Detecting and Controlling Smart Lights with LiTalk

Jagdeep Singh*
Toshiba Europe Ltd.
Bristol, UK
jagdeep.singh@toshiba-bril.com

Tim Farnham
Toshiba Europe Ltd.
Bristol, UK
tim.farnham@toshiba-bril.com

Dan Watkinson*
University of Bristol
Bristol, UK
dan.watkinson@bristol.ac.uk

Daniele Puccinelli
University of Applied Sciences and Arts of Southern
Switzerland (SUPSI)
Lugano, Switzerland
daniele.puccinelli@supsi.ch

ABSTRACT

The rapid increase in demand for wireless controlled Smart Lighting has created a need to automate the mapping between the identifiers for individual light sources and their physical locations. To control Smart Lights, their IDs and physical locations relative to each other must be determined. Nowadays, skilled technicians perform this process manually, which requires a lot of effort, is time-consuming, and incurs high costs, particularly with non-stationary lights. Visible Light Communication has been presented as a possible solution to this problem. This paper presents an approach based on Visible Light Communication that leverages Machine Learning to automate the mapping process between the identifiers and the relative physical location of Smart Lights. We show that our approach provides a better location-mapping performance compared to existing methods.

CCS CONCEPTS

• **Computing methodologies → Image processing**; • **Networks → Location based services**.

KEYWORDS

Visible light positioning; CMOS sensors; Machine learning.

1 INTRODUCTION

As light-emitting diodes (LEDs) become increasingly ubiquitous in commercial environments due to their energy efficiency, there has been a growing demand for wirelessly controlled Smart Lights. The Smart Lighting market was valued at $11 billion in 2020 and is expected to reach $18 billion worldwide by 2028 [1]. Aside from the clear advantages of remotely controllable lighting, Smart Lights can also enable location-based services through localisation techniques [5]. In order for LED-based Smart Lights to be integrated with control systems, their relative physical locations need to be known. In other words, a mapping between the identifier for each light and its actual location is needed. Currently, this process is carried out manually and is time-consuming, costly, and prone to human error. One solution to automate this process can be provided by Visible Light Communication (VLC), which uses the light in the visible spectrum to transmit information. With VLC, light can be used to encode and transmit the identification information to the camera system, which uses Machine Learning (ML) in order to map the identifiers to the actual light source location. VLC-based localisation has benefits over radio-based approaches, including robustness to interference and multi-path fading [9]. VLC generally uses LED as transmitters and photodiodes or cameras as receivers. LED-to-Camera communication is a more recent form of VLC that uses a CMOS (Complementary Metal Oxide Semiconductor) camera as a receiver rather than a more traditional photodiode. CMOS cameras are commonly available on a range of devices, making it possible to use commercial off-the-shelf devices to build a VLC receiver that can collect data from one or several sources simultaneously. The rolling shutter effect of a traditional CMOS camera can also be exploited in order to decode data that is being transmitted by the light sources [5, 7].

Camera-based VLC can be used to build a system that allows the mapping of new lights to their physical location. Subsequently, this information can be used to determine relative locations by other VLC receivers; it should be noted that most of the work done in state-of-the-art VLC localization approaches is based on the assumption of such prior knowledge. This paper focuses on the initial determination of the relative locations of the lights to enable lighting control and support other location-based services.

---

*Both authors contributed equally to this research.

Figure 1: LiTalk: VLC-based collection & mapping of light IDs.

LiTalk is an initial step in this direction to explore the use of VLC to automate the process of simultaneous mapping and localisation for Smart Lights by predicting the link distance between light sources and camera receivers using Machine Learning. Based on the predicted link distance, the location coordinates of the light sources can be assigned, eliminating the assumption that light source positions are known a priori. With **LiT**alk, we show that the use of Machine Learning can provide benefits compared to existing VLC decoding methods. Figure 1 depicts the **LiT**alk mechanism, where with the deployment of **LiT**alk solution, the technician or autonomous guided vehicles (AGVs) can point a camera towards the installed Smart Lights, which illuminates/transmits their ID. The camera will decode those IDs and further assign the location coordinates to the lights based on the determined link distance. In a nutshell, this paper makes the following contributions:

(1) A reliable many-to-one VLC link that can simultaneously decode the ID of multiple lights.
(2) A fast decoding pipeline for the above-mentioned VLC link, reducing the computation time to an average of 500 ms to determine the identifier of the lights compared to existing methods.
(3) A method to determine the link distance between the light and the camera based on the LED image captured by the camera. More specifically, we train a model to predict the link distance given the captured LED size within the camera frame.

2 LITALK OVERVIEW
In this section, we will first present the modulation scheme we use in **LiT**alk for the transmission of the light IDs and further discuss the image processing steps needed to decode those IDs from multiple lights simultaneously.

