An Ack Based Visible-Light Data Transmission Protocol

Avanish Kushal
Prasang Upadhyaya

ABSTRACT

We describe a communication protocol that uses visible-light to provide reliable and in order transfer of data between a pair of machines. Visible-light is a wireless transmission medium that also provides natural protection against eavesdropping since visible light rapidly attenuates with distance and angular variations. We experimentally verify these properties. We also design a flow control algorithm, similar to TCP, for dealing with dynamic ambient brightness, which has a strong impact on the performance of the system.

1. INTRODUCTION

The work aims to use visible light as a means of communicating small amounts of data in a client-server model. Devices that transmit and receive light are ubiquitous and attenuation of light with distance and under perspective distortions gives a naturally secure channel for wireless communication unlike traditional wireless.

We consider a simple hardware setup that uses standard desktop monitors as transmitters and stock web cameras as receivers of visible light. An example setup we used is shown in Figure 1. Unlike other visible-light communication protocols that use custom hardware or that use high mega pixel cameras, we restrict ourselves to consumer hardware with cameras with resolutions in the order of 2MP which is similar to mobile phone camera resolutions. Thus our system can also be used for mobile applications.

We assume the presence of a back channel from the client to the server for sending acknowledgements. In most phones today the camera is present at the opposite side of the display and so if we replace the client with a mobile phone this assumption would mean the receiver and transmitter at the server will have to be on opposite sides of the clients phone. In a client server model where the server is fixed and can take its input from any camera this problem can be overcome with apt placement of the webcam. Also new generation phones are coming with cameras on both sides. We also assume that the data is transferred with the devices being spatially stable - thus the devices do not need to register multiple times. This assumption naturally helps to improve throughput of the channel.

Figure 1: An Example Setup

A natural use case would be a situation where we would like to transfer some data onto the mobile phones of certain "qualified" users and so broadcasting the information over a wireless doesn't serve our purpose. For example, consider a book shop that wants to reward its premium customers with snippets from upcoming books - this information could be transferred by using this model without worrying about the different data transfer cables for the different makes of phones and do this in a secure manner.

1.1 Related Work

Recently there has been some interest in using the ubiquitous light sources to communicate data using wireless. To this end [9] build an LED based wireless communication system and study the attenuation of visible light. [8] present a fundamental framework for transmitting white LED light. These approaches use temporal techniques to transfer data.

In another line of work, using 2D Bar codes like QR codes with forward error correction, and motivated by the ISO standards developed [4, 6, 5] to utilize bar codes to code visual information, [10] setup a communication channel to study the properties of individual color channels and visible light in general. Our work is similar to this except we provide a reliable data communication channel using low resolution web cameras as opposed to 10MP SLR cameras used in their setup. Also they capture a

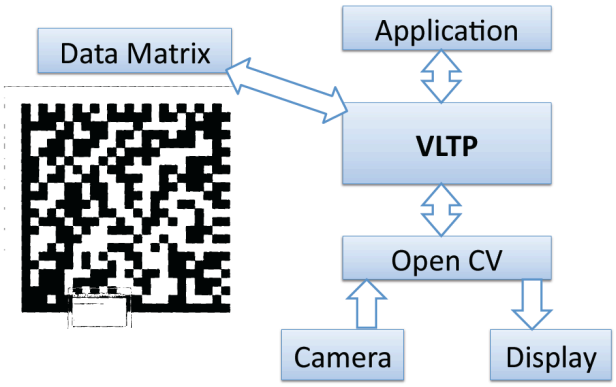

Figure 2: Overall Architecture

single frame and work with that - we setup a back channel with acknowledgements to send multiple frames in order.

Based on the comparative study [7] of prevalent 2D bar codes like QR code, VS code, Data Matrix, Visual Code and Shot code we use Data Matrix codes which are 2 dimensional barcodes and can store up to 2335 alphanumeric characters. Data matrix codes are used in various applications including encoding product and serial number and to identify circuits, lenses etc. Along with providing forward error correction, they ensure the brightness of the screen is roughly the same independent of the code, without which the web cam would frequently self-adjust brightness depending on the screen brightness and this dynamic adjustment causes more errors during frame capturing.

2. PROTOCOL

Our protocol provides functionalities of the link layer and the transport layer as defined by the OSI architecture. We do not provide any network layer functionality since we only experiment with a pair of machines and thus have no need for addressing features. But this is not a design decision that affects our protocol's correctness and our protocol can be easily augmented with network layer features.

