such polar sweep in the context of the example of Figure 3. This sweep used bilinear interpolation. As a means to extract the orientations of each component from such polar sweeps, a combination of local maximum and weighted average seems to yield to satisfactory accuracies.

3. DESCRIPTION OF ACODE

3.1 Features

We recall that our primary motivation is to develop a symbology relying on both translation and rotation independent features. In the light of the previous section, the orientation of a single frequency component is translation independent. Relative angles between frequency components are both translation and rotation independent. We chose to base the symbology upon similar angular features.

As for the ability to determine the center location of a sub-section of an encoded image, radial components offer a very appropriate behavior. Indeed, we have seen that an angular measurement in the Fourier domain can provide a direction linking the center of our sub-section and the center of a radial component. Having two radial components at known locations and a reference orientation, we can therefore use these directions to triangulate the (x,y) center position of any sub-section of our image.

3.2 Architecture and conventions

Acode is based on the following objects and their relationships.

Elementary Component: An *Elementary Component* will refer to a pattern that exhibits a singular distribution of energy in the magnitude Fourier domain. In the current implementation of Acode, these are the sinusoidal linear patterns we’ve illustrated in Figure 1.

Beacons: A *Beacon* will refer to a pattern that exhibits a radial symmetry. Unlike an *Elementary Component*, a *Beacon* is not invariant through translation. Yet, its radial symmetry is key to the localization of its center by means of a direction. In the current implementation of Acode, these are the radial sinusoidal patterns we’ve illustrated in Figure 2.

Reference Orientation: In order to measure the angular features the symbology is based on, the system needs a reference orientation. All angular features will be measured with respect to this reference.

Carrier: A carrier will refer to a spatial frequency (i.e a radius in the Fourier domain). A *Carrier* serves as a modulating frequency for the chosen *Elementary Components*. One of these carriers will be reserved for the *Reference Orientation*. As such, this *Reference Carrier* will contain a single *Elementary Component*. All other *Data Carriers* may contain multiple *Elementary Components* (same spatial frequency but different orientations).

Angle: An *Angle* will refer to the difference between an *Elementary Component's* orientation and the *Reference Orientation*. It is the feature upon which the symbology is based.

Sector: A *Sector* will refer to a range of *Angles*.

Symbol: A *Symbol* will refer to a data element that is conveyed by a single *Angle*. *Data Carriers* will convey as many *Symbols* as they count *Elementary Components*. In the case of multiple *Symbols per Carrier*, *Angles* will be organized in *Sectors*.

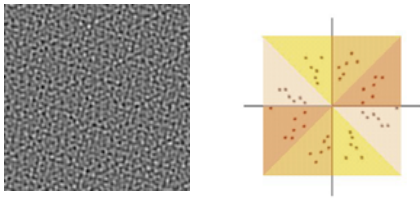


Figure 5 – Spatial and Fourier domain representation of the proposed symbology

Figure 5 illustrates a coded texture using 7 *Carriers* (1 *Reference Carrier* and 6 *Data Carriers*). Each data carrier conveys 4 *Symbols*. The image on the left is the coded texture itself while the image on the right is its magnitude Fourier domain representation. The carrier of highest frequency conveys the *Reference Orientation*. Moving clockwise from this reference orientation, the $[0,\pi]$ half plane is divided into four *Sectors*. Each data carrier contains 4 *Elementary Components* whose orientations fall within each sector. With this configuration, Acode conveys a total of $4 \times 6 = 24$ symbols.

Note that in practice, we typically assign different numbers of symbols to the different carriers. Indeed, the accuracy of our angular measurements will typically increase with the frequency of the carrier. Hence, higher frequency carriers have the ability to convey more data than lower frequency carriers.

3.3 Data layer

Number of bits per symbol: Acode's data capacity depends upon the accuracy of our angular measurements. The more accurate these measurements are, the more bits can be conveyed by each symbol. Tuning this parameter is a direct compromise between robustness and data capacity for the symbology. Note also that in practice, different carriers may use different values for this number of bits per symbol within the same configuration.

Gray code: Once a number of bits per symbol is chosen, the data is carried by a quantization of angular measurements. It is desirable that the binary representation of these symbols is such that two adjacent "quants" differ by only 1 bit. It was in a rather similar context that Gray [5] came up with his binary coding system and Gray codes are used here.

Error detection and correction: Acode's current implementation is capable of a variety of configurations for error detection and correction, combining both standard parity bits and Bose-Chaudhuri-Hocquenghem (BCH) error correction schemes [3].

