Strata: Layered Coding for Scalable Visual Communication

Wenjun Hu*, Jingshu Mao†, Zihui Huang†, Yiqing Xue†,
Junfeng She†, Kaigui Bian†, and Guobin Shen‡

*Yale University †Peking University ‡Microsoft Research

wenjun.hu@yale.edu {scenetree, rogerhzh, alfalfa.xue, plutoshe}@gmail.com

bkg@pku.edu.cn jacky.shen@microsoft.com

ABSTRACT

Existing code designs for display-camera based visual communication all have an all-or-nothing behavior, i.e., they assume the entire code must be decoded. However, diverse operational conditions due to device hardware diversity (in camera resolution and frame rate) and distance range motivate more scalable designs. In this paper, we borrow the notion of hierarchical modulation from traditional RF communication, and design Strata, a layered coding scheme for visual communication. Strata can support a range of frame capture resolutions and rates, and deliver information rates correspondingly. Strata embeds information at multiple granularity into the same code area spatially or the same frame interval temporally. It ensures all layers are decodable independently, by controlling the amount of interference between adjacent layers. Further, our design is recursive and extends readily to generate more layers. Compared with existing codes, it significantly extends the operational range, though at the expense of less capacity than a single-layer code.

Categories and Subject Descriptors

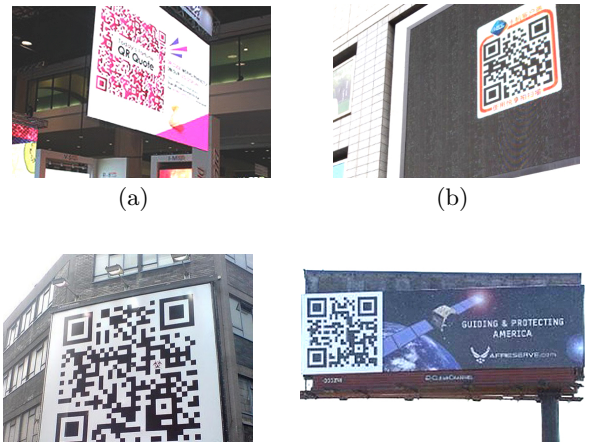
C.2.1 [**Network Architecture and Design**]: Wireless Communication

Keywords

Display-camera links; layered coding; scalable visual communication

1. INTRODUCTION

Displays and phone cameras are widely available, and the quality of both is improving at a fast pace. They offer communication opportunities over the optical channel, where information can be transmitted over the spatial-temporal domains. Even without the temporal domain, a single image can still convey much information. Indeed, 2D barcodes such as QR codes are simple examples that use such a channel. With scanner apps easily available, we can potentially turn any surface into a transmitter. A Google search suggests QR codes are now used widely in advertising (Figure 1), and we expect more innovative usage of similar codes in future. More generally, videos of barcodes can leverage the full spatial-temporal channel.



(a) (b) (c) (d)

Figure 1: QR codes are used in a variety of contexts.

From a communication point of view, however, existing 2D barcodes offer limited flexibility. For example, they only work for a small distance range, and the captured image quality (in terms of size, completeness, clarity) must be sufficiently high to ensure successful decoding, even with forward error correction. They have an all-or-nothing behavior.

In contrast, there are many situations where more flexibility is desirable. At an abstract level, we often see information layered and expressed at multiple granularity. Figure 2 shows a simple example. When viewed at a distance, we only see a big M. When zooming in, however, there is more detailed information about MobiCom within the big M. Many advertisements around us actually follow a similar spirit, often achieved by adopting fonts of different sizes to highlight information of varying importance. We may also acquire only partial information and gather the complete information only when further needs arise, e.g., the name and the phone number in a contact card, the name of a tourist attraction (and the first paragraph of its general description) on an information board, and the name and the manufacturer of some merchandise, as opposed to the full details.

Considering the barcode scanning operation, there are also many justifications for scalable support. The display itself could take a variety of forms (Figure 1), from a sheet of paper

MobiCom'14, September 7-11, 2014, Maui, Hawaii, USA.

http://dx.doi.org/10.1145/2639108.2639132 .

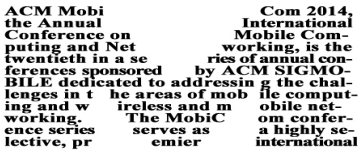


Figure 2: An example sign with information at multiple granularity.

or a wall, both reflecting light and potentially offering low resolutions, to a display, typically LCD or LED, which emits light and can also show dynamic content. The receiving end, the phone cameras, boasts of more diverse capability in terms of the captured image size, resolution, and video recording frame rate. The operational environment is no less simple. A barcode might be indoors or outdoors, under various lighting conditions, and is hardly viewed at a fixed distance away. When a single displayed code is expected to work under diverse conditions, we need a scalable coding mechanism to mimic a one-size-fits-all effect.

There are several challenges to support scalable coding. First, we need to understand the optical channel behavior under different conditions. All the effects can be modeled with some channel distortion. Specifically, pixel colors may mix together spatially (when viewed at a distance) or temporally (when captured at a low frame rate), and we need to handle undersampled signals. Second, we need to design a layered coding scheme that scale with capture conditions and device capability. Third, we need to handle the large dynamic range of color variation and lighting effects as a result of diverse operational conditions.

A previous proposal for hierarchical coding on an LED array of traffic lights [1] is the closest to meeting our requirements. It suggested the potential of accommodating several viewing distances by following a process analogous to multi-resolution image compression. However, it only addresses a subset of the issues we discussed, and involves a computationally intensive wavelet transform.

More generally, we can borrow the notion of multi-resolution coding from traditional RF wireless communication and scalable video coding (SVC) for *channel* and *source* coding respectively. In particular, hierarchical modulation (HM) is a technique that overlays symbols from two or more layers, intended to be decodable at different channel SNRs.

Following a similar spirit to HM, we design Strata, which achieves spatial-temporal layered coding for any input information. Strata embeds information at multiple granularity into the same code area spatially or the same frame interval temporally. It ensures all layers are decodable by controlling the amount of interference between adjacent layers. Spatially, Strata essentially uses code blocks of non-uniform resolutions while bounding the effect of the finer layers on the more coarse layers. Across frames, Strata adopts non-uniform frame intervals, such that a high-rate camera can decode more frames, while a low-rate camera simply treats the extra frames as noise. Furthermore, our design is recursive and can be readily extended to generate more layers. Our experiment results confirm that Strata can indeed scale with a range of capture distances and device capabilities, i.e., resolutions and frame rates. We do trade some amount of per-layer capacity for scaling over a large operational range.

In summary, we make several contributions in this paper:

First, we realize the vision of general multi-resolution visual communication with concrete layered coding schemes, by translating different axes of scalability all into scaling with the captured resolutions. Although there are many existing barcode designs, we are not aware of one that supports scalable coding as outlined in the paper.

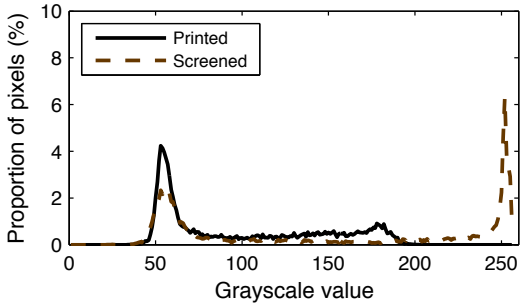


Figure 3: The grayscale value distributions in captured images of printed vs screened codes.

Second, we propose Strata, which adopts non-uniform spatial resolutions and frame rates. Further, the design can be readily extended following a recursive construction.

Third, we explore the design space and show the performance of Strata under various conditions.

2. MOTIVATION FOR SCALABLE CODING

Conventional coding schemes require the code to be received in its entirety, which implies a fixed supported range of spatial resolutions or frame rates. This is the case for each code instance, even if error correction is employed to ease the requirements. However, we face wide-ranging operational conditions in practice, which impose several challenges on the code and related system design.

