Demo Abstract: Image Storage and broadcast via Bluetooth Low Energy Beacons

Chong Shao
Department of Computer Science
Univeirsity of North Carolina at Chapel Hill
Chapel Hill, North Carolina 27599
cshao@cs.unc.edu

Shahriar Nirjon
Department of Computer Science
Univeirsity of North Carolina at Chapel Hill
Chapel Hill, North Carolina 27599
nirjon@cs.unc.edu

ABSTRACT

This demo is an implementation of the first 'image beacon' system that is capable of broadcasting color 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 an image processing pipeline that takes into account the background and foreground information of an image and then applies an adaptive encoding method which priorities more important regions of an image during encoding, in order to achieve the best quality image under a very strict size limit. We develop a Raspberry Pi application for image writing. It takes an image and user-requirements as inputs, shows previews of different quality output images, writes the encoded image into a set of beacons. We also develop a smartphone application that reads the broadcasted image from beacons. In practice, the system can broadcast image data for years without external power supply.

CCS CONCEPTS

•**Hardware →Signal processing systems;**

KEYWORDS

bluetooth beaconing, image processing

ACM Reference format:
Chong Shao and Shahriar Nirjon. 2017. Demo Abstract: Image Storage and broadcast via Bluetooth Low Energy Beacons. In *Proceedings of The 2nd ACM/IEEE International Conference on Internet-of-Things Design and Implementation, Pittsburgh, PA USA, April 2017 (IoTDI 2017),* 2 pages.
DOI: 10.475/123_4

1 PROBLEM DEFINITION

We propose a system for digital image storage and broadcast using Bluetooth Low Energy Beacons. The image broadcasting is achieved in a connectionless mode. The challenge in building such a system is that a BLE protocol and beacon is designed to broadcast a few bytes of data as BLE 4.0 advertisement packets at a rate of fewer than 30 bytes/sec. The bound in data rate comes from the lifetime requirement of these devices. Such a tight budget on payload size and the maximum data rate have limited a beaconfis capability only to be able to broadcast an identifier or a small amount of text. The next generation BLE 5.0 beacon is expected to have an 8X increase in broadcasting capacity (256 bytes). Such an increase opens up the possibility to design beacons that can serve larger assets such as an image. However, even a simple 72 x 72 PNG image, such as the Android launcher icon, has a size of over 3KB. To store and broadcast this image, either we require to use a dozen of BLE 5.0 beacons, or we will have to accept a very long image transmission and loading time.

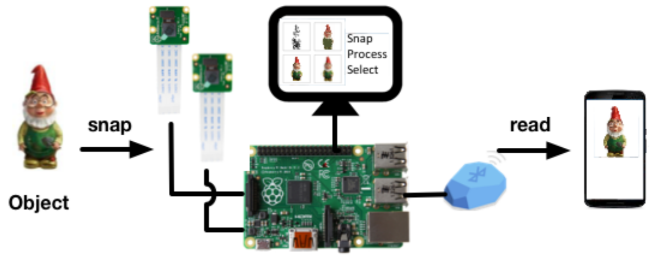

Figure 1: An image beacon system.

Image compression is a natural way to deal with this problem. Existing image compression algorithms [1, 3, 5], however, fail to achieve the desired compression ratio for an image to be broadcasted over BLE. Hence, a fundamental challenge toward realizing an image beacon is to devise an algorithm that efficiently represents an image using as few bits as possible, while taking into account the application-driven limits on the number of usable beacons per image, broadcast message size, data rate, latency, and lifetime. In an earlier work [4], we developed an image beacon system that broadcasts binary images of a few limited categories (e.g., handwritten characters) only. This demo is a continuation of that line of work, but this time, we have taken a harder challenge, i.e., to develop a beacon system that works for color images, e.g., images taken with a mobile phone.

2 APPROACH AND IMPLEMENTATION

The image compression pipeline compresses an image for a given set of user-defined beacon system requirements. We implement our the pipeline (Figure 2) on a Raspberry Pi 3 device. The pipeline consists of five basic stages: multiple view capture, depth estimation, depth-refined segmentation, image compression, and user selection.

(1) Multiple View Capture: The proposed system requires a user to capture two views of an object. This enables the depth map

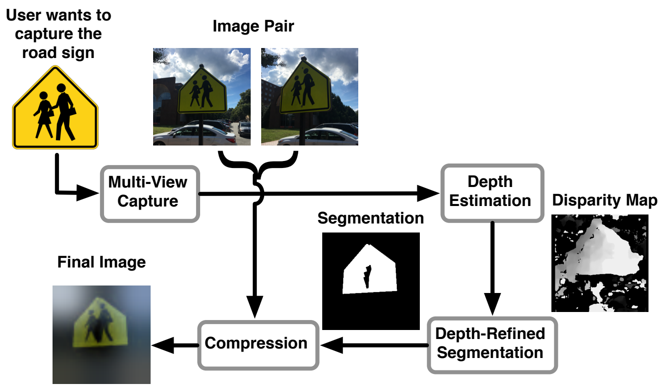

Figure 2: **Image processing stages.**

generation in a later stage. Here we used two Raspberry Pi board cameras connected to a Raspberry Pi. One Raspberry Pi board camera is equipped with an Omnivision 5647 sensor with 25921944 pixels. The two cameras are placed with fixed related position as the capture unit.