3.4 Positioning principle

Although one may decide to use a different configuration, we will explain the basic principle in the context of two *Beacons* located at the top corners of the image. The width of the image will act as a reference scale for the estimated center position (x,y) of imager.

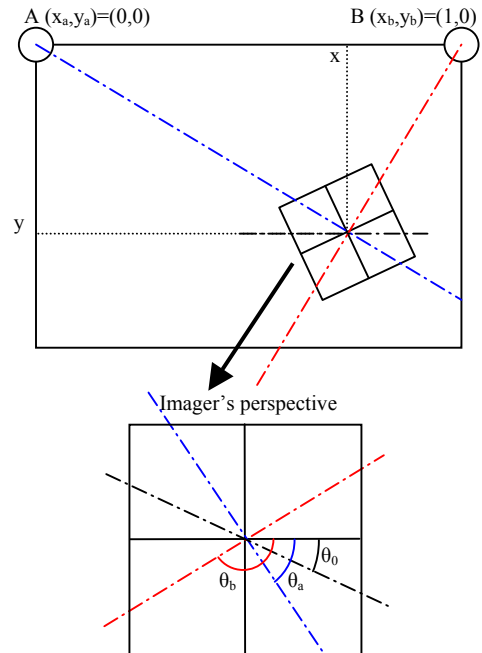


Figure 6 – Positioning principle.

Figure 6 illustrates the viewable of the imager with respect to the printed image.

We recall that Acode conveys a reference orientation. Therefore upon a successful decode we can recover the "horizon" of the viewable θ_0 . The directions of the positioning beacons θ_a and θ_b with respect to the imager are subsequently derived from local maximum estimations in the Fourier magnitude domain. From these three orientations and subject to $(x \neq 0; x \neq 1; \theta_a \neq \theta_b \text{ mod } \pi)$, basic trigonometry leads to estimates for the center location (x,y) of the viewable:

$$x = \frac{\tan(\theta_b - \theta_0)}{\tan(\theta_b - \theta_0) - \tan(\theta_a - \theta_0)},$$

$$\text{and } y = \frac{\tan(\theta_a - \theta_0)\tan(\theta_b - \theta_0)}{\tan(\theta_b - \theta_0) - \tan(\theta_a - \theta_0)}$$

4. ENCODING STAGE

4.1 Creation of a raw coded texture

Using nc carriers, $nspc_k$ ($k=0, \dots, nc-1$) symbols per carrier and $nbps_k$ ($k=0, \dots, nc-1$) bits per symbols, we wish to create a texture containing a N bits payload where:

$$N = \sum_{k=0}^{nc-1} nspc_k nbps_k$$

This payload of N bits contains the data we wish to convey as well as other redundancy bits that are necessary for error detection and correction.

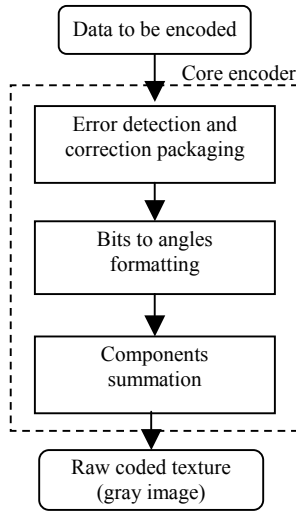


Figure 7 – Core of Acode encoder.

That payload is then fed to the **Bits to angle formatting** block. There, it is first segmented into $(nspc_0 + \dots + nspc_{nc-1})$ binary words in accordance with the chosen number of bits per symbol parameter. Each data segment is subsequently turned into a symbol value $S_{k,i}$ according to Gray's code. Provided with a reference orientation, these $(nspc_0 + \dots + nspc_{nc-1})$ symbol values are subsequently turned into orientations as follows.

$$\theta_{k,i} = \theta_0 + \pi/40 + i\pi/nspc_k + \pi\left(1/20 + 1/nspc_k\right) \frac{S_{k,i}}{2^{nbps_k}}$$

where $k=\{0, \dots, nc-1\}$ and $i=\{0, \dots, nspc_k-1\}$.

Note: As shown above, these orientations do not span their entire sector ($\pi/20$ radians are systematically taken out of the range). Enforcing a “dead zone” between sectors is a way to ensure that two elementary components residing on the same carrier will never interfere.

These orientations, along with the reference orientation, will lead to a total of $(1+nspc_0 + \dots + nspc_{nc-1})$ elementary components. The corresponding $(nc+1)$ spatial frequencies $(\omega_R, \omega_0, \dots, \omega_{nc-1})$ are chosen as part of the system's configuration.

Note: The choice of carrier frequencies will typically depend upon the expected minimum resolution and size of the captured section.