2.1 Multi-resolution information sources

Examples abound of information expressed or acquired in multiple resolutions. For example, it is common for an advertisement to use various font sizes to highlight different information. When one visits a Wikipedia page, quite often only the synopsis at the top of the page is of interest, rather than the full page. Therefore, papers start with abstracts and news articles start with headlines to highlight the key information. It would be nice to provide a coding primitive to cater to the requirements from the information sources for transmission.

2.2 Diverse operational conditions

Diverse transmitters. As Figure 1 suggests, barcodes are now shown on a wide variety of surfaces. For a printed code, its resolution is constrained by the printing quality, and typical laser printers support 600×600 dpi. The material, especially its smoothness property also affects the amount of light reflected and the perceived resolution of the displayed image. Similarly, LCDs and LEDs also vary in performance. For simplicity, we consider the most common cases, where barcodes are printed on paper or displayed on computer screens. As long as the display resolution is sufficiently high, the captured images will mainly differ in brightness. Figure 3 shows the grayscale value distributions for images captured on a Nexus 5, 5 m from either the printed code or the screened code. The screened code is noticeably brighter, showing a high peak of white blocks.

Table 1: Device specifications.

Device name	Image	Video	CPU	Sensor Aperture
Nokia Lumia 1020	38 MP effective 7152 × 5368 px	1080p@30fps	Dual-core 1.5 GHz Krait	2/3-in F/2.2
Samsung Note 3	13 MP 4128 × 3096 px	1080p@60fps 2160p@30fps	Quad-core 2.3 GHz	1/3.2-in F/2.2
iPhone 5s	8 MP 3264 × 2448 px	1080p@30fps 720p@120fps	Dual-core 1.3 GHz	1/3-in F/2.2
LG Nexus 5	8 MP 3264 × 2448 px	1080p@30fps	Quad-core 2.3 GHz	1/3.2-in F/2.4
Samsung S2	8 MP 3264 × 2448 px	1080p@30fps	Dual-core 1.2GHz	1/3-in F/2.0
Samsung Note 2	8 MP 3264 × 2448 px	1080p@30fps	Quad-core 1.6 GHz	1/3.2-in F/2.6
HTC Incredible	8 MP, 3264 × 2448 px	720p@30fps	Scorpion 1GHz	1/3.2-in F/2.4
iPhone 4	5 MP 2592 × 1936 px	720p@30fps	1 GHz	1/3-in F/2.0
HTC Desire	5 MP 2592 × 1944 px	480p@15fps 720p@30fps	Quad-core 1.2 GHz	1/4-in F/2.8

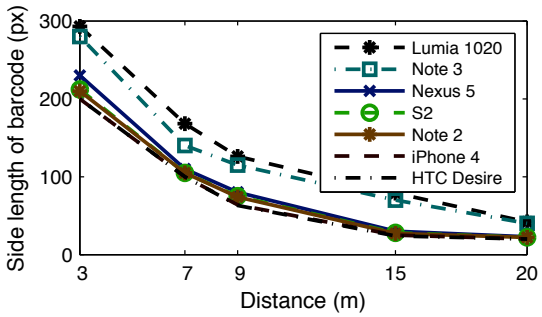


Figure 4: The captured code area size on a phone camera varies with hardware quality and the capture distance.

Diverse receivers. Phone cameras exhibit notable differences in terms of hardware capability, in particular, the image resolution and video frame capture rate. Table 1 lists the specifications for several recent models.

Transmission range. Given a 20×20 (cm^2) barcode displayed on a laptop screen outdoors, Figure 4 shows the sizes of the captured barcode at various distances by the devices. The captured area size drops by almost a factor of 8 from 3 m to 20 m.

The captured image has to be small enough to fit inside the overall captured scene, while large enough to show a sufficient resolution to permit decoding. As a result, a typical QR code image can only support a certain distance range, and cannot be successfully decoded either too close to or too far from it. With a Note 3 to capture a version 13 QR code with 65×65 blocks, Table 2 shows the range determined by the overall displayed code image size. Table 3 further shows the longest distance at which a single-bit monochrome code block of a given size is still decodable using the Note 3.

2.3 Challenges

The above measurements suggest three main challenges to designing a system to cater to diverse conditions.

Understanding the optical channel. All the diverse conditions discussed can be captured by the channel quality, and we need to analyze the channel. In particular, we are dealing with spatial or temporal undersampling, i.e., when the received resolution might be smaller than the displayed, or when the receive frame rate might be lower than needed.

Multi-resolution code design for scalability. Barcode display is a broadcast scenario, where the same display is expected to work for all receiving cameras. Ideally, the higher the resolution of the captured code image, the more information we expect to decode. A high resolution may be achieved with a high-end camera or by capturing the code image at a short distance. We can similarly define *rate scalability*, i.e., the throughput should scale with the frame capture rate of a barcode video over the same time period, the higher the frame rate, the more decoded bits. This is another view of temporal scalability. Therefore, we need a multi-resolution coding mechanism that scales spatially and temporally.

Table 2: QR code sizes and supported ranges.

Image side length (cm)	6.4	15	22	33
Range (m)	0.2–2	0.45–3	0.55–4	0.8–5

Table 3: Capture limit of a single-bit code block.

Block side length (cm, px)	0.31 26 px	0.625 52 px	1.25 104 px	2.5 208 px
Limit (m)	3	5	9	> 20

Handling large dynamic ranges. A side effect from the wide-ranging operational conditions is a large dynamic range of the raw color input. The receiver needs to be able to adapt to the color ranges to ensure correct decoding.

2.4 A leaf from a traditional RF wireless book

For traditional RF wireless, there are also similar issues of designing scalable codes to suit diverse channel conditions. Hierarchical modulation (HM), also called layered modulation, is a technique for multiplexing and modulating multiple data streams into a single symbol stream and transmitted concurrently. Only the base layer can be decoded at a low channel SNR, while the enhancement layer, typically carrying more information, can be further decoded at a sufficiently high SNR. This way, HM offers some scalability at a range of channel SNRs.

Power allocation between layers in HM potentially poses a large dynamic range requirement, which is handled by gain control for the amplifiers for traditional radio transceivers. We face a similar issue.

Therefore, we can borrow similar concepts and techniques from traditional RF wireless for the scalable visual communications scenarios.

3. UNDERSTANDING THE CHANNEL

We start with a simplified view of the channel model, and use that to analyze the capacity. This builds an intuition for the code design. We then empirically measure the channel to understand some system design considerations.

3.1 Channel formulation

Channel unit. Our channel unit is composed of two corresponding square blocks in the displayed and captured images respectively, each containing some number of pixels. This is analogous to the notion of a narrowband single-antenna channel for traditional RF wireless. Symbols are represented by different colors for the displayed code block. For simplicity, we will only consider grayscale color values.

Undersampling. When the captured frame resolution or rate is too low compared to the displayed, we observe undersampled signals. Handling these cases is the key to achieving scalable code design.

The colors of the original blocks mix together when undersampled, either spatially or temporally. Assuming no other distortions, the mixed color is simply a linear average of the original colors [5]. Therefore, the observed signal on our un-

dersampled channel unit is produced by combining the colors of the corresponding displayed blocks adjacent either in time or space. This is analogous to directly adding the channel coefficients of the original channels to obtain that for the composite channel.

The overall channel. Each frame supports many spatial channel units, analogous to MIMO [2], except that the *visual* MIMO channel units are orthogonal, since light propagation is directional. Temporally, each frame corresponds to a time slot, so the code blocks at the same position in different frames simply indicate orthogonal temporal channels.

3.2 Capacity analysis

Since there is no multipath for our optical channels, the total spatial and temporal code capacity is just the per-block capacity summed over all blocks in time and space. We therefore focus on the single-block capacity analysis.

Code block color and miscolor errors. In either the displayed or the received image, we use (u, v) to denote a code block at the u-th row and v-th column.

Each pixel uses k colors to represent $\log\ k$ bits of information (e.g., two colors can represent a single bit, 0 or 1). Color $\kappa \in [0, k-1]$ is assigned to a pixel if the pixel's grayscale value falls within the range of color κ.

