1 Introduction

Much attention in computer science and engineering has recently been devoted to the development of low cost, highly portable, distributed sensor networks. In particular, many have focused on distributed location discovery algorithms that would be instrumental in data collection and geographic routing algorithms such as proposed in [1]. However, problems at the physical level have prevented current implementations from widespread acceptance. Radio signal strength approaches have been found to be unpredictable in ad-hoc environments, while ultrasound is limited by the laws of physics to be bulky and power consuming. Ultrawideband requires expensive base stations, and RFID only provides proximity information.

The aforementioned technologies typically implement very general, robust communication links which allow a high level of connectivity within the sensor network. In many cases, however, this may not be necessary. In sensor networks in particular, much of the communication occurs from a low-level sensor node and a higher-level query node. Bidirectional communication is often required in order to establish routing paths due to the limited range of the link technology. However, the flow of data remains largely unidirectional in a typical network. If the need for routing can be eliminated, then a many-to-one, or broadcast, communications channel can be implemented.

Consider a scenario where experimenters are concerned with the saline concentration within a water tank. In order to provide a picture of the concentration gradient, a large number of sensor nodes are deployed throughout the water tank. For each node, the experimenters need to know both its location and the sensor reading. Rather than forcing the low-power nodes use their resources to determine relative locations, it would be ideal to have the receivers calculate the positions based on the properties of the transmission. This can be achieved by a visual communications channel, wherein data can be transmitted over sequential frames, and the position of the transmission can yield location information. Similarly, visual communication could provide an alternative implementation to RFID, wherein tags would emit a sequence of flashing light to be received by cameras. This would allow the cameras to determine the location of some item which can be found through a database lookup based on the address.

In this project, a broadcast channel is developed using optical transmission from simple LEDs to a cheap webcam. A prototype two-tiered system is proposed to enable optical broadcast. The first tier consists of sensor nodes equipped with only a few basic sensors and an LED.
Sensor data, along with node address information, can be transmitted by a blink pattern. The second tier consists of higher-power computers attached to the webcams. This work provides the algorithms for optical communication and also assesses the viability of their application toward motion tracking using the localization algorithms provided by Savvides, et al. [2] Concurrently, Mascia [3] has evaluated the possibilities in engineering low-cost solutions which implement optical broadcatch in hardware. In addition, Mascia demonstrates that the transmitter nodes can be implemented at extremely low cost with minimal power consumption.

This paper is organized as follows. The algorithms for data transmission via optical broadcatch are outlined, followed by an analysis of the algorithms. Physical specifications for the prototype system are provided, and results from localization based on optical broadcatch are shown. From these, results an analysis is made of the modifications required in order to perform motion tracking with this communications paradigm. Finally, possible directions for further research are suggested.

2 Algorithm

2.1 Data Transmission

A binary stream is represented by a blink pattern, in which a 1 corresponds to the LED being on, and a 0 to the LED being off. The bitrate of the communication channel is dictated by the Nyquist sampling rate [4], where the limiting factor is the frame rate of the camera. Due to the way cameras capture an image, two samples must be taken for every bit. The camera does not take an instantaneous sample; rather, the lens is exposed for some finite amount of time, and the captured image contains light throughout the exposure.

Figure 1: A mismatch between the transmit frequency and frame rate causes an incorrect number of bits to be read. Here, only fourteen frames are taken for the eight bits transmitted, causing a bit error in the sixth bit read and the last bit to be dropped.

If one sample is taken per bit, any phase shift between the sampling rate and transmission rate will cause the camera’s exposure to capture light outside of the period of each bit. In particular, it is very common for bit transitions to be read as two high bits, as the high bit blends into the frame that is supposed to contain the low bit. Figure 1 provides an example of this problem.

By applying Nyquist’s sampling theorem, at least one frame is guaranteed to fall wholly within the transmission period of a bit. A simple solution to reading bits is to sample every other frame after synchronizing on a dark frame. This synchronization guarantees that a correct bit is being read since a completely dark frame indicates that no transition occurs. Unfortunately, this requires very tight frequency matching between the transmitter and the camera, or else a bitshift may occur causing the camera to read the incorrect frame. By reading every frame, the receiver can detect and recover from a bitshift caused by a frequency mismatch. A one is read when two consecutive high frames are seen, and a zero is read when the first low frame is seen. The finite state machine represented in Figure 2 accounts for the possibility of both frequency shift and phase shift, and thus allows for robust data transmission.
2.2 LED Recognition