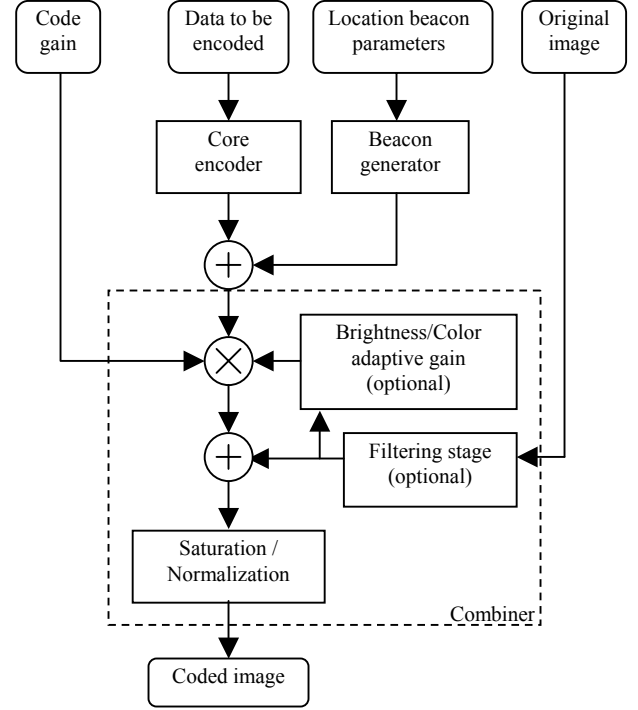


Figure 8 – Acode encoder

It is then up to the **Components summation** block to create the coded texture by adding these modulated components together:

$$C(x, y) = \cos(\omega_R(x \cos \theta_0 + y \sin \theta_0) + \varphi_0) + \sum_{k=0}^{nc-1} \sum_{i=0}^{nspc_k-1} \cos(\omega_k(x \cos \theta_{k,m} + y \sin \theta_{k,m}) + \varphi_{k,m})$$

Note: While this section illustrates the most straightforward method for creating such coded texture, it is worthwhile to note that in practice, creating such textures in a transform domain (e.g Fourier) is a more efficient alternative. This can be a little tricky due to the fact that the ideal coded texture is not periodic. Within the current implementation, the components' frequencies and orientations are slightly modified in order to create a periodic tile. This is done as a quantization to a square FFT grid. We found the resulting distortions to have a minimal impact on Acode's reliability for FFT blocks above 512 bins square.

4.2 Positioning beacons

In order to allow a reader to estimate its location within the printed image, we may wish to add positioning beacons to the coded texture before combining it with any potential image. This additional texture can be represented as the following sum.

$$B(x, y) = \sum_{k=0}^{nbeacons-1} \cos\left(\omega_{b,k} \sqrt{(x - X_{b,k})^2 + (y - Y_{b,k})^2}\right)$$

where the beacons (spatial frequencies $\omega_{b,k}$) are centered at the locations $(X_{b,k}, Y_{b,k})$. This summation is the function of the **Beacon generator** block shown in Figure 8.

4.3 Optional pre-filtering

Combining the raw coded texture with an existing image can raise some interference issues. Indeed, the coded texture carries its data payload in a spatially spread fashion (the data is spatially de-localized) but unlike the case of a spread spectrum approach, the data payload is strongly localized in the frequency domain. If an arbitrary image contains strong frequency components in the coding band of the symbology, then these components are likely to interfere directly with the data payload. In order to prevent such interference, it is sometimes necessary to low-pass filter the existing image prior to combining it with the coded texture. This stage is object of the **Filtering stage** block shown in Figure 8. This optional stage helps to guarantee the integrity of Acode’s payload over a small physical areas. For larger printed areas however, experience has shown that the payload can always be retrieved from a good portion of the printout without tempering with the image via such filtering stage.

Performing a straightforward sliding correlation with a $L \times L$ matrix M will typically involve L^2 multiplications per pixel in the image. Given that our case doesn’t require a specific impulse response, our preferred embodiment uses a design trick that dramatically speeds up the filtering process. We chose to design our finite impulse response 2D low-pass filter in such ways as to ensure that the matrix M is an outer-product:

$$M = U.U^T, \text{ where } U \text{ is a vector of length } L.$$

With this choice, the sliding correlation with the matrix M over the entire image can be expressed as the cascade of two 1D filtering stages over the two dimensions of the image. The elements of U constitute the impulse response of these 1D filtering stages. Each 1D filtering stage involves L multiplications per pixel and therefore, the entire sliding correlation with the matrix M involves only $2L$ multiplications per pixel.