2.1 Transmitter
In order to transmit data at a relatively high frequency, any transmitter must be capable of switching its light state at relatively high speeds. Most LEDs can support this fast switching as long as they are connected to a suitable driver circuit. VLC can also be carried out using an LED display as a transmitter[8]. Data in VLC is most commonly transmitted by modulating the state of light to on or off depending on the binary data being transmitted.

2.1.1 Modulation Scheme. In **LiT**alk, basic On-Off Keying (OOK) is used as the modulation scheme for the data. OOK works by representing a binary '0' by turning the light off and a binary '1' by turning it on. Using this method without any additional data manipulation will cause noticeable flickering if many consecutive '0' or '1' symbols were to appear in the data. In order to overcome this, we combine OOK with Manchester encoding of the data. Manchester encoding uses two bits to represent one symbol from the original data. In our case, a '0' is represented by the sequence '10' and a '1' is represented by the sequence '01'. This encoding strategy creates a balanced amount of ones and zeros in the transmitted data that keeps the light at a constant intensity. As it is not possible to have more than two consecutive symbols, this technique results in the absence of any noticeable flicker.

We use packets with a fixed preamble size and variable payload size. We use 5 symbols '10001' as the preamble (the use of 3 consecutive zeros makes it distinguishable from the rest of the data). A start frame delimiter of '01' is also used after the preamble before the payload.

The total size of the packet is calculated using the size of the preamble and the size of the payload. The size of a packet carrying N bits of data is given by Equation 1.

$$\text{Packet Size} = M + S + N * n_{\text{symbol}} \tag{1}$$

where "M" represents the size of the preamble, "S" is the size of the start frame delimiter and "N" is the number of data bits in the payload. "n_{symbol}" is the number of symbols that represent a single data bit after encoding (in our case $n_{\text{symbol}} = 2$).

2.1.2 Modulation Frequency & Packet Size. For successful detection of information bits when using the camera as a receiver, the amount of information encoded in a packet also depends on the transmission frequency. We tested various transmitter frequencies in order to determine the most suitable frequency that maximises the amount of information encoded without creating too much decoding difficulty. Theoretically, there is a maximum transmitter frequency of that, which is half the camera scan rate. This theoretical upper bound would cause a band to have a width of only one pixel on the image. This would make it difficult to decode. In **LiT**alk, the chosen frequency value is 2.5 kHz, this produces

a band width, W of 12 pixels. Modulating at this frequency causes no perceivable flicker and allows for easy decoding of the data. It is calculated that the maximum expected band of the preamble will have a width of 36 pixels. Given this pixel width and our calculated packet size from Equation 1, we can determine the total length in pixels of a packet in a frame.

$$\text{Packet Length in Pixels} = 12 * 23 = 276 \text{ pixels}. \quad (2)$$

2.2 Receiver

Any hardware that can detect the presence of light can be used as a receiver. Traditionally, photodiodes are preferred due to their fast response and high bandwidth. Some researchers have investigated the use of an LED in reverse bias mode [4] and others have used smartphone cameras [3]. Cameras have much better spatial resolution than photodiodes so can give advantages when decoding from multiple transmitters or carrying out localization[5]. In **LiT**alk, we use the Raspberry Pi Camera to decode the ID of multiple lights simultaneously.

Rolling Shutter: Most traditional cameras contain either a CMOS or a *charged-coupled device* (CCD). CCD-based cameras most commonly use the *global shutter* method of capturing images from a scene. This is when all pixels are captured simultaneously so the frame taken represents a single instant in time. On the other hand, most CMOS-based cameras capture images with the *rolling shutter* method. This is an alternative to a global shutter, where an image is taken by scanning across the scene either horizontally or vertically. A rolling shutter mechanism works by exposing each row or column of pixels sequentially. Once all the rows have been exposed, an image is created by merging them together. This means that not all parts of the image are recorded at the same instant in time. Predictable distortions can be produced when capturing fast-moving objects or rapid flashes of light. When building a system that exploits the rolling shutter effect, it is important to carefully choose the camera parameters to obtain the correct *exposure time* as this has a significant effect on the width of the light bands and, consequently, on the ability to decode.

2.2.1 Camera Parameters. When designing a receiver for an LED-to-Camera system there is a range of aspects to consider. This section describes the most important parameters and how they should be set based on the design of the system.

Exposure Time: It is defined as the time that the camera sensor is exposed to the light and is arguably the most important parameter. The exposure time closely relates to the rolling shutter speed of the camera, as previously mentioned. A shorter exposure time will create more definition between the dark and light bands, making them easier to distinguish. Kuo et al. [5] show that independent of film speed (ISO), the best performance is achieved with the shortest exposure time.