Given $b \times b$ pixels in code block (u, v), we determine the color of a code block (u, v) as color κ if this color is assigned to the maximum number of pixels in this code block. In other words, color κ is the *dominant color* among the pixels of this code block.

Incorrectly recognizing a code block color will cause bit errors in the decoding process. We refer to this as a *miscolor error*. Miscolor errors may occur for three reasons. First, the colors of the individual pixels may be determined incorrectly due to hardware-induced pixel noise or ambient lighting effects such as reflections. Second, each pixel in a captured image may be a mixture of multiple displayed pixels in the raw image due to spatial or temporal undersampling. Third, the dominant color of a code block may be incorrectly determined due to the proportion of the pixels showing the expected color.

SNR and capacity. We consider the pixels showing the intended color of the block as *signal*, and any pixel exhibiting a miscolor error for whatever reason (hardware artifacts, ambient lighting effects, or image distortion due to imperfect capture) as *noise*. Let $SNR(u, v)$ denote the signal-to-noise ratio detected in the block of the received image. We can define it as

$$SNR(u, v) = \frac{\alpha(u, v)}{\beta(u, v)}$$

where $\alpha(u, v)$ denotes the number of pixels with the *right* color, i.e., the color of the corresponding block of the displayed image, and $\beta(u, v)$ denotes the number of pixels with the *wrong* color(s) or 1, when no such pixels exist.

Let $c(u, v)$ denote the channel capacity of a code block (u, v). Following the classical RF channel capacity definition per single-antenna channel per time slot for narrowband,

$$c(u, v) = \log(1 + SNR(u, v))$$

This applies generically to the channel regardless of the specific modulation choices. For example, we can use black and white to represent a single bit, 0 or 1, which requires an SNR of at least 1. We can also use four colors to represent 2 bits each, which requires an SNR of at least 3.

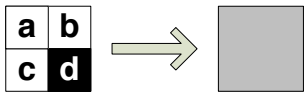


Figure 5: Spatial mixing of blocks.

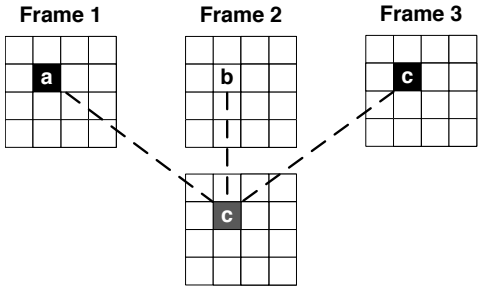


Figure 6: Temporal mixing of blocks.

Multi-layer code capacity. The same capacity argument applies to a multi-layer code with a slight modification to the SNR definition. If our modulation consists of a coarse layer and a fine layer, then the latter is interference when decoding the former. Therefore, the noise term for the coarse layer can count in the contribution from the fine layer.

3.3 Intuition for code design

Based on the capacity expression, we need the SNR$\geq$1 to guarantee 1 bit in the coarse layer. In other words, at least half of the pixels in the coarse layer code block should take the dominant color. Since the input bit distribution for the fine layer is unpredictable, the only way to ensure this color dominance is to always set some of those pixels to the intended dominant color. Hence, we need some notion of a *reserved block*.

Figures 5 and 6 illustrate how blocks mix spatially and temporally to produce a block at the more coarse resolution.

Spatially, we need to reserve some pixels in the fine layer, i.e., we can only use a portion of the fine blocks as effective information blocks. For example, a and b in Figure 5 should both be white to ensure the mixed block is perceived as white.

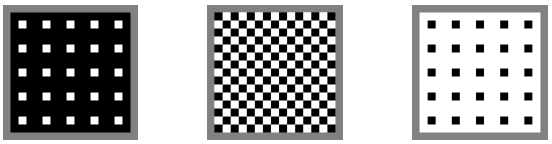
Temporally, we need to reserve some number of frames, or control the frame rate in effect. For example, a and c in Figure 6 should both be black to ensure black dominates in the mixed block.

3.4 Empirical measurements

Practical conditions are more complex than the formulation above, so we next measure the channel empirically and study relevant system effects, mainly the exposure control.

Color mixing. The color mixing behavior is more complicated in practice than simply averaging the colors of the raw pixels. First, light from the raw displayed pixels diffuses over distance, and so the displayed images appear darker (*grayer*) when viewed further away. Second, the average phone camera automatically adjusts focus, contrast, and exposure, whose effects also vary with distance. As it is neither easy nor necessary to separate the individual contributions of these factors, we simply investigate their combined effects on the color mixing behavior.

We display five patterns on the screen for the color mixing test, all black, all white, and the three mixed black and



(a) (b) (c)

Figure 7: Three mixed-color patterns for testing: (a) Units of 1 white block surrounded by 8 black blocks; (b) alternating white and black blocks; and (c) Units of 1 black block surrounded by 8 white blocks.

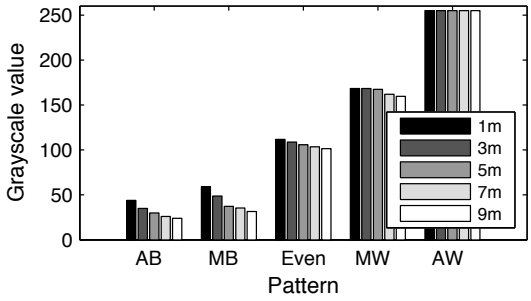


Figure 8: Grayscale values of captured images with different proportions of white displayed. Left to right: all black, mostly black, even split of white and black, mostly white, and all white.

white frames shown in Figure 7. These are captured on the Nexus 5 at several distances. Figure 8 shows the average colors of the individual captured frames, sorted by the display pattern and the distance. For the same distance, we see an approximately linear color mixing behavior as earlier assumed, although the grayscale values for longer distances are smaller (closer to the black end). White is better preserved over distance. For mixed-color frames, the more white blocks, the less the difference across distances.

Exposure control. We make a specially arranged block pattern (Figure 9(a)) on screen to assess the effects of exposure control. Figure 9(b) shows the captured image at 28 m, enlarged to the same size as that of the original image. The captured image becomes noticeably grayer with distance, due to light diffusion. Pixel colors are noisier near color change boundaries than inside the blocks. Further, isolated black blocks can be overridden by the surrounding white blocks at a distance. Therefore, we should avoid isolated black blocks in our encoding and avoid sampling colors at block boundaries for decoding.

4. Strata ENCODING

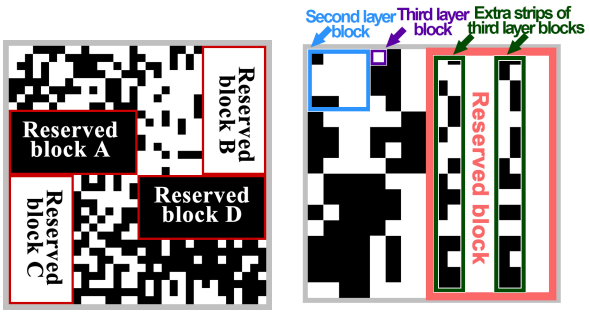
Strata builds on the intuition and system considerations developed previously, and works for either a single barcode image or a video of barcodes.. It is essentially a hierarchical modulation achieved by adjusting the number of code blocks for different layers. This is analogous to adjusting the power allocation between layers in classical HM.

4.1 Assumptions and scope

Strata aims to achieve spatial and temporal scalability as defined in Section 2.3. The latter is translated to *rate scalability*, although we will discuss the relevant operations in the time domain, as they apply across frames. Specifically, Strata is built on three assumptions:

(a) (b)

Figure 9: (a) Original screened image of white and black blocks. (b) Captured image at 28 m, enlarged to the same size as the original image.



(a) (b)

Figure 10: Examples of layered spatial coding. (a) A two-layer code with 4 base layer blocks and 256 enhancement layer blocks within each base layer block. The reserved block can take up different positions and orientations within the base layer block. (b) A three-layer code with bits from the third layer in the reserved block of the first layer.

First, spatially we still capture the entire code image, only that the sizes of the captured images differ, hence the resolutions. In future, we could also consider the case of capturing only a fraction of the overall barcode.