4.4 Printed media considerations

There remain a few aesthetic concerns when the raw coded texture is combined with an image.

Saturation versus normalization: The resulting image, whether it is saved to a digital file of printed onto some medium will always be subject to a limited amount of dynamics for each color channel. Normalizing the coded image will lower the color dynamic attributed to the original image and the result will tend to lack contrast and definition. Acode’s current implementation favors saturation.

White areas on printed medium: Once the coded image is printed on a paper medium, the Acode texture closely resembles dithering or paper texture and it doesn’t raise any major aesthetic concerns when it is combined with most images. However, it becomes a lot more obvious in areas of the original image that are intended to be white and inkless. Indeed, such area fail to mask the Acode texture and the result is not aesthetically satisfying. Acode’s current implementation detects such problematic areas through with pixel brightness measure over the original image. These measures control the gain of the coded texture at every location of the image.

Color Smart compensations: An application to high quality art reproduction also revealed that a uniform distribution of the

coded texture over the color channels (RGB or CMYK) of an original image could lead to a slight shift of the overall color saturation of the piece. Detecting such shift required a trained eye and side-by-side comparison. This phenomenon was eventually eliminated with the introduction of a color compensation stage consisting of a proportional distribution of the coded texture over the color channels.

4.5 Tiling

The system may embed different data in different portions of an underlying image or identical sets of data in various portions of the image. This is simply done by tiling various coded textures. To smooth the transitions between the coded tiles, the system overlaps fade-in/fade-out portions of the tiles, leaving the centers of the tiles intact for decoding. The overall coded pattern can be formulated as a weighted average of the tile textures, with the weighting associated with a given tile decreasing with the distance from the tile center. Acode’s current implementation uses weights that vary exponentially with distance.

5. DECODING STAGE

Provided with a digital capture of an encoded image and an original guess as to the image’s scale, the decoder’s goal is to estimate the reference and the symbols’ orientations. As we’ve discussed it in section 2 of this document, this will involve a 2D hamming window, a 2D fast Fourier transform (FFT) and some polar lookup using bilinear interpolation in the Fourier magnitude domain. These angular measurements are then translated into symbol values, which are subsequently turned into binary words. These binary words are assembled into a binary stream that is finally sent to the error detection and correction stage. Upon a successful decode and assuming that the chosen configuration included positioning beacons, the system will proceed to estimate the location of the imager with respect to the printed coded image.

5.1 Scale: the remaining unknown

While the features that Acode is based upon are both translation and rotation independent, they are still sensitive to scaling. The decoder needs to know the values of the reference and the data carriers’ frequencies. Fortunately, trying multiple scale values in order to retrieve valid data from the image is not a computationally intensive task.

The most computationally expensive part of the decoding process is the first windowing and transform stage, which estimates the images’ magnitude Fourier transform domain. This stage doesn’t need to be revisited during a scale search. Indeed, any scale variation can be compensated by linearly scaling the frequency values of the reference and the data carriers prior to our polar lookup.

