Years-Long Binary Image Broadcast using Bluetooth Low Energy Beacons

Chong Shao, Shahriar Nirjon, Jan-Michael Frahm
Department of Computer Science
Univerisity of North Carolina at Chapel Hill
Chapel Hill, NC 27599
Email: {cshao, nirjon, jmf}@cs.unc.edu

***Abstract*—This paper describes the first 'image beacon' system that is capable of broadcasting binary images over a very long period (years, as opposed to days or weeks) using a set of cheap, low-power, memory-constrained Bluetooth Low Energy (BLE) beacon devices. We design a patch-based image encoding algorithm to produce encoded images of reasonably high quality, having sizes of as low as 16 bytes – without any prior knowledge of the test images. We test our system with different types of images that contain hand-written alphanumeric characters, geometric shapes, and arbitrary binary images having complex shapes and curves. We empirically determine the tradeoffs between the system lifetime and the quality of broadcasted images, and determine an optimal set of parameters for our system, under user-specified constraints such as the number of available beacon devices, maximum latency, and life expectancy. We develop a smartphone application that takes an image and user-requirements as inputs, shows previews of different quality output images, writes the encoded image into a set of beacons, and reads the broadcasted image back. Our evaluation shows that a set of 2–3 beacons is capable of broadcasting high-quality images (75%–90% structurally similar to original images) for a year-long continuous broadcasting, and both the lifetime and the image quality improve when more beacons are used.**

I. Introduction

Lifeless, passive everyday objects have started to become smarter in this age of the Internet of Things. Advancement in low-power microcontroller and wireless technology, miniaturization of circuitry, lower cost of fabrication, and higher surge in use cases are making it possible today to literally glue tiny computers to everyday objects – so that they can sense, react, and tell their own stories. The industry has embraced wireless standards such as Bluetooth Low Energy (BLE) [3], and developed protocols such as iBeacon [8], in order to create programmable 'beacon' devices that periodically broadcast a small amount of preloaded data, and last for multiple years [11] on a coin-cell battery. Broadcast messages from beacon devices typically contain information about an object, a location, a web-resource, or just an arbitrary string. These messages are received by a BLE capable mobile device to obtain relevant information – just-in-time and on-the-spot.

Emerging applications of beacon devices include advertising merchandise in retail stores [9], identifying late passengers at the airports, authorizing people at the hospitals, smarter signage, indoor navigation [6], and tracking moving platforms like airline cargo containers, computers on wheels, museum artworks, or even humans [2]. The enabling technology behind these applications is the ability of a beacon device to simply broadcast a few bytes of data (called UUID) as BLE advertisement packets at a rate of less than 16 bytes/sec. The bound in data rate comes from the lifetime requirement of these devices. Such tight budgets on payload and maximum data rate has limited a beacon device's capability to only be able to broadcast an identifier or a small amount of text (about 16–18 bytes). To transmit a moderate sized image, either we require to use hundreds of beacon devices, or we will have to accept a very long transmission delay.

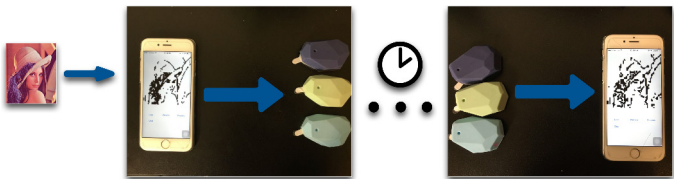

Fig. 1. The image beacon system showing a mobile device that writes and reads the binary image ('Lenna') into three BLE beacons. The expected lifetime of this system is two years (for 3 beacons, and 2 seconds latency).

Hypothetically, if we could broadcast high-resolution images from a beacon device in real-time, the technology would enable even more powerful and feature rich applications. Like the web has evolved from serving hypertexts to streaming multimedia contents, we envision that the natural successor of a beacon device would be the one that broadcasts images, while meeting the same energy and lifetime requirement. Applications of such an image beacon system would be in scenarios where there is no Internet connectivity but there is a need for storing and broadcasting information that can be best described by an image. Potential applications of beacon image systems include coordinating rescue workers in disaster areas, creating a bread-crumb system for adventurous hikers and mountaineers, remote surveillance (when coupled with a camera), or even a simple system just to let someone know that 'I was here'.

In this paper, we chase this seemingly impossible goal of creating a beacon device that efficiently broadcasts images over a long period. As a first step toward realizing an image beacon, we explore the challenges to broadcasting binary images of different categories (e.g., alpha-numeric characters, basic shapes, and arbitrary binary images), and design algorithms to efficiently store contents of an image inside a set of beacon

devices. The set of beacons simultaneously broadcasts chunks of an image over BLE, which are captured by a mobile device to reconstruct the image. A fundamental challenge toward achieving this is to efficiently represent an image using as few bits as possible. Standard image compression algorithms are not good enough to archive the required compression ratio so that an image can be stored inside a beacon. We investigate image approximation/coding techniques that take into account the limits on number of beacon devices, number of bits available in a beacon device, data rate, latency, and lifetime. Based on empirical analysis, we devise a patch-based image approximation algorithm which greatly reduces the image data while keeping the image distortion under a threshold. We investigate the tradeoffs between the image quality and the power consumption to determine the best set of parameters for the system under user-specified constraints.