In order for the algorithm to operate in real-time based on the camera’s frame rate, the algorithm attempts to avoid the problems typically associated with computer vision. Techniques for high-level object detection require extensive matrix operations as operators are convolved with the image. Logical-linear operators have been used to distinguish objects from the background, but it is an iterative method that may require a number of iterations for complex scenes [5]. Even implementation of Canny edge detection, the current standard in computational vision, is insufficient for our real-time requirements. Neoh and Hazanchuk [6] evaluated the viability of real-time high resolution edge detection. They estimated that the Canny edge detector requires 130 operations per pixel. A 600MHz machine could process about 4.6Mpixels per second. Even at our resolution of 352 × 288 (CIF), Canny edge detection would begin to overload the processor, without even accounting for capture time and memory operations. Neoh and Hazanchuk [6] achieved their goal only by optimized design of a hardware DSP.

To keep the edge detection algorithm as simple as possible, some basic assumptions are made concerning the behavior of the image. Each frame is differenced with a picture of the background, so that the algorithm receives a predominantly dark scene sparsely populated with the LEDs within the camera’s field of view. This scene is then thresholded against a user-defined level, which can remove any false positives that may result from noise in the camera, or objects entering or leaving the scene. With the frame preprocessed to reduce the number of objects in the frame to essentially only the flashing LEDs, the algorithm is ready to perform object detection.

The LEDs are assumed to be well-behaved, in that they are expected to be continuous blocks of bright pixels. An LED is repre-
presented as a rectangle bounded by four coordinates. Object detection, then, is performed as follows. When the algorithm finds a pixel higher than the threshold value in the image array, it creates an LED object, with the initial representation as a $1 \times 1$ rectangle. It then attempts to expand the node in all directions by checking the neighboring points for above-threshold pixel values. As soon as a pixel above the threshold is found, its value is blanked by setting it below the threshold in order to prevent duplicate detection. In this fashion the algorithm can determine the maximal bounding rectangles for any LEDs in the image with this processing scheme. The algorithm is described in Algorithm 1.

### 2.3 Optical Broadcast

With a model of object detection and a robust method for optical data transmission, it is now possible to read bitstreams flashed by the low-level LED nodes. First, we define a communications protocol which allows for two different packet types — node identification packets and data packets. An 8-bit preamble of alternating ones and zeroes starts every packet, which allows the receiver to eliminate any false positives that may arise from noise sources. The preamble is followed by a one-bit packet type, and then a variable length node address. This is by default set at four bits, but can be user controlled through the capture interface. If the type bit was set for the data packet, then the receiver will read a three-bit payload length, and then a maximum of eight bytes of data. The packet format is represented in Figure 3.

To implement optical broadcast, a global list of previously detected objects is maintained. At the start of each frame processing cycle, the implementation of the data transmission finite state machine reads a bit value from the pixels within the boundaries of each previously detected object. All pixels within

---

**Algorithm 1 Object Detection**

```plaintext
procedure OBJECT DETECT
    for pixels i in image do
        if image[i] > thresh then
            ExpandNode(x_i, x_i, y_i, y_i)
        end if
    end for
end procedure

procedure EXPANDNODE(left, right, bot, top)
    nleft ← left, nright ← right
    nbot ← bot, ntop ← top
    for i ← left to right do
        if image[i, bot − 1] > thresh then
            nbot ← bot − 1
        end if
        if image[i, top + 1] > thresh then
            ntop ← top + 1
        end if
    end for
    for j ← bottom to top do
        if image[left − 1, j] > thresh then
            nleft ← left − 1
        end if
        if image[right + 1, j] > thresh then
            nright ← right + 1
        end if
    end for
    if boundaries were changed then
        ExpandNode(nleft, nright, nbot, ntop)
    end if
end procedure
```
these objects are blacked out after processing. The algorithm then performs object detection as outlined in Algorithm 1, adding new objects to the global list. Once a packet transmission is complete, the LED object is deleted from the list. In addition, an object is pruned from the list as soon as it fails to match the preamble.

In the current implementation, completed packets are written to a TCP server. Interested clients can open a socket connection which will give all the data captured by the receiver. Additionally, the variable-length address allows the client to extend the functionality of the packet, such as some error checking. Currently, a seven-bit ”address” is received by the camera, four of which are actual address bits, with the other three transmitted as a Hamming code. The postprocessing is left to the client.