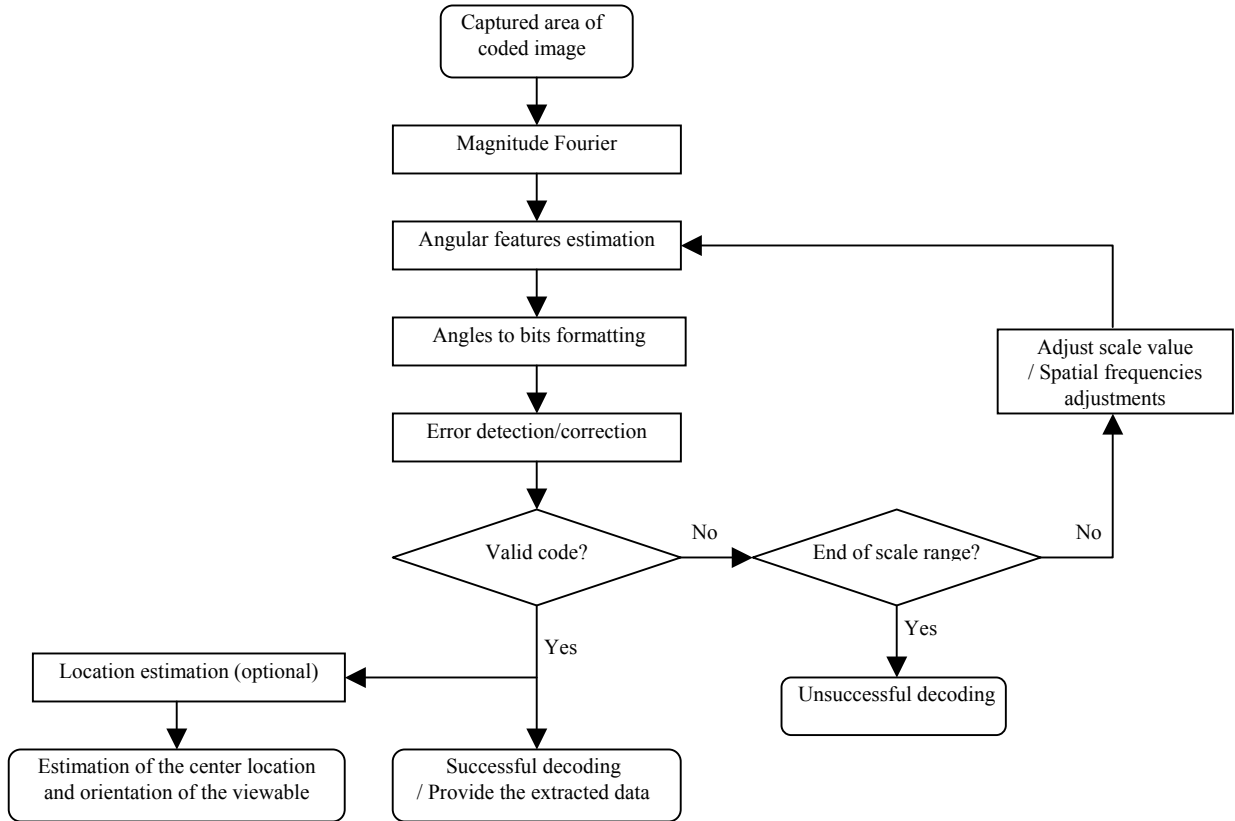


Figure 9 – Block diagram of the decoding process

Acode's current scale search implements the following principle: provided with an original "best guess" and a range around that guess for the image's scale value, the decoder will prepare a set of hypothetical scale values that spans that range. After the images' magnitude Fourier transform domain has been estimated and working from the center and towards the boundaries of that range of scale values, it will try these hypotheses until valid data is retrieved from the image or until the boundaries of the scale search range are reached. Using such approach, the current implementation was proven capable of compensating for scale ambiguities of over 50%. The range of scale values is typically spanned in intervals of 1%.

5.2 A complete decoder

Figure 9 is a block diagram of the entire decoding process. An area of the coded picture that was captured and digitized (by a camera aiming at a printed version of the image for instance) is provided to the *Magnitude Fourier* block. This block converts the provided image to grayscale by summing the color values of each color channel. It then applies a Hamming window to this grayscale image, leading to the windowed gray image X , before performing a 2D Fast Fourier Transform (FFT). The square magnitude of the FFT's result is subsequently computed and it constitutes the output of this block.

Using bilinear interpolation, the *Angular features estimation* block will retrieve the polar representations P that are pertinent to the carrier frequencies:

$$P(\omega, \theta) \equiv \|FFT[X]\|_{(\omega \cos \theta, \omega \sin \theta)}^2,$$

where $\omega \in \{\omega_R, \omega_0, \dots, \omega_{nc-1}\}$ and $\theta \in [0, \pi[$

The same block will then retrieve the reference orientation.

$$\theta_0 = \arg \max_{\theta \in [0, \pi[} P(\omega_R, \theta)$$

Once the reference orientation is identified, the polar representations of the data carriers are wrapped with respect to this orientation before being subdivided into the appropriate sectors. The *Angular features estimation* block will then proceed to look for the maximum values of the magnitude Fourier domain for each data carrier frequency and over each one of that carrier's sectors.

$$\theta_{k,i} = \arg \max_{\theta \in \left[\theta_0 + i \left(\frac{\pi}{nspc_k} \right), \theta_0 + (i+1) \left(\frac{\pi}{nspc_k} \right) \right]} P(\omega_k, \theta_{\text{mod} \pi})$$

where $k = \{0, \dots, nc-1\}$ and $i = \{0, \dots, nspc_k-1\}$.

Once taken with respect to the reference orientation, these orientations yield to the set of angular features that constitute the output of this block.

The *Angle to bit formatting* block is responsible for tuning these angular measurements into an actual bit stream. The block must first translate these angles into symbol values. This is done in a way that is obviously symmetrical with respect to the encoder and it is as follows:

$$S_{k,i} = 2^{n_{bps}_k} \frac{(\theta_{k,i} - \theta_0)\pi^{-1} - 1/40 - i/n_{spc}_k}{1/20 + 1/n_{spc}_k}$$

where $k=\{0,\dots,nc-1\}$ and $i=\{0,\dots,n_{spc}_k-1\}$.