Second, it is fine to receive and decode only *partial* information. This is especially the case for the temporal domain to support streaming. A previous work, LightSync [6], already provides a mechanism for recovering *all* information from a high-rate looping display, where a low-rate camera needs more time.

Third, Strata guarantees the performance of the layer with the most coarse granularity, while opportunistically packing and recovering bits in the finer-grained layers.

As with most barcode designs, we only use black and white for now to represent single-bit symbols. This also means the code design could tolerate more noise than if more colors are adopted to represent multi-bit symbols.

Terminology. Since most discussion centers on how to manage encoding and decoding for two adjacent layers, we will refer to the more coarse layer of the two, layer n, as the *base layer* and the finer layer, layer $n+1$, the *enhancement layer*. Where multiple layers are mentioned, the layer with the finest granularity takes the largest index number.

4.2 Spatial code

The spatial component of Strata is simply a per-frame base code. Therefore, it works regardless of whether the transmitter is a static poster or a display with dynamic content.

Basic recursive code construction. The spatial code is constructed by recursively increasing code block resolutions at successive layers. Figure 10(a) shows an example two-layer code image.

A block in layer n (the *base layer*) is divided into k smaller blocks for the next finer layer, layer $n+1$ (the *enhancement layer*). We refer to k as the scaling factor. A fraction, p, of the enhancement layer blocks form a large *reserved block* carrying the same color as the base layer. The remaining smaller blocks carry data for the enhancement layer. We find that $k = 16$ and $p = \frac{1}{2}$ provide good tradeoffs in terms of the overall performance (Figure 14).

The block division approach aligns pixel boundaries of the blocks of adjacent layers and minimizes the effects of noise at the block edge (noted at the end of Section 3.4), so this simplifies code block detection. Furthermore, this makes it easier to control the color proportion of the composite enhancement layer blocks. Reserving half of the enhancement layer blocks limits the effect of the colors of the remaining enhancement layer blocks on the base layer block regardless. Adopting a contiguous reserved block mitigates the effects of over-exposure (Figure 9) and lets us harness extra bits.

Taken as a whole, Strata produces a barcode with non-uniform spatial resolutions. The reserved block effectively reduces the local resolution to address undersampling.

Additional bits from the reserved block orientation. By using the positions and orientations of the reserved blocks, we can accommodate 2 more bits per group of multi-layer code, as shown by the blocks A-D in Figure 10(a). This requires that there are no contiguous chunks of white or black code blocks in the enhancement layer. To ensure we recognize the reserved block unambiguously, we also apply a *block mask* to the data blocks, such as the one for QR code, to randomize the locations of the white and black blocks. This minimizes the probability of data blocks in the same color accidentally forming the shape of the reserved block. The mask is equivalent to a scrambler for conventional wireless communication systems to avoid long runs of 1 or 0.

Although it might be possible to use other shapes and orientations as well, it would take exponentially more combinations for us to gain each additional bit, and the return for such efforts diminishes very quickly.

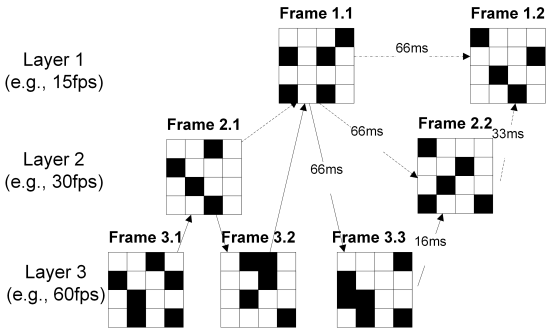
Harnessing extra capacity. Furthermore, code blocks for layer $n+2$ (or $n+3$) can be inserted to the reserved block of layer n. Figure 10(b) shows how two more strips of layer $n+2$ blocks can be added. The capacity of layer $n+2$ (or $n+3$) doubles with these two additional strips.

This process is possible due to the base layer capacity being slightly larger than 1 with high probability, and so we could tolerate more noise in the reserved block for layer n. If we allocate more bits to the immediate next layer, layer $n+1$, the granularity might be too coarse. After another layer, however, the granularity is fine enough that we have more control in how much noise is added to layer n.

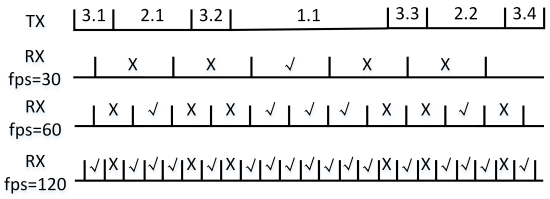
It is difficult to fully use the extra capacity, however, because the information bits could be completely random. We need to be conservative when not knowing the bit distribution of the original message, though this could be relaxed with the addition of error correction bits.

The spatial code parameters, the reserved block size (p) and the scaling factor (k), can be tuned based on the bit value distribution of the information bits and whether error correction is used. We will discuss this more in Section 6.3.

Encoding in the pixel domain vs frequency domain. We follow pixel domain encoding with code blocks for several reasons. First, frequency domain encoding requires high-precision pixels, whereas code blocks are more robust when the display or printing resolution is lower than ideal. Further, our preliminary experiments suggest that code blocks can tolerate slightly blurry captured images caused by hand motion, whereas frequency domain encoding is more sensitive to motion-induced blur. Second, spatial undersampling causes inter-pixel interference in the captured images, which cannot be resolved in the frequency domain. Third, frequency domain processing is computationally more intensive, as shown in previous work [4].



(a) Interleaved temporal layers.



(b) Received frame components.

Figure 11: Example of temporal Strata encoding.

4.3 Temporal code

The temporal component of Strata mainly applies to barcode videos. The base layers are the frames received at the low(est) frame rates.

As shown in previous work [6], such frames are often mixtures of the original frames. If we could control the proportion of the original frames in such a mixed frame, we could follow a similar approach to the spatial code construction. Unfortunately, this is not the case due to typical phone cameras employing a rolling shutter [6]. We therefore construct the temporal code differently, while still following the notion of the *reserved block*.

When the frame capture rate is twice the display rate, we receive a single, decodable frame, alternating with a mixed frame of two consecutive original displayed frames. Since the mixed frame is normally to be discarded, it would not matter even if it is a mixture of more frames. Therefore, we sandwich each base layer frame between two immediate enhancement layer frames. Further layers can be added recursively. Figure 11 illustrates how frames are interleaved at the display and mixed at the receiver. The times on the arrows in Figure 11(a) indicate the perceived intervals between consecutive frames in the layered code. Frame durations are shown schematically in Figure 11(b) following the interleaving. A tick indicates a decodable, single frame, whereas a cross is a mixed frame to be discarded. While Layer 1 frames can be decoded by all receivers, Layer 3 frames can only be decoded by the 120-fps receiver.

Overall, the barcode video essentially adopts non-uniform frame rates. Given that the common frame rate capability varies from 15 to 120 fps (Table 1 and [6]), our target rate for different layers ranges from 7.5 fps to 60 fps.

It is also possible to adopt non-uniform rates within each frame, i.e., a regular frame can be divided into a few parts, with each part following a different uniform frame rate. However, this approach performs worse (Section 6.5) and does not compose easily with the spatial code.

4.4 Spatial-temporal code

By design, our spatial code is an intra-frame code, whereas the temporal code is inter-frame, and the coding mechanisms are orthogonal. Therefore, we can concatenate both to form a layered spatial-temporal code. In other words, each frame of the barcode video can follow the design of the spatial code, with the frames arranged at non-uniform intervals according to the temporal code.

Given the layer correspondence, when using such a code, information for the same layer should be filled across frames.

It is possible to have a more sophisticated design, including more interleaving both intra-frame and inter-frame for error protection. We leave this to future work.

4.5 Frame format

Spatial. As with typical barcodes, we need some mechanisms to locate the barcode image per frame and to identify the code block sizes.

In theory, we could just treat the overall image as a single bit as the first layer and decode from there. In practice, we need a minimum number of bits to transmit useful messages, so we still add some *timing blocks* to indicate the number of first layer blocks present.