We have developed a prototype of an image beacon system using a set of commercially available Estimote beacons [1], and developed an iPhone application that takes an image along with user-specified requirements and constraints on broadcasting the image as inputs, generates previews of the image to be written, writes the image representation into a set of beacons, and reads the broadcasted image back. Figure 1 shows the complete system where a writer application on the phone takes a standard JPEG image, converts it to binary format, and shows the user a preview of the compressed image to be written. The user is allowed to change the settings (e.g., the number of available beacons and/or expected device lifetime) and the app immediately shows the best possible compressed image under these constraints. The application writes the image data into the beacons and the image is broadcasted by the beacons. A reader application reads the broadcasted image and displays it on the phone.

We perform an in-depth evaluation of the system. We describe a set of results showing the tradeoffs between device lifetime and image quality, when image type, number of beacons, and other system parameters are varied. We also deploy the image beacon system and perform user studies in two real-world scenarios where a group of participants (1) use the smartphone application to draw, preview, write into, and read from a set of image beacons, and (2) use a beacon guided navigation system in a multi-floor indoor setting.

The main contributions of this paper are as follows:

- To the best of our knowledge, we are the first to propose an image beacon system that uses multiple BLE beacons to broadcast binary images over the BLE advertisement messages.
- We have devised a patch-based image approximation algorithm that greatly reduces the image sizes. We quantify the tradeoffs between the image quality and the device lifetime, and determine the best set of parameters, under the user-specified constraints on the number of beacons, latency, and expected system lifetime.
- We have developed and evaluated a prototype of an image beacon system that broadcasts binary images of various types (e.g., alpha-numeric characters, basic shapes, and arbitrary binary images). Our evaluation shows that a set of 2–3 beacons is capable of broadcasting high-quality images (75%–90% structurally similar to original images) for a year-long continuous broadcasting, and both the lifetime and the image quality improve when more beacons are used.

II. Problem Formulation

We propose a self-contained image beacon system that is capable of storing and broadcasting contents of a binary image, without the aid of any additional sources of information about the image. The image is either a photo taken with a camera which has been converted to a binary image, or an image containing basic shapes, or a hand-drawn image by the user on his smartphone's touchscreen. The image data will be written to and read from the image beacon system using a smartphone application. We assume that the system is self-contained, i.e., no additional information about the broadcasted image is available from any other sources globally (on the web) or locally (on the smartphone).

The problem is formally stated as: given a binary image x (i.e. each pixel is represented by one bit) having the dimensions of $\mathrm{N} \times \mathrm{M}$ bits, the number of available beacon devices K, the payload size of each beacon packet C bytes, the maximum allowable broadcast rate of R packets/sec, and the maximum allowable latency for an image T, the objective is to find an approximate representation of the image $\hat{\mathrm{x}}$ so that the lifetime τ of the beacon system is maximized while the approximation ratio $\lambda(\mathrm{x}, \hat{\mathrm{x}}) \in [0, 1]$ of the image is high ($\lambda = 1$ means no distortion).

Now, for a single beacon, the broadcast rate:

$$\mathrm{R} = \left(\frac{\mathrm{NM}}{8\mathrm{C}}\right)\frac{1}{\mathrm{T}} \tag{1}$$

For K beacons, considering $\log K$ overhead bits for addressing the beacons, and K times more payload capacity:

$$\mathrm{R} = \left(\frac{\mathrm{NM} + \log \mathrm{K}}{8\mathrm{CK}}\right)\frac{1}{\mathrm{T}} \tag{2}$$

Both (1) and (2) are for undistorted images.

The lifetime τ of a BLE device depends on its inter packet interval and in general, $\tau \propto \frac{1}{\mathrm{R}}$. Replacing R and incorporating approximation ratio λ into (2):

$$\frac{1}{\tau} \propto \left(\frac{\lambda\mathrm{NM} + \log \mathrm{K}}{8\mathrm{CK}}\right)\frac{1}{\mathrm{T}} \tag{3}$$

The above equation relates the lifetime of an image beacon system and the approximation ratio of any image compression algorithm. In this paper, we devise a patch-based image approximation algorithm that achieves a sufficiently large λ for a reasonably high lifetime of the system.

III. Challenges

There are two major challenges in designing an image beacon system. First, the limited payload size of beacon packets, and second, the limited bandwidth of BLE.

A. Limited Beacon Payload

The maximum payload size C available in beacon devices is 18 bytes (from the specification). However, there are 33 special characters which cannot be read from the beacon devices since they are control characters. So, practically the payload size is $\lfloor \log(256-33)^{17} \rfloor \approx 132$ bits ≈ 16 bytes. On the other hand, the size of a jpeg compressed 64×64 pixels binary image is typically between 400 bytes to 2 KB, depending on the quality settings. Therefore, to transmit such an undistorted binary image, a beacon would require $25-125$ broadcast packets, or alternatively, we would require up to $K = 125$ beacon devices to simultaneously broadcast different slices of an image. By using an image approximation algorithm having a sufficiently high compression ratio, the required number of beacons can be reduced significantly.