These symbol values are subsequently rounded to integer values before they are decoded accordingly to Gray's codes. The resulting binary words are concatenated into a bit stream.

This bit stream is subsequently fed to the *Error detection and correction* block. We recall that Acode's current implementation is capable of a wide variety of combinations between parity bits and BCH error correction codes.

If the provided bit stream is not valid then the decoder can either terminate or try again after readjusting values of the reference and the data carriers' frequency. Such spatial frequencies adjustments are performed by the *Spatial frequencies adjustment* block in Figure 9.

Upon the successful retrieval of the embedded code, the decoder may then proceed with the estimation of the imager's center location. Such estimation will only be appropriate in cases where Beacons were inserted at the encoding stage. Such estimation is the function of the *Location estimation* block, and it is based upon the premises outlined in section 3.4 of this document.

6. PERFORMANCE

6.1 Robustness

Robustness is a measure of the embedded data's persistence through degradations of the medium the symbology is printed on.

The robustness of a communication channel is often measured in terms of the data's persistence through decreasing signal to noise ratios (SNR). The relevance of such measurement comes from the fact that the typical degradation of the channel can be reasonably modeled as an additive white noise and that the ratio of the carrier and the additive noise energies is a reasonable quantitative measure of degradation.

Unfortunately, this approach doesn't translate well to our case. Among the numerous types of degradations that may be applied to a piece of paper, very few could be reasonably modeled as an additive white noise. So while it may be possible to derive a similar SNR measurement of a printed symbology's robustness, the relevance of such measure would be greatly questionable.

Figure 10 illustrates a few types of physical degradations that are likely to occur in the real world (ink rubbed out, scribbles and punctures), none of which sharing much similarity to an additive white noise. Despite the severity of these examples, the

106-bits total payload of the shown example survives these degradations.

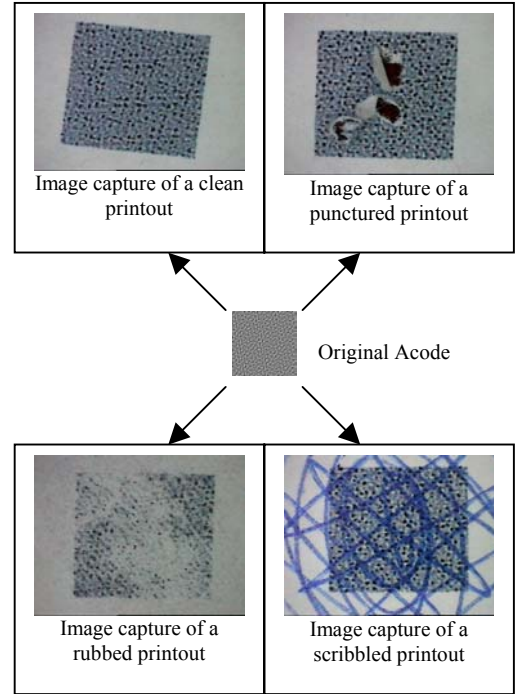


Figure 10 – Example of physical degradations.

6.2 Data Density

Because of its highly configurable nature Acode's data density doesn't boil down to a single number of bits per square inch. The achieved data density is always a function of the tradeoffs that took place during the design of a particular application. Additionally the non-locality of the data it carries is intrinsic to Acode's design and its mode of operation is better explained as the encoding of a fixed amount of data over a variable amount of area, rather than a fixed area per data symbol. This is a departure from symbologies such as traditional barcodes. However, one can still derive a similar "bit per area" data density measure C (bit.in⁻²) by considering the minimum area A_{min} (in²) that is required to convey a fixed amount of data L (bits).

$$C = \frac{L}{A_{min}}$$

The principal limiting factor for Acode's data density is the lower of the printing and scanning resolutions that are used for the application. From now on, we'll refer to this lower resolution as $dRes$ (dot.in⁻¹). As the achieved data density C will tend to increase linearly with the square of the value of the device resolution $dRes^2$, it is useful to define a normalized data density C_0 (in bits per printed dot) as follows:

$$C_0 = \frac{C}{dRes^2}$$

C_0 quantifies the achieved data density of the symbology regardless to the printing and imaging devices that may be

chosen for a specific application. It can be finely tuned in accordance with specific tradeoffs between robustness and aesthetic considerations. Figure 11 maps out values for C_0 that were achieved in the context of three very different applications.