Since our code design applies to the data area only, we can simply follow existing corner and timing block designs such as those for QR code or Data Matrix for the most coarse layer. This way, we also inherit any built-in mechanisms for correcting perspective distortion and rotation when capturing the image[1]. There is no need for more dedicated blocks for the finer layers, since we follow the block division rule to encode and detect successive layers. For the spatial part, we could add recursively defined localization blocks to better combat perspective distortion, although we omit these in the current version.

Temporal. For the temporal code, a frame sequence number field is added to each frame to identify individual frames. The block size for this field is the same as that of the most coarse layer in each frame.

5. Strata DECODING

5.1 Exposure control

As mentioned earlier, we need to avoid over-exposure especially when capturing the barcode from afar. Phone cameras are usually set to normal exposure for natural scenes in daylight, but generally support a few additional settings. For the Android camera app, these are -2, -1, 1, and 2. Our measurements show that the optimal exposure settings need to be lower than for natural scenes, hence, typically -2 for screened barcodes and -1 for printed barcodes (Figure 12).

Therefore, we first estimate the average intensity of the first captured barcode image. If the average is too close to the white end, we adopt -2; -1 otherwise.

5.2 Detecting and decoding spatial layers

For each frame, we first detect the image area and the code block size of the most coarse layer following the same procedure as for QR code or Data Matrix. The decoder then determines the color of each pixel within the image area in preparation for the decoding. We also divide the image area into the first layer blocks.

Decoding then proceeds layer by layer. For each block of layer i, the decoder skips the outermost 1/3 of pixels from each side to avoid inaccurate edges. It examines the colors of the remaining pixels of the block, counts the numbers of the black and white pixels respectively, and determines the dominant color. This is considered the color of the block. All blocks of layer i can then be decoded as 0 or 1, which completes decoding for this layer. Each block is then divided in 16 to obtain the blocks for layer $i+1$, and their colors are similarly determined by the majority rule.

The reserved block. We identify the location and orientation of the largest contiguous color chunk to decode the extra bits for the immediate next layer. For the enhancement layer blocks within each base layer block, we assign indices to each block row-wise and column-wise, starting from the top left corner, and calculate the grayscale value of each enhancement layer block.

Say the reserved block is white, containing half of the enhancement blocks. It can take one of the four position and orientation combinations, left, right, top, or bottom. In the row-wise index assignment, the reserved block is at the top (or bottom) if the low-indexed (or high-indexed) blocks show very high grayscale values. Similarly, in the column-wise index assignment, high grayscale values for the low-indexed (or high-indexed) blocks indicate the reserved block is on the left (or right).

The strips for layer $i+2$. The decoding of the added data bits in these inserted strips is the same as that of the original data bits for layer $i+2$.

Stopping rule. If there is error correction, decoding stops if the number of bit errors in the current layer exceeds a threshold, which is set based on the error correction code rate. Otherwise, decoding stops once we reach the pixel layer, i.e., when the code block cannot be divided further.

Note that the information in each layer is independently decodable. In other words, bit errors in layer n have no direct effect on layer $n+1$. Due to the code construction, however, the bit error rate increases with reduced resolution, and therefore we follow a layer decoding order to avoid decoding mostly corrupt enhancement layers. This also has the benefit of simplifying data block localization, or we would need nested timing blocks to indicate the size of data block at each layer.

5.3 Temporal

There is no specific decoder for the temporal part. We simply try to decode any frame captured using the spatial

[1]While the color mixing behavior in different parts of the frame might differ due to perspective distortion, in practice the difference is negligible. When the camera is close to the display and perspective distortion is most noticeable, there is little undersampling. Conversely, when the effect of undersampling is noticeable further away, the effect of perspective distortion is negligible.

decoder. In addition, we decode the frame sequence number of the frame. A frame is discarded if the error rate exceeds a threshold, which indicates unrecoverable frame mixing. If multiple frames with the same sequence number are decoded, we keep results for the best one.

Compared with the spatial decoding process, the temporal process tries to recover all layers as they are received in an interleaved fashion, and then collates frames for the corresponding layers after all frames are decoded.

Note that the temporal process can only decode frames without mixing, and therefore, the color mixing spatially will not be affected further across frames, and we do not need to adjust the reserved block size.

6. PERFORMANCE

6.1 General setup

We follow the setups below unless otherwise stated.

Strata instance. The Strata instance for experiments is composed of four spatial layers per frame, with 1, 10, 160, and 2560 bits respectively. Extra strips of fourth-layer bits were inserted to both of the first two layers. No error correction was included. This can be added easily using an existing algorithm.

The per-frame barcode measures 20×20 (cm^2) when displayed on screen, with the smallest code blocks measuring 26×26 pixels. The printed version measures 21×21 (cm^2).

For the video version, the Strata instance also includes four layers, corresponding to 7.5, 15, 30, and 60 fps respectively, interleaved as described earlier.

Decoder. We implement both an online, real-time decoder as an Android app and an offline version run on a desktop. The latter takes a captured barcode image (in `.jpg`) or video (in `.mp4`) and decodes it using the same algorithm as the app. The offline decoder offers a fairer comparison for the experiments involving the iPhone 5s and the Lumia 1020, and also makes it easier to log and analyze intermediate results[2]. Therefore, we mainly report results from the offline decoder, but uses the online Android decoder to measure timing. We also perform simulations to show the effects of certain spatial parameters without additional channel distortion.

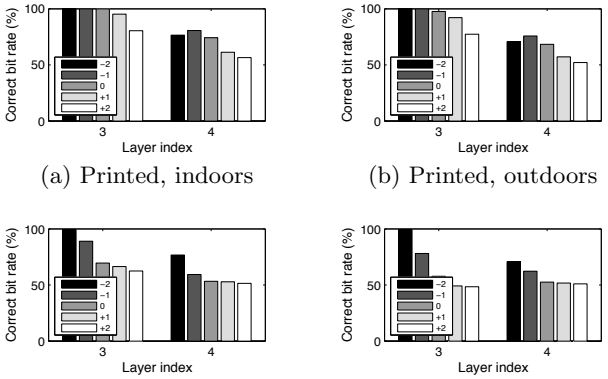
Default experimental conditions. The default receiving phone is the Nexus 5, which has the average camera among all phones that are common on the market. The Lumia 1020 and the iPhone 4 have the best and worst cameras respectively among our devices.

Experiments were mostly run outdoors to capture the barcodes over a large distance range, from 1 m to 28 m. At each distance point, the camera was held as steadily as possible[3]. Each data point is averaged over 5 measurements. As the performance degraded to showing almost random bit errors after 20 m, we omit results for longer distances.

As the spatial and temporal components of Strata are orthogonal, there is little extra to show for the combined version. We will mainly note whether a spatial frame can be decoded fast enough, such that the temporal results would be equivalent to spatial-temporal results.

[2]Logging intermediate results in the app version often makes it too slow to achieve real-time processing for barcode videos.

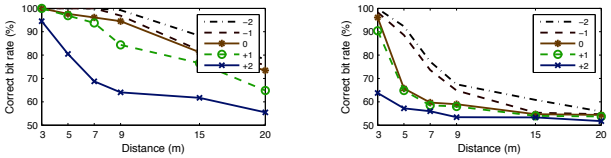
[3]While slight hand shake does not pose an issue as mentioned in Section 4.2, images captured on a moving camera requires deblurring before they can be decoded, which is beyond the scope of this paper.



(a) Printed, indoors (b) Printed, outdoors

(c) Screened, indoors (d) Screened, outdoors

Figure 12: Decoding performance at various exposure settings for printed barcode vs screened, indoors and outdoors.



(a) Decoding performance of the third layer (b) Decoding performance of the fourth layer

Figure 13: Decoding performance for various exposure settings and distances.

6.2 Exposure setting and environment

We start by verifying the exposure settings and any effect from the ambient lighting conditions.

Using the Nexus 5 to capture a barcode 5 m away from where it is displayed, we evaluate the decoding performance at various exposure settings, for a printed barcode or a screened barcode, both indoors and outdoors. Figure 12 suggests that the difference between indoors and outdoors is minor once the exposure setting is adjusted properly. A low exposure setting is better for barcode capture, with printed code favoring -1 and screened code favoring -2. This is further seen in Figure 13 for screened code over a large distance.