B. Limited BLE Bandwidth

The main advantage of BLE over any other wireless protocols is its extremely low-energy packet transmission capability. This is achieved by aggressively maximizing the sleeping interval and sending data packets at a much lower rate than classic Bluetooth's.

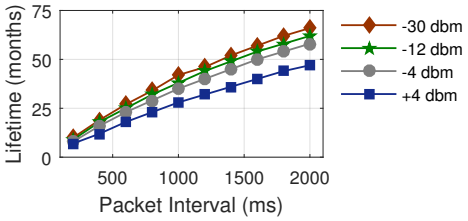

Fig. 2. The lifetime depends on packet transmission rate and signal strength.

Figure 2 shows the expected lifetime of Estimote BLE beacons for various inter-packet interval. For example, the beacon would last up to 3.5 years if a packet is sent at every second (i.e. R = 1). Therefore, for a beacon system that lasts for 3.5 years, its broadcast bandwidth is bounded to the maximum limit of 16 bytes/sec, and the latency of a complete image transmission cycle would be 125 seconds for a single beacon, or 1 second for a set of 125 beacons. Again, by using an image approximation algorithm having a sufficiently high compression ratio, this latency can be reduced significantly.

IV. Algorithm Design

The process of converting an input image to its approximate equivalent is described in this section. The process has an one-time, offline phase where an 'alphabet' of carefully designed 'patches' are generated. During the on-line phase, the input image is encoded using these patches in order to generate a reduced version of it, which is suitable for writing into the beacons. Figure 3 illustrates the overall process.

A. Offline Processing: Patch Set Generation

In order to store an image into a limited amount of storage, one has to make a tradeoff between image variety and the quality of the compressed image. By limiting the image type to binary images that contain mostly curves, we show that it

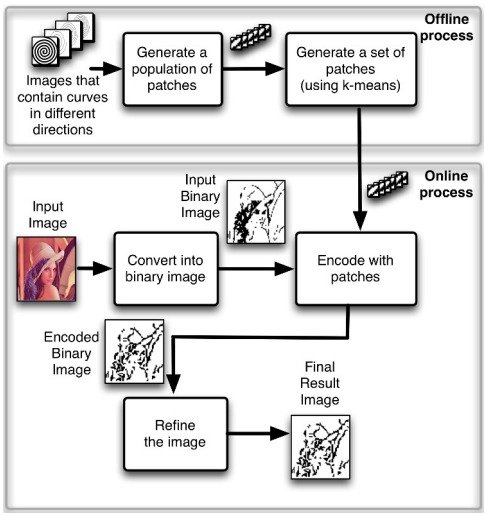

Fig. 3. Beacon image processing pipeline.

is possible to design a predetermined set of patches, which can represent the image in a way that the encoded image is extremely compressed, while it's of high quality. The encoding of an image is based on an agreement on the set of patches between the encoder and the decoder. Hence, only the indices of the patches are needed to be stored/broadcasted in order to encode/decode an image. An immediate question is therefore: *how do we design a suitable set of patches?*

There are two main considerations in designing the set of patches. First, determining the number of patches to use, and second, determining the content of each patch. The set of patches should be general enough to be able to represent a wide variety of binary images. On the other hand, the size of the set should not be too large, otherwise, the number of bits required to encode the patch indices will be large, resulting in larger images. A rule of thumb in designing base elements for images is to let the set of patches be rich enough for an input image so that– *given any sub-region within the image that has the same size as that of a patch, there is always patch in the set whose texture is roughly the same as the sub-region's texture, with a very high probability*.

Since we have limited the image type to binary images, we seek to design a set of patches that contains binary textures of varies types and curves in different directions. The two-step process of generating the set of patches are as follows.

Step 1. Generating Patch Population: A simple approach to generating a population of patches is to divide a spiral image using a $g \times g$ grid to obtain a total of g^2 patches. We use spiral images since they are easy to parameterize. By having a set of parameterized spirals we can easily control the curvature and the direction of the curves in the patches. Figure 4(a) shows an example of a spiral in a 3×3 grid. This approach, in general, produces a diverse set of patches containing curves of different orientations and directions. However, the patches are biased by the choice of the original spiral. Because of this, we choose to start from a set of binary images, where each image contains a parameterized spiral whose parameters are different from other spiral images. Figure 4(b) shows an example where we have

m spiral images. The parametric equation of the n^{th} spiral is: $\mathrm{x} = \mathrm{t}\sin(\mathrm{t}),\ \mathrm{y} = \mathrm{t}\cos(\mathrm{t}),\ 0 \leq \mathrm{t} \leq \mathrm{n}\pi$. Each of these n spirals is divided using a $\mathrm{g} \times \mathrm{g}$ grid to obtain a population of $\mathrm{g}^2\mathrm{n}$ patches. Let us denote the population of patches as $\{\mathrm{P}_{\mathrm{i,j}}\}$, where $1 \leq \mathrm{i} \leq \mathrm{n}, 1 \leq \mathrm{j} \leq \mathrm{g}^2$.