3 System Implementation

An initial system was developed using Logitech Quickcam Communicate webcams and ENALAB XYZ nodes modified with an omnidirectional LED. The webcams are connected to Windows PCs which run the camera capture application. The application provides an implementation of the broadcast algorithm and uses ActiveX controls provided by VideoOCX [7] to capture frames from the attached webcams. The cameras recorded CIF images at the maximum frame rate of 15 fps. It would be possible to capture at 30 fps at lower resolutions, but we found it difficult to consistently achieve this frame rate, particularly with two cameras attached to each computer.

The XYZ nodes were outfitted with Lumex CCL-CRS10SR surface mount LEDs, which provide a 180° viewing angle. The nodes run a simple application that periodically blinks the node address within the specified packet format.

As a test of the optical broadcast algorithm, localization was performed using the output from the application. The output was modified to give the center image coordinates of each LED detected, along with the node address. A Matlab script implemented the localization algorithm described by Savvides, et al [2], which extends the SVD algorithms proposed by Tomasi and Kanade [8]. The script processed the addresses output by the application with a Hamming code algorithm for error detection and correction.

4 Evaluation

4.1 Algorithm Analysis

Since the aim of the broadcast algorithm is to provide a unidirectional, many-to-one communications channel, it is very important for it to be scalable to a large number of nodes without incurring any performance penalty that would prevent real-time processing of the frames. Because of the real-time requirements, both the order of growth and the bounding constant are pertinent to this evaluation. For this analysis, it is assumed that there are \( N \) LED nodes in the system, with each covering a \( 1 \times 1 \) pixel area in the image. In addition,
there must be at least a single pixel between LED node objects to prevent recognition as a contiguous object. To make the analysis more clear, two cases will be considered: one in which all nodes have already been recognized, and the algorithm collects packets from the bitsreams, and one in which the object detection algorithm must find all of the nodes.

Assume that all objects have already been detected. In this situation, the algorithm has to check every LED object to read the transmitted bit, and then search the entire image for new objects. Reading each bit requires a constant number of operations, basically involving comparison to the threshold and updating a few state variables. In particular, it updates the state of the bitstream based on the finite state machine in Figure 2. The LED object also has an internal representation of the packet, so it must shift any bits read in to the current value, in addition to maintaining which segment of the packet is currently being transmitted.

All of this should require no more than ten operations. In the event that packet transmission is complete, all the data is copied to buffers maintained by the TCP server, so this only involves memory operations based on the length of the packet. The current packet specification allows for at most twelve bytes, so that each node should require no more than 22 operations during data transmission.

The object detection algorithm will proceed to look for new objects, but, since all of the nodes are blacked out after processing, it will find none. Object detection simply requires comparison of each pixel in the image to the threshold value. The number of pixels can be related to the maximum number of nodes that can be contained within image. Assuming maximum density of the nodes, the image looks like a checkerboard, so that \( P = 2N \), where \( P \) is the number of pixels.

Therefore, data transmission requires at most \( 2N + 22N \) operations, so the order of growth is \( \Theta(N) \).

Analysis of object detection is a little more complex, as a pixel value may be checked multiple times due to the node expansion algorithm. Fortunately, this is also well-bounded – in the worst case, each pixel is evaluated five times. The loop in the Object Detection procedure described in Algorithm 1 checks the pixel value once against the threshold, and then proceeds to expand the node based on the four immediate neighbors. Object detection can be performed in linear time \( \Theta(N) = 5N + N \), where the extra \( N \) term comes from the half of the pixels which do not represent LEDs, and thus are only compared to the threshold value without node expansion.

The entire algorithm is linear with respect to the number of nodes, which allows broadcast to scale to very large systems. In practice, there are boundary conditions and memory allocations that will increase the constant factor, which will affect the algorithm’s real-time processing. However, it would require over one thousand operations per pixel to overload a 2GHz machine processing CIF images at 15 fps. Since there are no I/O operations, there should be very minimal time spent outside the basic operations of the algorithm.

### 4.2 Physical Characterization

Since a large part of this work involves establishing a new communications channel, it was necessary to determine the physical conditions under which the channel could reliably operate. The first parameter evaluated was the range at which the cameras could detect the transmitter nodes. The stock LED mounted on the XYZ node could only be detected up to two meters away from the camera. Using the much brighter, omnidirectional Lumex LED extended the range to five meters. Table 1 shows the typical software rep-
Table 1: Pixel area represented by an LED at varying distances.