Therefore, we only run experiments for screened code outdoors in the following, so that we can manage a large distance range more easily.

6.3 Spatial microbenchmarks

Choice of spatial parameters. There are two main parameters in the spatial code design: *the reserved block size* and *the scaling factor for block division* to proceed to the next layer.

We consider three cases, where the scaling factors between adjacent layers are 1:4, 1:16, and 1:64, i.e., whether there are 4, 16, or 64 enhancement blocks in each base layer block. We set the reserved block to a fixed color (say black), and randomly assign black or white to the remaining enhancement blocks. We then compute the miscolor error rate for the base layer block as the reserved block varies in size. This set of the experiments are done in simulations to show the inherent tradeoff in the code design even without any hardware

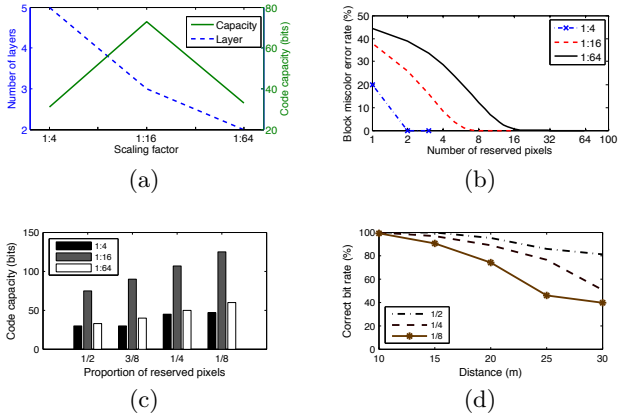
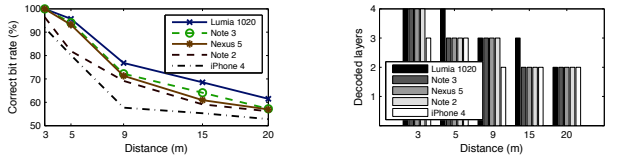


Figure 14: (a) Capacity and layer count tradeoff with scaling factors. (b) Base layer block miscolor rate vs reserved block size for different scaling factors. (c) Capacity vs reserved block sizes for different scaling factors. (d) For 1:16, *measured* miscolor rate at different distances.



(a) Correctly decoded bits across layers (b) Completely decodable layers

Figure 15: Decoding performance across distances.

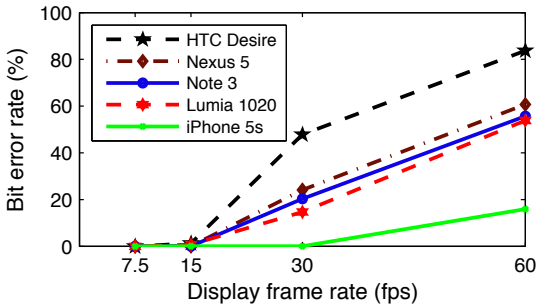


Figure 16: Decoding performance per temporal layer across phones.

distortion in practice. Phone-based experiments confirm the simulation results, so we only present the simulation results unless otherwise stated.

Figure 14 shows several tradeoffs between the scaling factor, the reserved block size, and the total capacity.

We can accommodate more layers with a small scaling factor like 1:4 (Figure 14(a)). However, the reserved block size must also be larger proportionally (Figure 14(b)), and there is more wasted capacity due to the reserved block with each additional layer. Between 1:16 and 1:64, 1:64 needs a smaller reserved block size, while 1:16 supports more layers, and hence a higher capacity overall (Figure 14(a)). This is confirmed by Figure 14(c), which shows that 1:16 achieves the fastest increase in capacity with more layers.

As a byproduct of the scaling factor, we can harness additional bits from the reserved block shape. Assuming a rectangular contiguous reserved block shape, using the current position-and-orientation approach, we get 0 bit for 1:4, slightly over 2 bits for 1:16, and slightly over 3 bits for 1:64. Hence, 1:16 is again a good tradeoff point.

Figure 14(d) shows how the *measured* miscolor error rate of the base layer block varies with the reserved block size for 1:16. This shows that the reserved block needs to be about half unless the enhancement layer blocks are very fine. Since this works per pair of adjacent layers, we do not need to worry about the effect of even finer layers (layers $n+2$, $n+3$, ...) on the base layer.

If we know the data bit distribution or use an error correction code, we could also reduce the reserved block size. In subsequent experiments, we use a conservative reserved block size to guarantee correct decoding.

Extra strips. Given we have decided to use 1:16 and half as the reserved block size, previous figures show that the reserved block is actually larger than necessary. However, the block resolution of the immediate enhancement layer, layer $n+1$, does not permit efficient use of the redundancy in the reserved block. Therefore, we could carve out extra areas for layer $n+2$, where its blocks are 1/64 of the layer n blocks. We can use the results for 1:64 previously to understand how much effect the layer $n+2$ blocks have on the layer n block. Figure 14(b) shows that, for 1:64, a reserved block size of 1/4 is probably sufficient. This means we could repurpose half of the reserved block for layer $n+2$ blocks for extra capacity, hence the justification for the extra strips.

Single-image decoding times. Using the slowest phones, we measure the decoding time of a single frame using the online Android app. For a Strata image captured at 4 m, decoding all four layers after barcode localization takes around 5 ms on the Note 2, and 7 to 8 ms on the S2. The localization step currently takes 52 ms, though could be reduced by adopting a COBRA[4] like corner design.

6.4 Scalability of Strata

Strata is designed to scale with the frame capture resolution and rate, so we assess how its performance scales with the capture distances and hardware capability.

Spatial. Figure 15 shows the performance of a single captured Strata image across devices over various distances. The performance degrades with increasing distance and scales with increasing camera resolution as expected. The Lumia 1020 and the iPhone 4, carrying the best and worst phone cameras, can decode the most and fewest bits correctly. If we consider a layer fully decoded with a BER$\leq$ 3%, the Lumia 1020 can decode more layers at 5 m and 15 m.

Temporal. Figure 16 shows the decoding performance of the Strata video on different phones, measured in bit error rate for each layer. Counting the distinctive frames only, the effective average frame rate is 22.03 fps for the iPhone 5s, 14.30 fps for the Nexus 5, and 7.23 fps for the HTC Desire.

6.5 Comparison with alternatives

We next assess how Strata compares with existing single-layer designs and potential alternative multi-layer designs.

Strata vs existing, single-layer codes. For the temporal component, the conventional approach is to adopt a uniform, low frame display rate that is half that of the lowest frame capture rate supported across devices. For the phones used in the last experiments, this would be 7.5 fps, and the effective frame rates on all phones would also be in

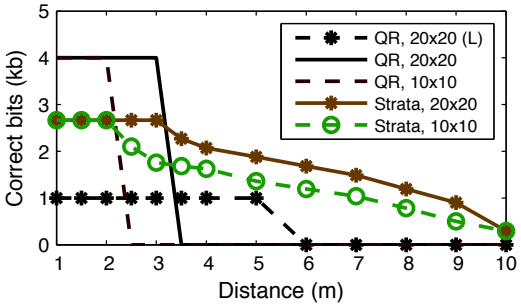


Figure 17: Performance of Strata vs QR code.

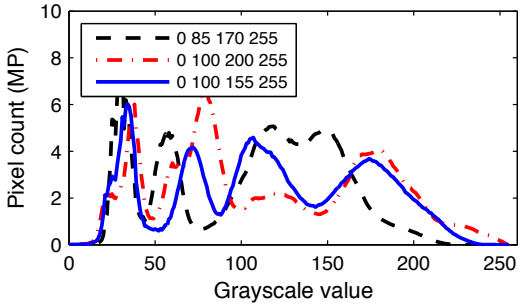


Figure 18: Pixel color distribution of the images of multi-level grayscale designs, captured 1 m away.

the vicinity. Clearly, Strata offers better performance for a high-rate phone. LightSync [6] takes a different approach to frame synchronization and could achieve a higher average rate for a high-rate phone. However, the mechanism does not exactly compare with that in Strata. We will discuss this more in Section 7.