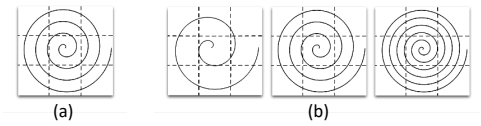

Fig. 4. (a) single spiral, (b) multiple spirals.

Step 2. Selecting Patches using k-means: Given a set of test images $\{\mathrm{X}_\mathrm{i}\}$, $1 \leq \mathrm{i} \leq \mathrm{l}$, the goal is to find a K-sized optimal subset $\mathrm{S} \subset \{\mathrm{P}_{\mathrm{i,j}}\}$, so that the sum of distances of the fitted image to the original image is minimized, i.e.–

$$\underset{\mathrm{S}}{\operatorname{argmin}} \sum_{\mathrm{i}=1}^{\mathrm{l}} \sum_{\mathrm{j}=1}^{\mathrm{g}^2} \min_{\mathrm{s_k} \in \mathrm{S}} \left\{ \mathrm{d}(\mathrm{x_{i,j}}, \mathrm{s_k}) \right\} \tag{4}$$

where, $\mathrm{s_k} \in \mathrm{S}$ denotes a selected patch, $\mathrm{x_{i,j}}$ is the j^{th} patch-sized subregion of image $\mathrm{X_i}$, and $\mathrm{d}(.)$ is a distance function that measures the fitness of a patch to a subregion of an image.

Finding an optimal subset of S for any arbitrary distance function is in general an NP-complete problem. Hence, we employ a clustering-based approach where we use k-means algorithm to cluster the population of patches into K clusters. We use the structural similarity metric (SSIM) [12] as the distance function. The intuition behind this approach is that, given the distance metric, as long as the distribution of patches in S resembles that of the unknown images, k-means will select a near optimal subset which minimizes Equation 4. More specifically, had we used patches from actual test images, k-means would be highly likely to find an optimal subset of patches that best fits the images.

B. Online Processing: Encoding and Refinement

During the online phase, every new image at first is converted to binary images based on a color threshold, and then it is encoded (using the selected patches from the offline phase) and is refined to improve the quality of encoding, prior to writing it into the beacons.

We use a simple, fixed-length encoding scheme to describe each input image as a sequence of patch identifiers. Non-overlapping, patch-sized regions of the input image are sequentially accessed, and for each region, the patch (within the set of selected patches) that has the maximum structural similarity to the region is noted, and its index is stored in a queue. When all the regions of the input image have been processed, the queue contains the encoded image. This simple encoding scheme can be further improved if we have prior knowledge on each patch's probability of occurrence. For example, by assigning short-length codes to more frequent patches, the overall length of encoded image could be reduced. To keep things simple and generic, we do not employ such a variable-length encoding approach in this paper.

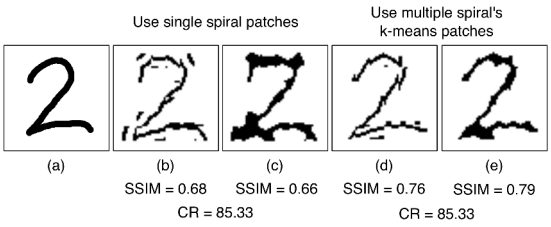

Fig. 5. (a) original image; (b) result from patches generated from single spiral image; (c) result image from patches generated from single spiral image after morphology refinement; (d) result image from patches generated from multiple spiral images and k-means; (e) result from patches generated from multiple spiral images and k-means after morphology refinement.

After encoding an image, two standard mathematical morphology operations [10] –dilation and erosion– are applied to enhance the quality of the resultant image. Examples of images before and after these refinements are shown in 5. The parameters of these operations are their 'operator sizes' (in pixels), which depend on the curve-width of the input image and the patch set. In our experiments, we found that the resultant image has its best quality when the erosion/dilation operator sizes are 3 pixels.

C. Image Decoding

To decode an image, the broadcasted patch indices from all the beacons are received and serialized by the decoder application. The image is reconstructed by arranging the patches in the correct order as dictated by the index sequence. The refinement process is applied in the decoder as well.

V. Empirical Evaluation

In this section, we describe a series of empirical evaluations. At first, the patch-based image compression approach is compared with JPEG encoding. Then we describe a set of results that quantifies the tradeoffs between the device lifetime and the image quality, when the type of images, number of beacons, patch function generation method, number of patches, and the grid or patch size are varied. We also perform full system evaluations involving real users and multiple use cases – which are described in the next section.

A. Experimental Setup

In all of our experiments, we have used Estimote model Rev.D3.4 Radio Beacons [1] having a 32-bit ARM Cortex M0 CPU, 256 KB flash memory, 4 dBm output power, 40 channels (3 for advertising), and 2.4–2.4835 GHz operating frequency. We vary the BLE broadcast interval for a beacon between 100 ms to 2,000 ms. However, an encoded image (broadcasted from multiple beacons) reaches a user's device in less than 1 second. The transmission power is set to -12 dBm, which limits the range of each beacon to about 30 meters. The image writing and reading application runs in an iPhone 5s having an ARM v8 based dual-core 1.3 GHz Cyclone CPU, Apple A7 chipset, 1 GB DDR3 RAM, BLE v4.0, and runs iOS 9.2.