<table>
<thead>
<tr>
<th>Distance (m)</th>
<th>T= 50</th>
<th>T= 75</th>
<th>T= 100</th>
<th>T= 125</th>
<th>T= 150</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>9 x 7</td>
<td>6 x 6</td>
<td>5 x 6</td>
<td>4 x 6</td>
<td>3 x 6</td>
</tr>
<tr>
<td>4</td>
<td>3 x 5</td>
<td>2 x 4</td>
<td>2 x 3</td>
<td>2 x 2</td>
<td>2 x 2</td>
</tr>
<tr>
<td>5</td>
<td>4 x 5</td>
<td>3 x 4</td>
<td>3 x 3</td>
<td>3 x 3</td>
<td>1 x 1</td>
</tr>
<tr>
<td>6</td>
<td>2 x 1</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

representation of an LED at varying ranges.

The camera is also limited by its field of vision, which was relatively small for the Logitech webcam. The viewing angle was 32° in the horizontal plane and 28° in the vertical plane. Assuming an application where a camera is hung from a ceiling four meters high, the camera could read LEDs in a 2m x 2.3m area.

Finally, it is necessary to characterize the resolvability of distinct nodes with the Logitech camera. The spacing between two nodes was varied until the camera could no longer consistently recognize them as two distinct LEDs. At a distance of 2.4m, the camera could resolve nodes separated by 2.3cm. Moving the nodes 1.86m from the camera allowed a 2.0cm separation.

4.3 Localization Results

A brief summary of the localization results obtained by Savvides, et al [2] is included here as a possible application of optical broadcatch. In these tests, the program captured approximately 90 percent of the packets transmitted under ideal lighting conditions. In a room lit with overhead fluorescent lights, a much more typical setting, Hamming code error detection and correction was necessary in order to achieve the same consistency. Figure 4 gives a plot of the localization accuracy at various resolutions.

4.4 Motion Tracking

The aim of this project was to determine what would be required to perform cheap and efficient motion tracking. Having established the localization algorithm, it would be possible to extend optical broadcatch to motion tracking. Currently, a node cannot be localized until multiple cameras recognize it, along with other nodes common to their field of view. Since a node identification packet requires thirteen bits, and optionally an additional three bits for the Hamming code, a node can only be localized about once every two seconds, at best. Since a node must remain stationary while the camera receives its data, or at least conform to some definition.

Figure 4: Localization accuracy with respect to camera resolution. Figure courtesy [2]
of continuity, this poses severe limits on the velocity of the node.

Based on the physical parameters of the camera system and the transmission speed of the broadcatch protocol, a set of general equations can be defined that govern the maximum velocity a node can travel while still being tracked by the camera. Let continuity for motion tracking be defined such that an LED’s representation in an image must overlap with at least one pixel within the representation of the same node in the previous frame. Furthermore, let \( n \) be the smaller dimension of the node’s pixel area, let \( \theta \) be the viewing angle along the dimension of \( n \), and let \( d \) be the normalized distance from the node to the camera, as measured between two parallel planes, both of which are parallel to the image plane of the camera. In addition, \( N \) is the number of pixels captured by the image in the same dimension as \( n \), and \( T \) is the time required for localization. The following equations define \( dx \), the distance represented by each pixel, and yield the maximum velocity, \( v_{\text{max}} \) that can be tracked.

\[
dx = \frac{2d \tan \theta}{N} \quad (1)
\]
\[
v_{\text{max}} = \frac{(n-1)dx}{T} \quad (2)
\]

If both dimensions of the node representation are equal, then the values of \( \theta \) and \( N \) should be chosen to minimize Eqn. (1). Alternatively, the component velocities could be calculated for each dimension using these equations.

As a typical example, assume that the node is four meters away from the camera. From Table 1, the node will be represented by a 3 x 3 pixel area. Using the standard CIF resolution, Eqns. (1) and (2) give a maximum velocity of 6.9 mm/s. The Ragobot project at UCLA [9] adds mobility to portable computing devices using a tank traction platform with a maximum velocity of 8.57 cm/s [10]. The current algorithm is clearly unsuitable for motion tracking for these devices.

From the equations, it is evident that either the distance represented by each pixel or the time required for localization must be changed in order to increase the maximum trackable velocity. \( \theta \) is a parameter intrinsic to the camera, so it must be accepted as is. \( N \) is user selectable, but decreasing it also increases the error in localization, essentially negating the benefits of increased distance per pixel. \( d \) can be controlled, but for the sake of robustness should be as wide a range as possible, extending to the maximum distance possible for node recognition. This leaves \( T \) as the only variable for optimization.