2.1 Link Layer

The link layer has two main responsibilities in our design:

1. To determine the area of interest in a webcam's field of vision. We achieve this goal using *registration*.
2. To efficiently determine when a new packet is available to process.

The server transmits data only after the initial registration (step 1) completes. During data transmission, the server breaks the file into chunks (of sizes determined by flow control algorithms) and displays the Data Matrix encoding of the chunk. The client regularly captures images and applies the perspective correction defined by the registration on each captured frame and then tries to decode it. On success, the client sends an acknowledgment using its display monitor. On the server side, this acknowledgement causes the server to send its next frame. Thus, we use a window size of 1 in this protocol.

We now describe these steps.

2.1.1 Registration

In order to obtain a robust mechanism that works under different perspective transformations, we perform a registration between the web cameras and the displays to determine where the transmitted data pattern (2D barcode) is located within the captured image. We describe now how the server display is registered with the client. The other direction is similar to the server to client case.

All the steps we now describe are summarized in Figure 3

We first display a black screen at the server and capture that at the client. Next the server displays a white image and the client captures that as well.

Then we compute the difference image between the black and the white frames to detect where the display screen is within the image. However because of noise and color seepage due to brightness modifications it need not be a quadrilateral and cannot be used as is to determine the area of interest.

So, as a next step we compute the gradient image of the difference image using a Sobel filter to get the boundary of the quadrilateral. Note that this process is still noisy with many different lines forming some quadrilateral.

In the next step we use the Random Sample Consensus (RANSAC) algorithm [2] to detect lines in the gradient image. RANSAC randomly samples pixels and fits the best line through these pixels using least squares. If the line is indeed valid in the scene then many other pixels would be expected to lie on this line within some error bound. On the other hand if the randomly chosen points do not determine a line in the scene then very few points would be close to it in the gradient image. We choose the top 4 lines which give us the 4 most appropriate lines for the quadrilateral for the boundary of the display screen in the image (both in terms of least square error and number of points actually on the line). Note RANSAC is in particular robust to noise which is omnipresent in the image acquisition process.

Once we have determined the 4 boundary lines we find the intersections and determine the corner points of the quadrilateral. This now gives us a mapping between pixels on the display screen and their location on the acquired image. This mapping between 2 planes is called a homography. Once we have the homography, the area on interest is restored using well known techniques [3] for recovering perspective projections. We do this step both for the server and client to establish a back link for sending acknowledgment as well.

For the example we have shown, the four lines we detect and the corresponding corners are shown in Figure 3.

2.1.2 Efficient Identification Of New Frames

The receiver webcam regularly capturing images. For each such capture we need to determine if the frame captured has a new barcode. The naive approach would be to encode a sequence number in the data before generating the barcode. However, this approach requires us to to decode each barcode we capture to just determine whether the barcode was old or new. As we experimentally observed the bottleneck in this protocol is this decoding process at the client and thus to do this every time at the client for each captured frame would not be very efficient.

Thus, we choose to encode the sequence number in the color of the barcode (note that our window size is 1 and hence we need only 1 bit to represent whether the frame is odd or even). We alternate colors from black-white to red-white between consecutive frames as shown in Figure 4.

This helps the client to quickly determine if the frame is an old frame or a new frame without needing to decode it. The client determines the color by sampling pixels in the image to