We use three types of images in our experiments images containing alpha-numeric characters, basic shapes, and arbitrary binary images. Examples of these images are shown in Figure 6. Some of these images are directly drawn on the

phone by a user (e.g., letters, numbers, and free hand drawings), while some are real pictures that have been converted to binary format by our writer application. All images are downsampled to 64×64 pixels prior to writing.

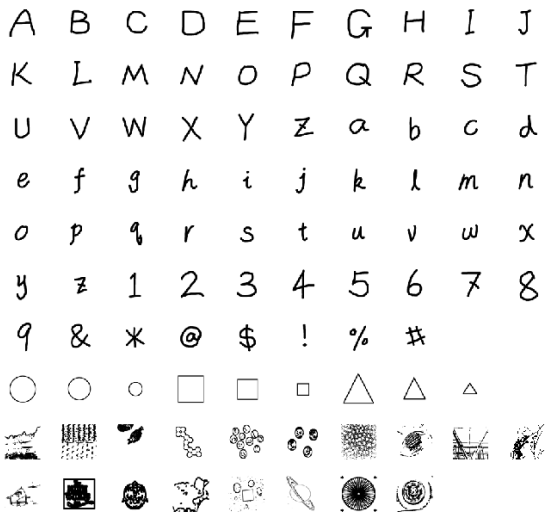

Fig. 6. Test images used in the empirical evaluation.

The two main metrics that are used in the experiments are structural similarity (SSIM) scores, and device lifetime in months. We measure these two under different conditions and show their tradeoffs. The structural similarity scores are used to measure the quality of the produced images when compared to the original ones. The device lifetime is estimated from its relation to a beacon's transmission frequency. Before each experiment, we program the beacons to set a transmission frequency and use the corresponding estimated device lifetime (as reported by the Estimote beacon API) in our experiments.

B. Comparison with JPEG

In this experiment, we compare the proposed patch-based image encoding algorithm with JPEG. We pick the image of a handwritten '2' as test images. This is also used in Section V-D, V-F, and V-G. In Figure 7, we plot the quality of encoded images obtained from three methods – two versions of the k-means patch-based approach (4 and 8 pixels per patch) and JPEG, for various sizes of images. The result suggests that– JPEG generates better quality images in general, however, it also requires larger sized images. For a 64×64 binary image, JPEG encoded image is about 464 bytes, even with the lowest quality settings. Compared to that, our patch-based method generates images in smaller sizes (30–200 bytes), making it possible to store an image inside a small number of beacon devices.

Besides JPEG, we also studied several other image compression techniques [7][13][14]. But none of these yield suitably small sized images.

C. Effect of Image Type

We conduct an experiment to measure how the patch-based image encoding method's performance (in terms of quality) changes when the type of input images is varied. The three types of images we use include hand-written alpha-numeric characters, basic geometric shapes, and arbitrary binary images containing complicated shapes and curves. For each type, we compute the average image quality for a given lifetime. We use three beacons in this experiment to store and read images.

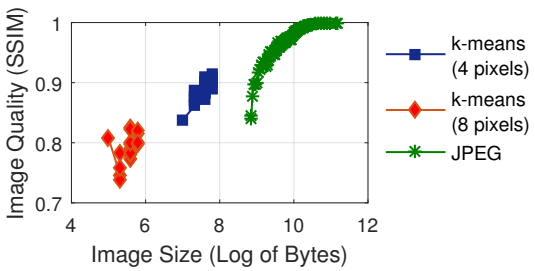

Fig. 7. Image quality versus image size for different encoding methods.

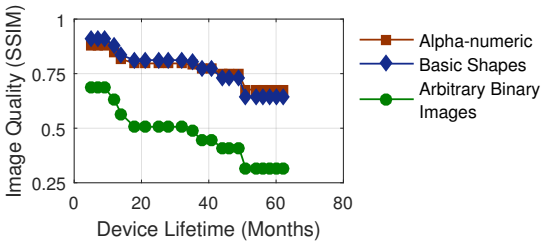

Fig. 8. Image quality versus beacon battery life for different images being approximated.

Figure 8 shows that the image quality drops when the beacon's power consumption is reduced to target a prolonged device lifetime. A longer lifetime limits the broadcasting frequency, and as a result, the amount of data we can transmit within a fixed period (1 second in our setting) is reduced– resulting in a poor quality encoding that uses less amount of bytes. This can however be fixed by allowing a longer image transmission delay (i.e. more than 1 second wait-time for the reader application). We also notice that the basic geometric shapes and alpha-numeric characters achieve very high structural similarity scores under all power settings. This happens since their sub images are very similar to the patches. However, the system does not perform its best when tested with arbitrary, complex binary images (specially the ones with a lot of dark regions).