A couple of approaches can be taken to reduce the time required to localize a node. Increasing the frame rate to 30 fps would allow the time to be cut in half. This could be done without incurring too large a penalty on image resolution, as the camera supports 30 fps at 320 x 240, which is fairly comparable to CIF resolution. This solution is easy to implement, but would require more computing resources, since it would be difficult to attain this kind of performance with two cameras attached to one computer, as mentioned previously.

Algorithmically, it would be possible to reduce the time required for localization by tracking an LED that is constantly on. Currently, nodes transmit identification packets which are immediately discarded by the receiver. However, the packets can be made persistent, so that the receiver associates a node with a known pixel area. In the next frame, the receiver checks to see if there is an LED that satisfies the continuity requirements for the node. This can be done using a similar algorithm as before. Rather than reading a bitstream, the receiver checks each pixel within the node boundaries. As soon as it finds a value over the threshold, it resets its boundaries to the 1 x 1 pixel area represented by the high value, and then per-
forms the usual node expansion algorithm. This modification does not change the order of growth of the optical broadcast algorithm. It only increases the frequency with which the node expansion procedure is called, which adds at most $4P$ operations, where $P$ is again the number of pixels in the image frame.

The protocol can be modified such that the node periodically blinks an identification packet to assure the receiver that it is following the correct node. During this transmission, it remains stationary. For the rest of the time, the node’s LED is constantly turned on and it is free to move about at maximum velocity. Some special cases would need to be implemented to handle scenarios when two nodes cross paths. This could be taken care of by some form of communication between the nodes as they approach each other, so that they blink identification packets immediately before and after the intersection. Better yet, the communication could be handled by the localizer when it detects that two nodes are approaching the same pixel coordinates in an image.

By updating the node locations at each frame, this algorithm could theoretically track objects moving at 19.5 cm/s under the current system setup, well exceeding the velocity of a Ragobot.

5 Conclusions and Future Work

This project established a method of communication using a simple LED and a cheap webcam. Its application toward localization demonstrates the reliability of the object detection algorithm, as well as its potential for use in motion tracking. As shown in the preceding section, the extension to motion tracking does not require much on the part of the algorithm. Future work in this direction could aim to determine the maximum time between transmissions of identification packets while maintaining accurate tracking. Furthermore, there is room for exploration regarding models of continuity. It may be possible that the requirements outlined here are unnecessarily strict, so that in sparse sensor networks only a proximity requirement is necessary. In addition, it would be useful to move the cameras from PCs to low-power devices, as envisioned in [2] and [3].

One aspect left largely uncovered here is the use of optical broadcast for massive data transfer. Although optical transmission is over three orders of magnitude slower than typical radio-link technologies, the target applications typically do not have stringent bandwidth requirements. In UC Berkeley’s sensor network at Great Duck Island [11], readings are transmitted once every 70 seconds. The low datarate of camera-based communication is irrelevant in these situations.

However, optical broadcast has a distinct advantage over competitors: it is incredibly scalable. In our previous work [12], we envisioned a camera simultaneously receiving data from a large number of nodes. Based on the physical parameters of the system, approximately 2000 nodes can be recognized in a single frame. The bandwidth of optical broadcast increases linearly with the number of nodes in the network, unlike other communications channels which suffer from increased usage. As a result, optical broadcast can approach the total bandwidth of radio-link technologies while achieving a substantially higher efficiency due to the rarity of transmission collisions. Work toward this end would include optimizing the packet format for data transmission. In particular, it would be useful to maximize the length of the data payload while maintaining reliability. A CRC checksum could be appended to the packet for additional robustness. Another
idea would be to add an Address Resolution Protocol-like functionality. In computer networks, ARP [13] is used to translate physical MAC addresses to higher-level IP addresses. An analogous model can be implemented for the optical broadcast protocol, where pixel coordinates can be associated with node addresses. Assuming static nodes, this pixel-to-address association can be used along with the Hamming code error correction to reduce the number of bad packets.

This project has provided a basic communications channel by which cheap, low-power broadcatching can allow for robust, scalable data transfer. The current implementation could easily be adapted into a linkable library, where the user would specify the output function as well as some parameters for the image processing. From this framework, it is possible to develop a wide range of applications, from the localization and motion tracking which formed the focus of this work, to the possibilities for massively parallel data transfer. This sort of communication has been largely ignored in the literature, but may gain increasing popularity as the applications of sensor networks find their needs accommodated by such a data link.

References