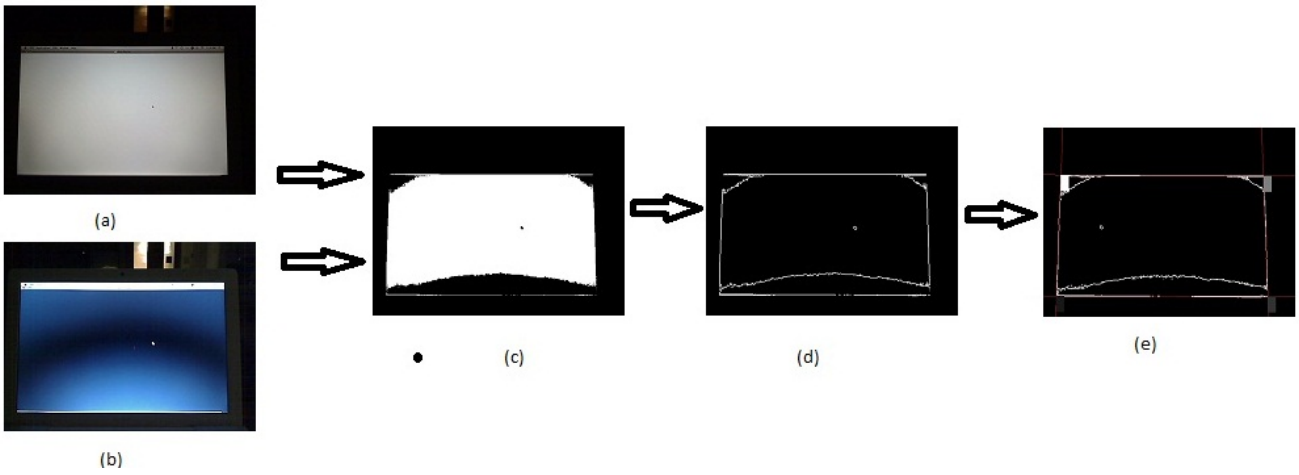

Figure 3: The figure shows the steps of the registration process in order. (a) shows the captured image at the web camera corresponding to a white screen on the server. (b) shows the captured image at the web camera corresponding to a black screen on the server. (c) shows the result of taking the difference of these 2 images(a) and (b) and thresholding the pixel-wise difference. (d) shows the result of applying the Sobel filter to (c) detecting gradient(boundary) for the varying area. (e) shows the result of applying RANSAC to (d) to detect lines and corners to determine the homography. Detected lines our shown in red and corners are shown as 4 small rectangles.

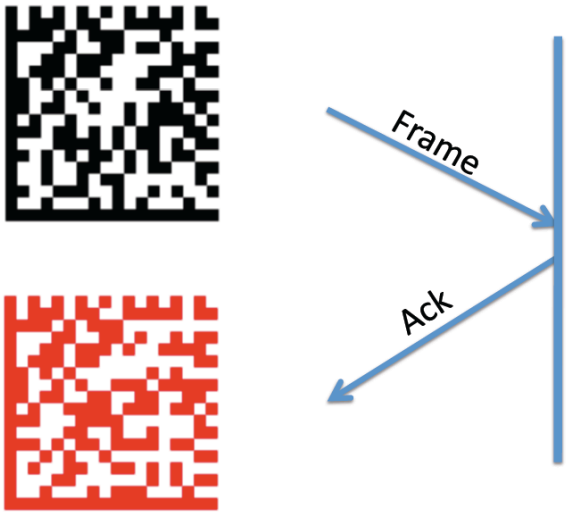

Figure 4: Acknowledgement Timeline. The server is on the left and the client on the right. The client switches its screen display from all white to all black to send an acknowlegement.

determine the percentage of red pixels. We set a low threshold (enough to take care of random noise) to classify the frame as an odd or even frame.

Thus the color of the frames acts as our sequence number - and we determine whether we need to decode the image or just drop it and move to the next frame.

Figure 4 summarizes the behavior of the link layer.

2.2 Transport Layer

In contrast with TCP where congestion control in the network is implemented by restricting the packet sizes, in our setup flow control is implemented by controlling the packet size as the we always have only one outstanding packet. Figures 8 shows the percentage of frames successfully received and decided as a function of the packet size, and Figure 9 the expected throughput as a product of the packet size and the acceptance rate. We can see that as we increase the packet size the success rate remains pretty steady before going through a very sharp falloff. Looking at the graph on the right, this is seen as there is a linear increase in the expected throughput in the region where the success rate is high. However once it crosses the falloff point the throughput falls down rapidly.

As we describe below and show in the experiments given the specific characteristic of the visible light channel we can do better than Additive Increase Multiplicative Decrease (AIMD) with Slow Start as done in TCP.

The primary difference between our system and TCP is that there are no other competing sources of traffic and each channel is dedicated to one server client interaction. Also, once the position and angle are relatively stable, barring changes in lighting conditions the optimal packet size does not vary much. Contrast this with normal networks where the target packet number is a much more dynamic quantity. Thus one can justify using a more aggressive strategy than arithmetic increase to reach the optimal packet size, or not reducing packet size by as much as half when a timeout occurs. We build on this intuition to propose the following algorithms for flow control:

2.2.1 Aggressive

Upon failure, The algorithm reduces the packet size to the last successfully transmitted size. If the packet was successfully decoded by the receiver before timing out the packet is incremented by a value δ and the value of δ is doubled for the next round. δ is reset to one after every timeout. If after a failure the frame size we dropped down to doesn't work we reduce the packet size by a half of its existing size. After every failure, this algorithm requires $\log(L - p)$ steps to cross the upper limit L where p is the last successfully acknowledged packet size.