Spatially, it is natural to compare Strata with an existing barcode, such as QR code. We generate a QR code image and a single-image Strata instance with four layers, such that the code area sizes for both are the same[4]. The code block resolution of the QR code is the same as that of the finest, fourth layer of the Strata instance. Therefore, the capacities of the QR code and Strata images are 4096 and 2731 bits respectively. Given the capacity difference, we plot the number of decodable bits at various distances for each in Figure 17, indicated by the suffix "20×20" in the legend. Further, since there is no error protection in the Strata instance, we skip error correction for the QR code as well and count the raw decodable bits. To highlight the tradeoff between the code image size and the decodable distance, we also include comparison of the same images but a quarter of the size (the "10×10" lines) as well as a 1024-bit, 20×20 QR code with bigger blocks, of the same size as the third layer blocks of the 20×20 Strata image (the "QR, 20×20 (L)" line).

As expected, QR code can only support *either* a high capacity *or* a long distance, but not both at the same time. It carries a high capacity at a short distance, but is then completely undecodable. Merely enlarging the overall code image size only increases the farthest distance in small increments. A longer distance can be supported by adopting a larger code block, but sacrificing capacity significantly. Adding further error protection achieves the same qualitative effect as adjusting the code block size, as the latter can be viewed as repetition coding over a block of pixels, the simplest form of protection.

In contrast, a Strata code strikes a balance between capacity and range. Compared with the high-capacity QR code, it offers a much longer range, though at the expense of a lower overall capacity in exchange for guaranteed performance for the more coarse layers. Compared with the larger, low-capacity QR code, Strata still achieves an extended range while offering more than twice the capacity.

Spatial *dithering* vs multi-level grayscales. Using the proportions of black and white enhancement layer blocks to control the overall base layer block color is analogous to *dithering*. A natural alternative is to use pairs of different grayscale values directly to indicate different layers. However, it is very difficult to distinguish more than black and white in practice. Assuming we choose two other grayscale color values in addition to black and white, use them in pairs, and generate a 2-layer code, we can then plot the grayscale value distribution of the received pixel colors. If we could see four distinctive, narrow peaks, then we would be able to calibrate block colors unambiguously to decode the two layers separately. Unfortunately, Figure 18 shows this is not the case for several different choices of grayscale values. In fact, Figure 8 earlier showed that the perceived grayscale values would decrease with distance and cluster near the darker end. Therefore, a multi-level grayscale design would not work[5].

Table 4: Pixel and frequency domain coding.

Error rate (%)	1 m	3 m	5 m	7 m	9 m
Screened (Strata)	0	0	0	0	0
Printed (Strata)	0	0	0	0	1.9
Screened (Haar)	5.4	9.7	12	25.4	47
Printed (Haar)	14.4	26.1	39	41	47.6
Screened (OFDM)	6.2	21	27.7	30.5	48.2
Printed (OFDM)	30.1	32.4	44.9	47	49

Spatial Strata vs frequency-domain coding. Previous work PixNet [12] can achieve a high capacity per frame by encoding in the frequency domain using 2D OFDM. The hierarchical LED array coding [1] also transmits in the frequency domain with a Haar wavelet transform. It divides a 16×16 matrix into four 8×8 regions for information of three priority levels, and then transforms the matrix with Haar wavelets before displaying the result on the LED array. In contrast, single-image Strata encodes in the spatial domain following the reasons outlined at the end of Section 4.2.

We generate a 16×16 Haar wavelet encoding as prescribed, and a three-layer Strata image of the same size. To mimic PixNet, we generate a code image of the same size using 2D OFDM, using only the lowest 20 frequency bands as in PixNet, but without any error correction.

Table 4 shows that frequency-domain encoding suffers much higher decoding error rates than Strata even at small distances. The error rates roughly reflect the decoding performance of the medium-priority layers. The higher the frequency bands, the more sensitive their performance to the spatial undersampling. Figure 19 shows that the high-frequency components very quickly drop below 0.5, the thresh-

[4]This QR code does not follow standard versions as a result.

[5]We find that it is possible to layer two decodable single-layer barcodes using more different colors, but this is equivalent to using four colors per symbol to modulate two bits and cannot be easily extended to more layers.

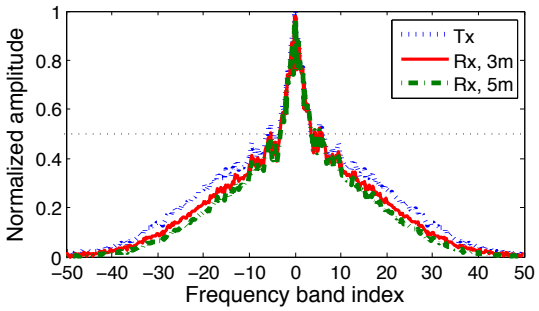


Figure 19: Frequency responses of a printed image encoded with 2D OFDM and captured images.

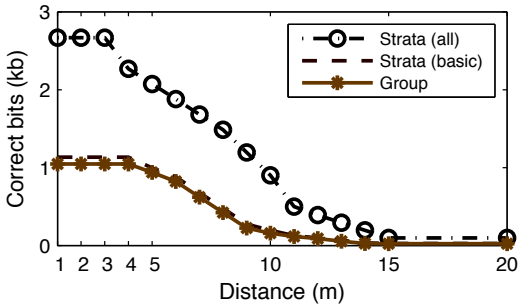


Figure 20: Performance of Strata vs Group of single-layer spatial code.

old for being recognized as the correct color. The results for other distances and for screened code images are similar.

Strata vs group of codes. Another simple alternative design is to have a group of single-layer barcodes of different code block resolutions, such that different parts of the group code will be visible at different distances. We generate such a code example by dividing the total 20×20 (cm^2) code area into four quarters, fitting a single-layer code in each, with 2×2, 4×4, 8×8, and 32×32 blocks respectively, 1108 bits total. The block resolutions for the latter two quarters correspond to those of the third and fourth layers in Strata. We also include a version of Strata without the extra fourth-layer strips or the bits harnessed from the reserved block placement. Its capacity is 1161 bits, compared to 2731 bits for the regular Strata with all the capacity-enhancing features.

Figure 20 shows that all three designs can scale with the distance, with the regular Strata able to carry more than twice as many bits as the group version at all distances. In fact, another main advantage of Strata is its extensibility and faster capacity growth with each additional layer. We can easily devise Strata code instances with five or more layers following the recursive construction. In contrast, it is difficult to similarly pack a group of five or more single-layer codes neatly in one area without tailoring the design to each new layer configuration. Hence, we do not show further comparison results for codes with five or more layers.

We also generate a similar group of barcode videos, essentially dividing a single frame in four, with the four quarters employing different frame rates. Figure 21 shows similar trends to the plots in Figure 16 with slightly different error rates. Therefore, the performance of the two approaches are comparable. However, having the whole frame following the same frame rate makes it readily composable with the spatial layers, and therefore offers more flexibility. Furthermore, it is also harder to extend the quarterly frame design to accommodate more layers.

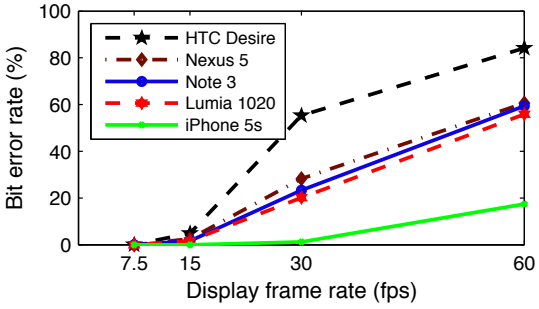


Figure 21: Performance of quarterly frame design for temporal scalability.

7. RELATED WORK

Visible Light Communication (VLC) techniques have been explored over communication links between LEDs (transmitters) and photodiodes (receivers) [10, 11, 2], or between off-the-shelf LCD screens and cameras [5, 12, 4, 6]. Our scheme is also a type of VLC, but works for both static (e.g., a sheet of paper) and dynamic displays.