D. Effect of Number of Beacon Devices

In this experiment, we investigate the impact of the number of beacons on image quality. Recall that, we have used the the Eddystone URL beacon packet format to store the custom image data where each beacon can contain at most 16 bytes of image data. Hence, the more beacons we use, the more bytes we have available to store the same image. Figure 10 shows that as the number of beacons is increased from 1 to 6, the quality of produced images also increases from 0.72 to 0.84.

E. Effect of Patch Function Type

In this experiment we compare three patch generation methods. The first one is the proposed k-means based ap-

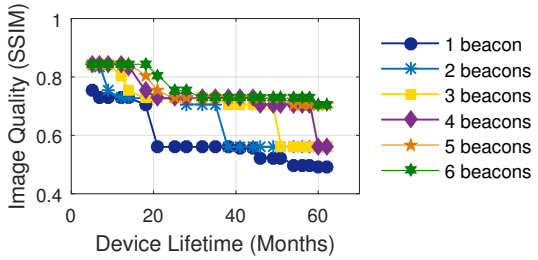

Fig. 9. Image quality versus device lifetime for various number of beacons.

proach that considers multiple spirals. The second one uses only a single spiral, and the third one generates patches from a standard test dataset for hand-written characters MNIST [5]. To generate patches, we up-scale the images to 64×64, adjust intensity, and dilate the background. Then we divide each image into patches and run k-means on the patches.

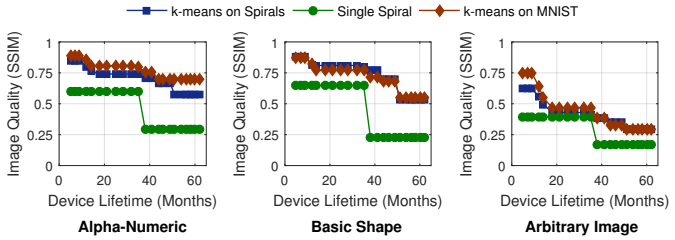

Fig. 10. Image quality versus lifetime for different patch generation methods.

From Figure 10 we observe that patches generated from MNIST dataset performs the best for alpha-numeric characters. This is expected since MNIST is designed specifically for handwritten digits. Our multiple spiral-based method performs better for basic shapes, and both algorithms perform similar when tested with arbitrary images. This shows that having prior knowledge helps, but even if we do not have have it, our method performs reasonably well.

F. Effect of Patch Set's Size

An important parameter of the beacon system is the size of the patch set, which is same as the number of clusters k in k-means clustering. A larger patch set is more capable in representing an input image, but requires more bits to encode the image. This leads to a higher broadcasting frequency (given the 1 second bound on the maximum transmission latency), and more power consumption. So, it is important to find a good value for k. Figure 11 shows image quality versus device lifetime for various k. The plot shows that $k = 64$ almost always outperforms others as expected. The other two values of k also perform reasonably well (0.65–0.7 when expected lifetime is between 20–40 months). Note that $k = 128$ is not applicable to our setup (i.e., 3 beacons and 1 second latency) since this would require an excessive amount of space to store an image.

G. Effect of Grid Size

Similar to the patch set's size, the grid size used to divide a spiral image to produce different sizes of patches also has impacts on the image quality and the encoded image's size. Using smaller patches results in high resolution image

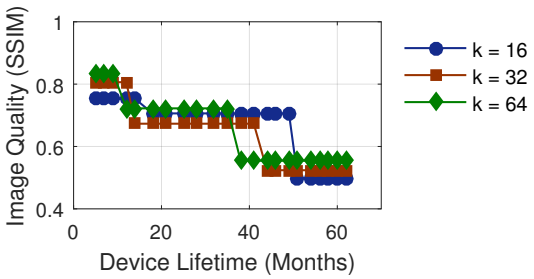

Fig. 11. Image quality versus device lifetime for various patch set sizes.

encoding, but the size of the encoded image will be larger. For example, a grid size of 1 pixel results in an exact replica of the original image. Figure 12 shows the system's performance for different grid sizes. Using 3 beacons and a grid size of 4 pixels, the system cannot encode images when the desired device lifetime life is more than 20 months, but produces best quality images for shorter expected lifetimes. With a grid size of 8 pixels, the system would last for about 50 months and will perfprm consistently well to produce images having about 0.75 structural similarity scores. Using larger grids (16 and 32 pixels) results in even longer device lifetime, but the quality of images degrades to 0.5.

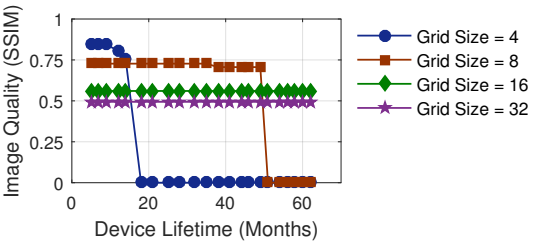

Fig. 12. Image quality versus device lifetime for various grid sizes per patch.

VI. Real Deployment

We deploy the image beacon system in two scenarios where a group of participants (1) use the smartphone application to draw, preview, write into, and read from a set of image beacons, and (2) use a beacon guided navigation system in a multi-floor indoor setting. All the participants are graduate students in our department.