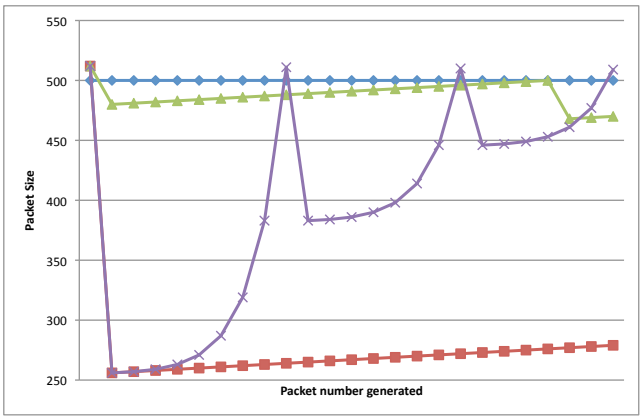

Figure 5: Comparison of packet size variation for the three different flow control strategies. The maximum possible packet size that can be transmitted for the configuration was 500 and is denoted by the blue curve. AIMD is denoted by the red curve; Aggressive by the Purple curve and Stable by the green curve. The graph start just at the point where Slow Start causes a timeout at a packet size of 512.

2.2.2 *Stable*

Like AIMD, the packet size is increased by 1 upon success. However, upon failure, Stable reduce the packet size to a smaller packet size that optimizes the average bandwidth observed during steady state. The calculation of the new value is described below.

Let x be the largest packet size successfully sent. Also assume that in one timeout we can transmit t number of packets. Suppose we tried sending packet size $x+1$ and it failed and we reduce the packet size by n. In such a case the average throughput is:

$$\frac{\sum_{i=x-n}^{x} i}{n+1+t} = \frac{n(x-\frac{n}{2})}{n+1+t}$$

Differentiating the above with respect to n we get the optimal value at,

$$\begin{aligned} n &= \frac{\sqrt{8x+(2+t)^2}-1}{2} \\ n &\approx \sqrt{2x} \end{aligned}$$

Thus, if a packet drop occurs at packet size x the new packet size we choose is $x-\sqrt{2x}$.

Figure 5 shows the behavior of the three flow control protocols for a setup where the maximum packet size that can be transferred is 500 for a setup where the maximum packet size that can be transferred is 500.

2.2.3 *Timeouts*

We experimentally determined that if the input barcode to the decoder library eventually produced a valid output the decoding process took at most 200ms while if the input barcode was too corrupted for decoding, the decoder took more than 1s to time out. Based on these values we chose a value of 3s to timeout. This allows the client to try decoding twice before the server reduces the packet size.

3. EXPERIMENTS

We conduct two kinds of experiments:

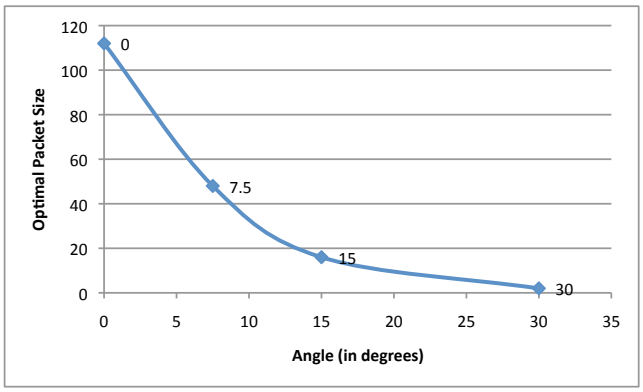

Figure 6: The y-axis represents the packet size with the highest observed bandwidth over 100 packets sent. The distance of the screen and the webcam was kept constant, only the angle between the two planes was changed.