In particular, hierarchical VLC coding on LED arrays [1] encodes three-level information with Haar wavelets to be captured by cameras at several distances. We have explained the disadvantages of frequency-domain encoding. A further issue due to the Haar wavelet transform is that decoding errors in the lower priority layers will propagate to the higher priority layers. In comparison, Strata also aims to deliver information at different resolutions, but more generally covering spatial and temporal domains with a simple, recursive design and independent layers.

Screen-camera based VLC systems. Several screen-camera based systems employ one-way video streaming for information transmission.

PixNet [12] is a high-capacity code optimized for high-resolution displays and cameras. Given the high decoding complexity, it is not suited to real-time streaming on phones. COBRA [4] adopts a special code layout supporting fast corner detection and code scan to achieve real-time decoding of barcode streams and some blur-resilience. LightSync [6] addresses imperfect frame synchronization as a result of the frame rate mismatch between the display and the camera. Through in-frame color tracking and an inter-frame linear erasure code, LightSync supports a much higher frame display rate than permitted with conventional sampling approaches. 4D barcode [8] also targets synchronization by repeating 2D frames on multiple, orthogonal color channels, which effectively reduces the display frame rate. VRCodes [15] encode over all dimensions of color, time, and space while remaining unobtrusive to the human eye. Decoding is possible with a rolling-shutter camera.

In contrast, Strata follows a different goal of scalable code design, by adopting layered coding structures both spatially and temporally. For the temporal part, we share with LightSync in achieving effective frame rates that scale with the camera's hardware capability. However, the two systems follow different philosophies. LightSync tries to recover undersampled signals over time, but Strata aims to remove undersampling locally. LightSync relies on a looping display to effectively slow down the display rate, whereas Strata works in a streaming mode to decode potentially partial information at all times. LightSync achieves a higher rate per unit time, but only after a minimum amount of capture time.

Barcode design and detection. A vast majority of existing work focuses on better detecting the barcode or redesigning the barcode for higher throughput, better user experience, or more robustness. All are intra-frame codes. We mainly focus on multi-layer designs.

PiCode (or ViCode) [7] takes a watermarking like approach to embed pictures (or videos) into the code. DualCodes [14] superpose a color barcode onto a black-and-white barcode to achieve a higher spatial information density. These approaches typically accommodate at most two layers of information, and do not exhibit the same scalability properties we aim for. In contrast, Strata supports more layers and allow information to be extracted at different granularity.

Recursive 2D barcode [13] is designed for transmission in the near infrared band to be hidden in a map. It recursively encodes simple map location indicators viewable at several distances, but with low information densities. Strata follows a more general goal and employs tightly packed code blocks spatially to achieve a much higher capacity.

Related concepts in source coding and traditional wireless. As we mentioned earlier, the notion of scalable, layered coding is borrowed from traditional wireless communications, especially hierarchical modulation as an example of scalable *channel* coding. SVC is among the best known examples of scalable *source* coding, and more recent proposals include [9, 3] on 3D camera sensing and [16] on image coding, to name a few.

Although SVC is also a type of multi-resolution coding, it follows a fundamentally different mechanism. The cause of distortion is lossy compression via the quantization of analog signals to digital representations, which can be controlled at the encoder by analyzing source information distribution. In contrast, HM and Strata are subject to channel distortions that are unknown at the encoder. Therefore, the encoder can only estimate the channel condition and needs to be sufficiently conservative to account for a range of possible channel distortions to ensure decodability.

Note further that the layers in HM or SVC are dependent, i.e., the enhancement layer would be useless if the base layer is not correctly decoded. In contrast, the layers in Strata are independent for more robustness.

8. CONCLUSION

As cameras become widely available along with mobile devices, increasingly they are capable of acting as the receiving end of communication over surface-camera or display-camera links. There are diverse operational conditions due to device diversity and distance ranges. However, existing coding schemes do not cater to these.

In this paper, we propose Strata, a layered coding scheme to achieve scalable performance, spatially with the frame capture resolutions and temporally with the frame capture rates. Strata borrows the notion of hierarchical modulation from traditional wireless communications. Our scheme embeds information at multiple granularity into the same code area spatially or into the same frame interval temporally. It ensures all layers are decodable by controlling the amount of interference between adjacent layers. Furthermore, our design is recursive and extends easily to add layers. Our experiment results confirm that Strata can indeed scale with a range of capture distances and frame rates. We do trade some amount of per-layer capacity for scaling over a large operational range.

The current capacity increase behavior still follows a few discrete steps. There is certainly room for improvement to achieve more graceful increases.

Furthermore, we essentially focus on scalable *channel* coding and provide a primitive for degradable communication. To better utilize such codes, we also need scalable *source* coding. This is analogous to layering scalable video coding over hierarchical modulation. We leave it to the application to partition the raw information to the effect of hierarchical or scalable source coding.

9. REFERENCES

[1] S. Arai, S. Mase, T. Yamazato, T. Endo, T. Fujii, M. Tanimoto, K. Kidono, Y. Kimura, and Y. Ninomiya. Experimental on hierarchical transmission scheme for visible light communication using LED traffic light and high-speed camera. In *Proceedings of VTC Fall*. IEEE, 2007.

[2] A. Ashok, M. Gruteser, N. Mandayam, J. Silva, M. Varga, and K. Dana. Challenge: Mobile optical networks through visual MIMO. In *Proceedings of MobiCom*, 2010.

[3] I. Daribo. Hierarchical requantization of depth data for 3D visual communications. In *Proceedings of International Conference on Digital Information and Communication Technology and its Applications (DICTAP)*, pages 104–109. IEEE, 2012.

[4] T. Hao, R. Zhou, and G. Xing. COBRA: Color Barcode Streaming for Smartphone Systems. In *Proceedings of MobiSys*, 2012.

[5] S. Hranilovic and F. R. Kschischang. A Pixelated MIMO Wireless Optical Communication System. *IEEE Journal of Selected Topics in Quantum Electronics*, 2006.

[6] W. Hu, H. Gu, and Q. Pu. LightSync: Unsynchronized visual communication over screen-camera links. In *Proceedings of MobiCom*, pages 15–26. ACM, 2013.

[7] W. Huang and W. H. Mow. PiCode: 2D barcode with embedded picture and ViCode: 3D barcode with embedded video. In *Proceedings of MobiCom*, pages 139–142. ACM, 2013.

[8] T. Langlotz and O. Bimber. Unsynchronized 4D Barcodes Coding and Decoding Time-Multiplexed 2D Colorcodes. In *Proceedings of the International Symposium on Visual Computing*, 2007.

[9] S. Lee, J. Choi, S. Oh, J. Ryu, and J. Park. A real-time 3D IR camera based on hierarchical orthogonal coding. In *Proceedings of IEEE ICRA*. IEEE, 2006.

[10] M. Nakagawa. The world of visible optical communication - It opens with LED — Light Communications. *Kogyo Chosakai Publishing, Inc.*

[11] G. Pang, T. Kwan, H. Liu, and C. Chan. LED Wireless. *IEEE Industry Applications Magazine*, 2002.

[12] S. D. Perli, N. Ahmed, and D. Katabi. PixNet: LCD-camera pairs as communication links. In *Proceedings of MobiCom*, 2010.

[13] D. Reilly, H. Chen, and G. Smolyn. Toward fluid, mobile and ubiquitous interaction with paper using recursive 2D barcodes. In *Proceedings of Workshop on Pervasive Mobile Interaction Devices*, 2007.

[14] M. Werner and M. Schönfeld. DualCodes: Backward Compatible Multi-layer 2D-Barcodes. In *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pages 25–36. Springer, 2013.

[15] G. Woo, A. Lippman, and R. Raskar. VRCodes: Unobtrusive and Active Visual Codes for Interaction by Exploiting Rolling Shutter. In *IEEE International Symposium on Mixed and Augmented Reality*, 2012.

[16] S. Zhu and B. Zeng. A novel enhancement for hierarchical image coding. *Journal of Visual Communication and Image Representation*, 24(1):12–22, 2013.