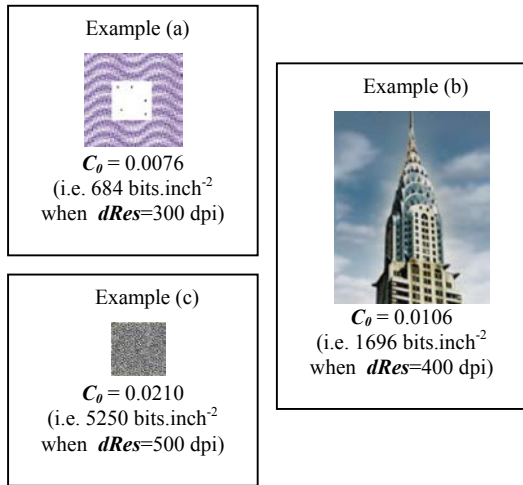


Figure 11 – Empirically achieved data density. Recall that C_0 is in “bits per printed dot”

Example (a): This configuration was used as the counterpart to Escher Labs’ FiberFingerprint analysis. The code and a clear area the paper were intended to be examined simultaneously. This implied that over the imaging device’s viewing area, the symbology would be significantly obstructed by a fiber analysis area. Coping with this constraint required from the symbology to use a conservative data payload.

Example (b): This configuration was used to convey data within a pre-existing image the size of a stamp. This configuration’s payload reflects a tradeoff between image quality and robustness.

Example (c): This configuration’s purpose was to convey data robustly through almost any printer/imaging channel. Indeed, this configuration is limited to black and white 2D representations by design. Its primary aim was to convey a relatively small and fixed amount of data over a small printed area.

6.3 Position feedback accuracy

It is difficult to quantify the precision of Acode’s positioning feedback because it greatly depends upon a large number of factors. The size and resolution of the printout, the size and resolution of the imager’s field of view, the print quality, the imager’s quality and the overall aggressiveness of Acode’s configuration will all influence this accuracy.

However we can still provide at least one data point by means of a specific case scenario. In the case of an encoded photo printed at 400dpi for a physical size of 6”x6”, and carrying two positioning beacons at its top corners, we used an imager configured to provide 528dpi captures (240x320 frame size). We proceeded to monitor the decoder’s position feedback as new frames were captured and observed that these estimates would never jump past a range of 1% or so. Recalling that the distance between the beacons act as a reference scale (i.e. 6 inches here), this means that for this application, the estimated

position of the reader afforded a precision of 6/100 inches (i.e. 1.5mm).

6.4 Computational requirements

Designing a symbology that avoided registration was partly motivated by computational requirements. As we stated earlier, the FFT stage is the most computationally intensive stage of the entire decoding process. In practice, these are in the range of 256x256 bins FFTs and they constitute a fairly small computational load by any computer’s standard nowadays. On a 750 MHz Pentium PC, the Acode reading stage from a 240x320 video capture was clocked around 150ms and most of that time can be attributed to moving that image data in memory throughout Windows OS.

On the encoding side, the computational requirement obviously scales with the size of the image we wish to encode. In the case of very large images and an early version of Acode’s encoder, it turned out that memory requirements were more stringent than processing power. This motivated two modifications to the encoding stage. The first was the creation of artificially periodic coded textures in the Fourier domain and the second was a software re-architecture allowing the encoder to work on large images block by block. For a project involving high quality art reproductions, original files could be in excess of 500MB each and depending on CPU and disk speed, encoding those would take somewhere in the vicinity of 4 to 5 minutes.

7. SAMPLE APPLICATIONS

7.1 Fiberfingerprinting

Escher Labs’ FiberFingerprinting technology is a means to produce secure documents from arbitrary and potentially low-value stock, using commercially available variable printing technology. By applying digital technology to a physical medium, an Escher FiberFingerprint can authenticate a document with a level of security that surpasses that of traditional techniques such as microprinting, pantographs or holograms.

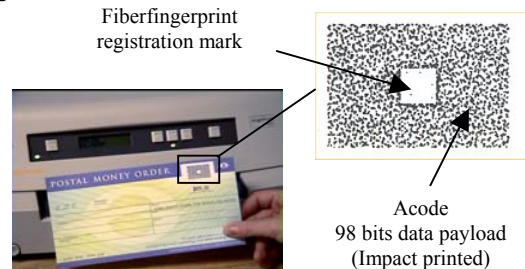


Figure 12 – Money Order Fiberfingerprint application.

To verify the authenticity of a document, an imager checks for a match between the physical substrate (using Escher’s FiberFingerprint analysis) and the data conveyed by the surrounding Acode. If the match fails, the document is revealed to be a counterfeit. This application demonstrated Acode’s reliability, its ability to accommodate low resolution printing technology, and its ability to accommodate a large obstruction (the Fiberfingerprint area and registration mark).