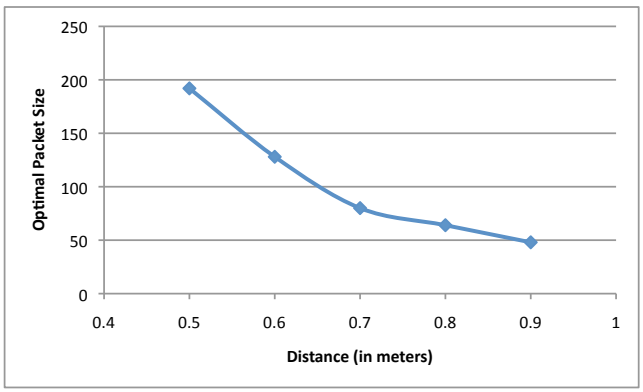

Figure 7: The y-axis represents the packet size with the highest observed bandwidth over 100 packets sent. The viewing angle of the screen and the webcam was kept constant, only the distance between the two planes was changed.

1. We experimentally determine the effective bandwidth of the visible light link capacity as a function of distance, view angle and data matrix packet size.
2. We then evaluate the performance of the three different flow control algorithms we mention in this report: AIMD, Aggressive and Stable.

Our experiments were conducted using the following apparatus:

- The server screen used was a Dell 24-inch full HD widescreen monitor. The pixel area we used was a 900×900 size screen area.
- The client was a 2.4GHz Intel Core 2 Duo processor with 2 GB of DDR2 SDRAM running Mac OS X version 10.6.3
- Both the client and the server received their input using 2 Megapixel Logitech webcam.
- We use the `dmtxread` and the `dmtxwrite` command line utilities from the Open Source Data Matrix Library *libdmtx* [1] version 0.5.2

3.1 Effective Bandwidth Variation

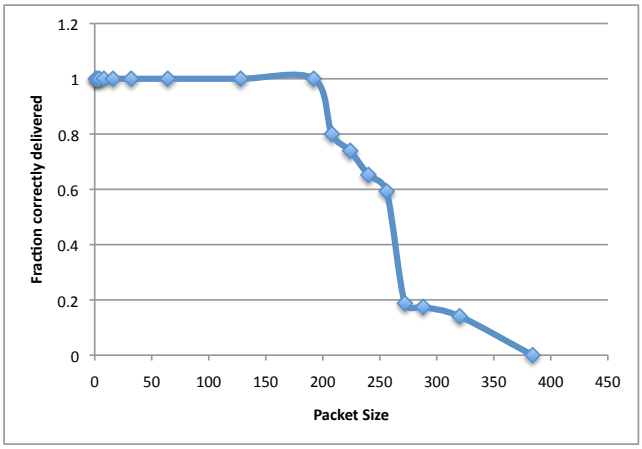

Figure 8: The y-axis represents the fraction of packets sent, in a batch of 100 packets, that were successfully decoded by the receiver. The distance and the viewing angle of the screen and the webcam were kept constant, only the packet size of the barcodes were changed.

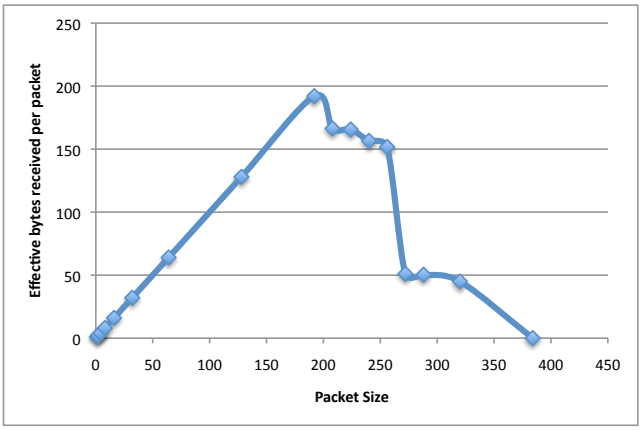

Figure 9: The y-axis represents the effective bytes per packet sent and is equal to the produce of the fraction of packets correctly decoded and the size of the packets. The distance and the viewing angle of the screen and the webcam were kept constant, only the packet size of the barcodes were changed. The total number of packets sent for each size were 100.

We define effective bandwidth as the average bandwidth for a given transmission that also accounts for the bandwidth lost due to wasted transmission time when the server times out.

We conducted three experiments where we individually varied: the distance of the client from the server, the angle of the planes of the client and the server and the density of the data transmitted by the server. Because the display areas is fixed, the density is directly proportional to the number of characters in each of the server's packets.

As observed in Figures 7, 6 the packet size used for the maximum possible effective bandwidth sharply drops as either the distance or the angle is increased. This observation leads us to conclude that visible light data link layer are relatively secure against eavesdropping if the clients are of similar technical specifications.