(2) Depth Estimation: Depth of each pixel is estimated by finding and matching corresponding fifeature pointsfi (e.g., corners and edges) in two or more images. The matched points are then used to generate the geometry relation between the cameras and the object so that the depth of every pixel can be estimated. However, due to the existence of noise, low resolution, and inaccuracies in estimation, depth map alone is not sufficient to segment semantic regions in an image.

(3) Depth-Refined Segmentation: Like depth map, color/texture based image segmentation algorithms often fail to identify semantically different/similar regions in an image. When we overlay one color/texture based image segmentation with the depth map to obtain a better segmentation. This step is inspired by one of our earlier work [2] that used RGB and depth images from Kinect sensors. In this work, we use only images to estimate the depth (previous step) and then apply this step to get the final segmentation.

(4) Image Compression: The image compression stage takes both an image (for texture and content information) and its segmentation map (for semantic region information), and produces the best quality image under the user-specific constraints of the beacon system. Until the resultant image size does not satisfy the system requirements, the algorithm gracefully degrades the quality of different semantic regions, starting from the least important one (e.g., the background).

(5) User Selection and Broadcasting: The image compression stage outputs three versions of the compressed image, each compressed by one of the three encoding methods: discrete Fourier transform, wavelet transform, and triangulation, respectively. The user views the three images via the user interface, as shown in Figure 3. The user then selects the preferred version of the image. The selected compressed image is then broadcasted via the Raspberry Pi's Bluetooth 4.0 USB module.

3 DEMO SCRIPT

1) Capture and write: In this demo, the user will place different foreground objects in front of the cameras connected to a Raspberry

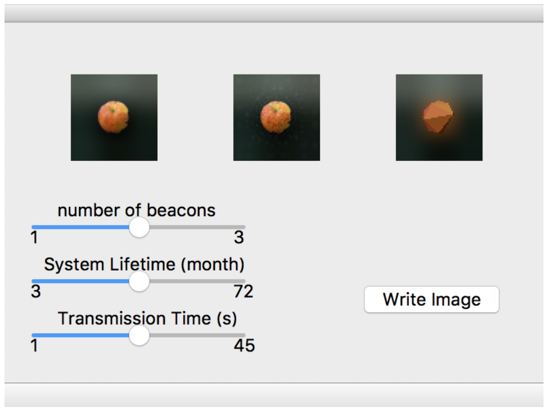

Figure 3: **Graphical user interface for parameter setting and compressed image selection.**

Pi. After the system captures the image, the user will be able to view the compressed image from a display connected to the Raspberry Pi. The user can select various image encoding strategies via the app's graphical user interface. The system will then show the user the image encoded by the selected encoding strategy. The user can confirm to write the selected image into a beacon.

2) Read: After the completion of writing, the user will use a smartphone running a customized app to read the image from the beacon. The user will be able to view the image on the smartphone.

4 RELATED WORK

Previously, a binary image beacon system [4] was developed to enable BLE beacons to store and broadcast binary images. The limitation of the system is that it only supports a limited category of binary images such as hand-written characters, shapes, and symbols.

Researchers have developed a variety of approaches to compress an image. One wavelet-based image encoding method was developed by Shapiro [5]. Said and Pearlman [3] improved the approach by designing a faster algorithm based on set partitioning in hierarchical trees. Lu et al. [1] introduced another image encoding method using surface triangularization. Both triangularization and wavelet-based image encoding are included in our adaptive encoding module, with changes in the method that focus on enabling smaller compressed image size.

REFERENCES

[1] Tianyu Lu, Zisheng Le, and DYY Yun. 2000. Piecewise linear image coding using surface triangulation and geometric compression. In *Data Compression Conference, 2000. Proceedings. DCC 2000.* IEEE, 410–419.

[2] Shahriar Nirjon and John A Stankovic. 2012. Kinsight: Localizing and tracking household objects using depth-camera sensors. In *2012 IEEE 8th International Conference on Distributed Computing in Sensor Systems.* IEEE, 67–74.

[3] Amir Said and William A Pearlman. 1996. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on circuits and systems for video technology* 6, 3 (1996), 243–250.

[4] Chong Shao, Shahriar Nirjon, and Jan-Michael Frahm. 2016. Years-Long Binary Image Broadcast using Bluetooth Low Energy Beacons. In *Proceedings of the International Conference on Distributed Computing in Sensor Systems (DCOSS 2016).*

[5] Jerome M Shapiro. 1993. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on signal processing* 41, 12 (1993), 3445–3462.