A. Write-Read-Recognize

We ask 12 participants to use the 'previewer' app to draw images on the phone using its touchscreen. These images are converted to binary and saved into the phone's internal storage. Each participant draws random images containing digits, letters, numbers, and recognizable symbols and shapes. They also label their drawings with meaningful tags. These images are then encoded so that they can be stored in 4 beacons, their transmission delay is 1 second, and the lifetime of the system is over two years. A subset of these encoded images are then shown to other participants who are asked to recognize them and then provide subjective scores for them. The score denotes how a participant feels about the overall quality of the shown image and his difficulty in recognizing the symbols and objects in that image. The score ranges from

1 to 10, where ‘1’ stands for the lowest perceived quality and the hardest to recognize. Seven of these images are scored by all participants. These are shown in Figure 13.

Fig. 13. Images drawn by the participants.

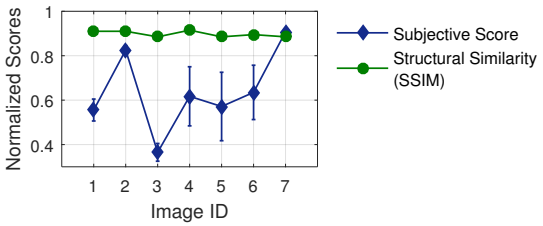

Fig. 14. Subjective and SSIM scores for each image.

We gather user scores and normalize them to [0, 1]. To compare these subjective scores with structural similarity (SSIM) based objective scores, we computed the mean of the user scores for each image and show them in the same plot with their SSIM scores. SSIM scores are also scaled to [0, 1]. Figure 14 shows that the trend in the SSIM curve and the averaged subjective quality scores are about the same except for the last image of ‘pi’. The averaged subjective scores for different images vary more than the SSIM scores. The reason for this is that the participants tend to give a relatively low score (around 1 to 3) for the images they fail to recognize. This has happened for the ‘@’ symbol. Many participants tend to recognize this as a compressed version of double ‘O’. The ‘pi’ image has a very high average subjective score but a relatively low SSIM score. This ‘outlier’ may be due to the fact that unlike other images, this image takes the whole image display area, and therefore, it is easier for human to recognize.

B. Navigation in the Building

In this experiment, we choose six locations inside the UNC Computer Science building, and at each location we place a beacon that broadcasts a compressed image. These compressed images serve as ‘guides’ for navigating inside the building. Each participant is asked to follow the navigation information displayed on his phone screen. The participant has been instructed that ‘walking forward’ is the default direction, and if he sees, for example, a ‘turn right’ image at a turning point, he should make a right turn and keep walking straight until the current image disappears and the next image is displayed. This experiment is designed to simulate a real world use case. In a place, where the Internet is not accessible, one can write arbitrary graphical information into a set of beacons for others to read at a later point. Therefore, it is critical to support arbitrary binary image writing functionality in our system, instead of having a predefined set of symbols.

The navigation path was designed so that it starts at a corner on the second floor, goes across half of the floor, continues to the third floor, goes across to the half of the third floor, and then stops. Figure 15 shows the path. Each beacon’s broadcasting interval is set to 1000 ms and the broadcasting

Fig. 15. Map of second and third floors showing the navigation path.

strength is set to -20 dBm so that it’s range does not overlap with its neighboring beacon’s. The experiment finishes when a participant successfully reaches his final destination (the last beacon showing the ‘STOP’ image) or the participant feels totally lost and cannot continue. After the experiment, the participants are asked a set of questions to answer: (1) overall difficulty of going from the starting point to the third floor, (2) overall difficulty of going from the starting point of the third floor to the destination, (3)–(8): how clear each image looked, (9) usability of the app (image reader). The score ranges from 1 to 10, where ‘1’ stands for the least perceived quality and the hardest to recognize.

Fig. 16. Images stored at the six locations.

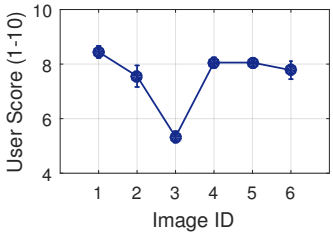

Fig. 17. Subjective scores for each image.

Of the 5 participants who completed the user study, 4 successfully research the destination point. 1 participant was lost after he reached the third floor. The images used in the study are shown in Figure 16. For each of these six images, we plot the average subjective scores in Figure 17. It is interesting to note that, although the second and the fifth image are the same, the fifth image has got a higher average score. We believe, this is because when a person sees the same direction for the second time, he tends to be more confident

in recognizing the image. The third image has got the lowest score. This is due to the fact that the instruction at that location asks the user to go upstairs which was confusing to many participants who did not expect a multi-floor travel path.

We also asked each user about his difficulty in following the navigation signs in floor 2 and floor 3. The average scores for these questions are 0.74 and 0.88, which means the users thought that the directions were easier to follow during the later stage of the experiment when they became used to the system. The overall satisfaction of the application was very high. All participants commented that the app is 'easy to use'.