Figure 8 and figure 9 show the variation in the percentage of successfully transmitted packet and the corresponding effective bandwidth respectively, in a batch of 100, as we increase the packet sizes while keeping the distance and angle constant. The fraction of successfully transmitted packets stays at 1 for a while and then suddenly drops to zero. This is expected since as the density of data matrix codes increase random noises in the camera and the ambience are more likely to corrupt the captured frame at the client enough so that it can not be decoded. A similar observation holds true for the effective bandwidth observed.

Note that the environment, consisting of the distance, angle, and the ambient lighting conditions, defines an upper limit on the packet size that can be successfully decoded by the client. In an ideal case, once we cross this upper limit the fraction of successful packets should drop to zero from one. Indeed, we observe an approximate step function in Figure 8.

3.2 Flow Control Algorithm Evaluation

In this experiment we transmit a file containing 20000 characters using our protocol. We run the three different congestion control algorithm and measure the average time taken for transmission. We take an average of three different runs. The results are presented in Figure 10.

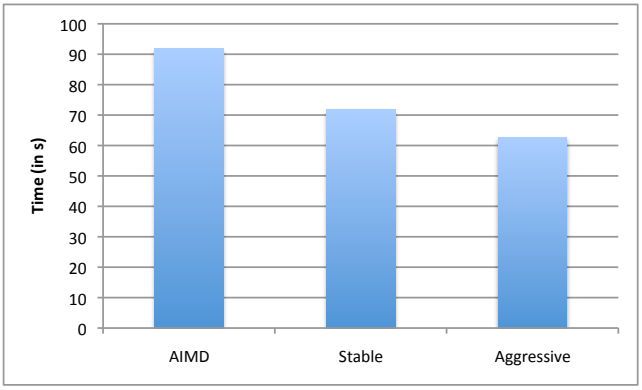

Figure 10: This graph shows the time taken by the three different flow control algorithms to transmit a file with 20K randomly generated alphanumeric characters.

Contrary to theoretical expectations, the Aggressive strategy outperforms AIMD and Stable. Compared to Aggressive, AIMD is 1.47 times slower and Stable is 1.14 times slower. The reason for this observation is that Stable will outperform Aggressive when the ideal step is actually observed and when the the upper limit remains constant. As the distance and angle increase these assumptions are very likely to be violated. Thus an early random packet drop will heavily penalize AIMD and Stable while Aggressive will recover from the random setback to reach the optimal in a logarithmic number of steps.

Note that in none of the cases does AIMD turn out to be the optimal flow control algorithm to use.

4. CONCLUSION

We present a wireless optical system built out of commodity hardware for transferring small amount of data using without wires and securely using visible light. We demonstrate experimentally how security is in built in our mechanism. A key learning is that in the absence of changes in surrounding light,

the performance remains quite steady, which is exploited by the making modifications to the TCP's basic AIMD with slow start protocol. With improvements in hardware and in cell phones, the throughput that can be achieved reliably with this protocol will increase.

Currently our system is designed as a server client model and building a duplex connection will allow the system to be used in many other scenarios. Another line of work could be to dynamically adjust screen brightness in response to changes in brightness of the surroundings.

5. REFERENCES

[1] libdmtx data matrix encoding/decoding library. http://www.libdmtx.org.

[2] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[3] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[4] ISO. International symbology specification maxicode. *ISO/IEC 16023:2000*, 2000.

[5] ISO. Automatic identification and data capture techniques - data matrix bar code symbology specification. *ISO/IEC 16022:2006*, 2006.

[6] ISO. Automatic identification and data capture techniques - qr code 2005 bar code symbology specification. *ISO/IEC 18004:2006*, 2006.

[7] H. Kato and K. T. Tan. Pervasive 2d barcodes for camera phone applications. *IEEE Pervasive Computing*, 6:76–85, 2007.

[8] T. Komine and M. Nakagawa. Fundamental analysis for visible-light communication system using led lights. *IEEE Trans. on Consumer Electronics*, February 2004.

[9] T. D. C. Little, P. Dib, K. Shah, N. Barraford, and B. Gallagher. Using led lighting for ubiquitous indoor wireless networking. *IEEE Intl. Conf. on Wireless and Mobile Computing, Networking and Communication*, October 2008.

[10] G. Woo, A. Mohan, R. Raskar, and D. Katabi. Simple lcd transmitter camera receiver data link. Technical report, Massachusetts Institute of Technology, 2009.