7.2 Token identification

Acode's versatility was further demonstrated with a token identification system. Here, the symbology was used to convey unique identifiers and it was engraved on plastic tokens.



Figure 13 – Token Identification

The engraving was accomplished using a laser cutter set to appropriately low power. The identity of each token was reliably recovered when placed on top of a 2D imager, once again demonstrating the symbology's reliability and ability to accommodate a wide range of "printing" technologies.

7.3 Traceable postage

The premise of this application was to convey a unique identifier on a postage stamp for track and trace purposes. Given the nature of the medium, this should be done in such ways as to minimize any visual artifacts associated with the embedding of the identifier while maximizing the code's robustness due to the small available physical area.

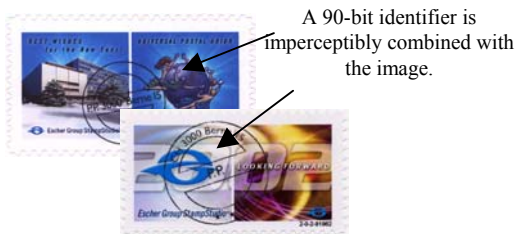


Figure 14 – Two samples of traceable stamps.

7.4 Interactive street maps

This system was primarily designed as a literal illustration of Acode's positioning feedback feature. A set of street maps from the Boston area were encoded to convey unique identifiers and two positioning Beacons at their top corners. Corresponding aerial photos were obtained for each of the maps. A demo software application was designed to retrieve Acode's payload and position feedback from a live video stream. The handheld reader simply consisted of a re-housed webcam fitted with proper optics and lighting as to focus on a 1/2 inch field of view.

As the reader is laid over the printout of an encoded map, Acode's data payload is used to identify the map and call back the associated aerial photo. The software application further provides visual feedback based on the positioning information, by overlaying a precise aiming mark and a horizon on the aerial photo.

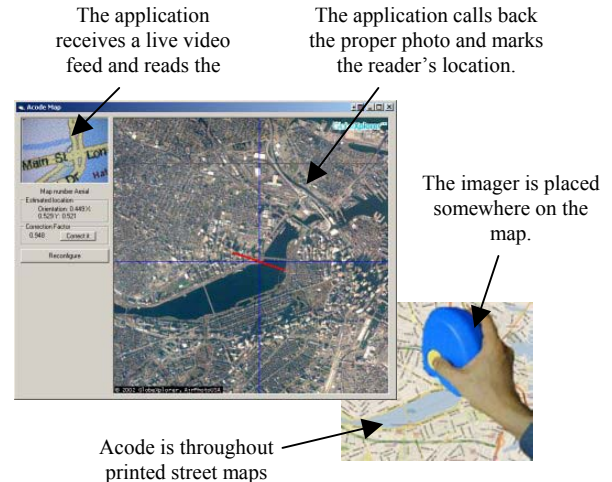


Figure 15 – Interactive Street Maps

8. CONCLUSION

Acode is a high performance symbology. It persists through severe degradation of the physical medium that carries it: obstructions, scratches, overlay, etc. It offers a high data density in the sense that it can recover a significant payload from a small physical area.

Acode is a versatile symbology. It supports a wide range of printing constraints in terms of resolution, color depth and quality. It also supports a wide range of imaging constraints in terms of field of view, color depth and quality. It supports a wide range of graphic design constraints as it can take virtually any shape and size, and be combined in a near-imperceptible manner with an arbitrary image. As a symbology, its applications may range from overt to covert encoding of identification data, security or other metadata.

Finally, Acode's positioning feedback sets it aside from any other image watermarking technique, and extends even further its range of possible applications to include interactive printouts and navigation.

9. REFERENCES

- [1] The Barcode Software Center. A Short Introduction to Barcodes. www.makebarcode.com/info/intro.html
- [2] Bender, W., Gruhl, D., Morimoto, N., Lu, A. Techniques for Data Hiding. IBM Systems Journal Vol. 35, No. 3&4. MIT Media Lab (1996) 313-336.
- [3] Bose, R.C., Ray-Chaudhuri, D.K. On a Class of Error-correcting Binary Group Codes. Inform. Control Vol. 3, pp68-79, March 1960.
- [4] Cox, I., Kilian, J., Leighton, Y., Shamoon, T. Secure Spread Spectrum Watermarking for Multimedia. Technical Reports 95-10, NEC Research Institute (1995).
- [5] Gray, F. Pulse Code Communication. United States Patent Number 2,632,058. March 17, 1953.