VII. Related Work

Previously people have developed various image compression techniques. Many of these techniques have been standardized and widely used, including JPEG that is based on discrete cosine transform, and JPEG2000 that is based on wavelet transform. If the images are limited to binary, a compression method designed for binary images is expected to show a better performance (smaller compressed image but equivalent image quality) than JPEG or JPEG2000. One approach to compress binary images is to partition the shape in the image into rectangles and record the upper-left and bottom-right pixel location of every rectangle [7][14]. This works great on images that contain shapes that can naturally be divided into rectangles. But in our system, many images contain curves in different directions and different curvatures. For binary images containing curves, rectangle-based approach do not give a desired performance within the limited storage constraint. Given the fact that many of our binary images contains curves, chain coding based compression [4] would be preferred. [13] proposed another chain-coding based compression method. They showed that their coding method could losslessly compress a complex contour shape into around 900 bytes. And their result outperforms previous image coding techniques including JBIG1, JBIG2 and data compression library WinZip. Their method yields compressed image sizes of at least 200 bytes for our data. This still does not satisfy our storage requirement, and at the same time we do not need the very-high compressed image quality that their method offers. Therefore, we design a binary image compression method that better fits our requirement.

In [9], the author discussed an intelligent system involving beacon devices for Customer Behavior Analysis (CBA). The goal is to show how beacon technology could help gather and classify customer behavior data in retail stores. They deployed the system into a real retailing scenario and collected the data from mobile devices interacting with Beacon devices. They also proposed the further data analysis process on the collected data. [6] talks about an indoor localization system constructed with beacon devices. With multiple beacons places in the environment, the author showed that the error of the estimated location of the object could be as small as 0.53 meters in average. [2] discusses an Beacon occupancy detection system deployed in smart buildings. In the implementation, they modified Apple's iBeacon protocol to better fit their requirement. They showed that with the BLE technology such system could be more energy and cost efficient than previous solutions.

VIII. Conclusion

In this paper, we described a system involving beacon devices and smartphones with BLE receiving functionality. Our system allows a user to generate binary images, write binary images into the very limited beacon device storage, and receive compressed binary images from beacon devices' broadcasting packets. The main contribution of this work is the overall system construction, the patch-based binary image compression method, the evaluation of various system parameters in the system, and the evaluation of the trade-off between image quality and beacon battery lifetime based on our patch-based image compression method. Our work widens the usage of the energy efficient, long lifetime beacon devices by allowing easy storage and access of custom image data in scenarios where there is no Internet connection.

Our system is most useful when the application scenario requires years-long working time without maintenance. In the future, it is meaningful to design a long timescale experiment to analyze the data write-read pattern within a long time period.

References

[1] Estimote Beacons. https://estimote.com/.

[2] G. Conte, M. De Marchi, A. A. Nacci, V. Rana, and D. Sciuto. Bluesentinel: a first approach using ibeacon for an energy efficient occupancy detection system. In *1st ACM International Conference on Embedded Systems For Energy-Efficient Buildings (BuildSys)*, 2014.

[3] C. Gomez, J. Oller, and J. Paradells. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors*, 12(9):11734–11753, 2012.

[4] S.-D. Kim, J.-H. Lee, and J.-K. Kim. A new chain-coding algorithm for binary images using run-length codes. *Computer Vision, Graphics, and Image Processing*, 41(1):114–128, 1988.

[5] Y. LeCun, C. Cortes, and C. J. Burges. The mnist database of handwritten digits, 1998.

[6] P. Martin, B.-J. Ho, N. Grupen, S. Muñoz, and M. Srivastava. An ibeacon primer for indoor localization: demo abstract. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, pages 190–191. ACM, 2014.

[7] S. A. Mohamed and M. M. Fahmy. Binary image compression using efficient partitioning into rectangular regions. *Communications, IEEE Transactions on*, 43(5):1888–1893, 1995.

[8] N. Newman. Apple ibeacon technology briefing. *Journal of Direct, Data and Digital Marketing Practice*, 15(3):222–225, 2014.

[9] R. Pierdicca, D. Liciotti, M. Contigiani, E. Frontoni, A. Mancini, and P. Zingaretti. Low cost embedded system for increasing retail environment intelligence. In *Multimedia & Expo Workshops (ICMEW), 2015 IEEE International Conference on*, pages 1–6. IEEE, 2015.

[10] R. J. Schalkoff. *Digital image processing and computer vision*, volume 286. Wiley New York, 1989.

[11] M. Siekkinen, M. Hiienkari, J. K. Nurminen, and J. Nieminen. How low energy is bluetooth low energy? comparative measurements with zigbee/802.15. 4. In *Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE*, pages 232–237. IEEE, 2012.

[12] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612, 2004.

[13] S. Zahir, K. Dhou, and B. Prince George. A new chain coding based method for binary image compression and reconstruction. *PCS, Lisbon, Portugal*, pages 1321–1324, 2007.

[14] S. Zahir and M. Naqvi. A new rectangular partitioning based lossless binary image compression scheme. In *Electrical and Computer Engineering, 2005. Canadian Conference on*, pages 281–285. IEEE, 2005.