(a) Original Frame

(b) Contrast increase and 3x3 blur
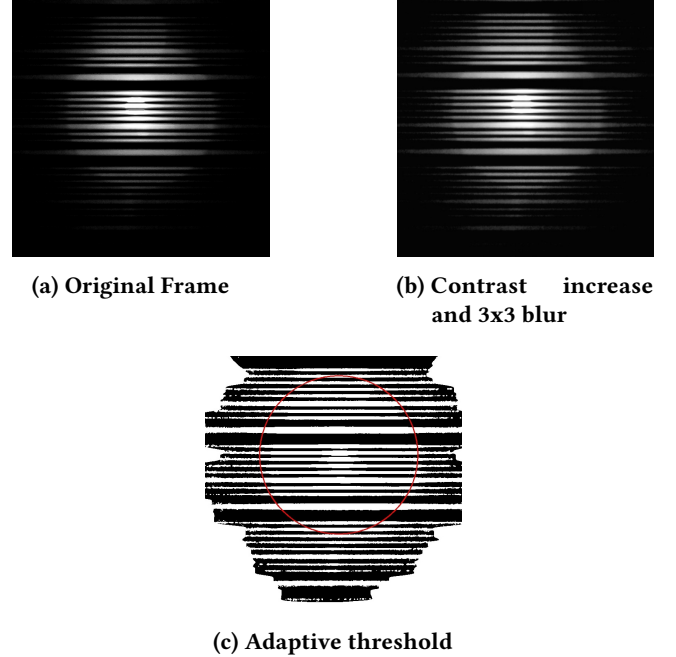

(c) Adaptive threshold

Figure 2: Decoding pipeline. After the contrast increase, we blur the image (b) and apply an adaptive threshold to create a binary image (c); the red circle shows the detected radius

In **LiT**alk, by using a camera connected to a single board computer rather than a smartphone, we have more control over the exposure time of the camera and can set a specific value to reduce the setup needed before transmission.

Film Speed: It is a measure of the sensitivity of the image sensor to light. It can be represented as the amount of photons needed to saturate a pixel. An increased film speed (high ISO) will reduce the amount of photons needed to saturate a pixel. Usually this is used in low light situations. In our case, a slower film speed is preferred, because it increases the definition between bands and reduces noise.

Scan Rate and Resolution: Due to the rolling shutter effect, the frames are captured row by row in a sequential way. The image resolution determines the number of rows in a frame. The scan rate is known as the total time it takes for the sensor to scan all the rows of the frame. The *camera scan rate* can also be thought of the sampling rate of the sensor. The total scan rate of the camera is determined by the time it takes to scan a single row of pixels. A resolution of 1280x720 px was chosen for the images. It was decided not to use Full HD (1920x1080) in order to reduce processing time and memory requirements. The readout time T_r is the time it takes for the sensor to read a single row of pixels. In [6], it is shown that the width W of each band of light can be given by Equation 3, where f is the frequency of modulation.
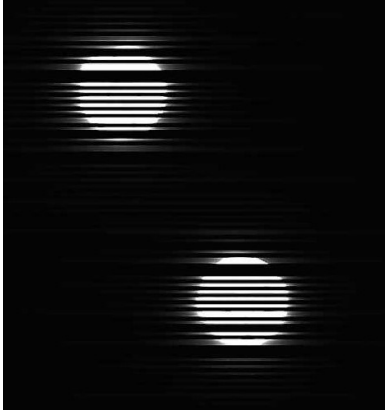
Figure 3: Captured LEDs using rolling-shutter in a single frame.

$$W = \frac{1}{2fT_r} \qquad (3)$$

Equation 3 indicates that the width of the light bands does not depend on either the size of the LED or the distance from the transmitter. With the current parameters & using Equation 3, the readout time calculated as $16.67\mu s$.

*2.2.2 **Image Processing**.* The receiver uses image processing techniques in order to detect the transmitter and decode the data. We use the image processing pipeline as in Luxapose[5]. An additional contrast increase is used as in DynaLight [10].

Decoding: We adapt the method proposed in DynaLight-Vasilakis [10] to decode the images captured by the camera. This includes the steps of blurring with a 3x3 kernel and then using an adaptive threshold to create a binary image. In the center of the detected area, where the light is located, the intensity is much higher than in other parts. This causes some of the pixels in this location to become overexposed and create bands that are larger than those expected, making it harder to decode. In order to overcome this, we implement a row offset method. Instead of selecting the center column of pixels for decoding, a column offset of half of the circle radius is used for selection.

3 IMPLEMENTATION

To evaluate the performance of our **LiTalk** design, we implement a prototype system using a transmitter circuit with off the shelf components and a commercially available single board computer together with a camera module.

Transmitter: On the transmitter side, we use an Arduino Uno R3 to perform modulation, control and encoding using an LED and a custom circuit. We can achieve the desired frequencies by using the built-in delay functions to create time periods between the on and off states of the light. We use this to achieve a modulation frequency of 2.5 kHz. The

transmitter always repeats the same packet on a loop until deactivated. The only information transmitted is the hardware identifier, so a simple solution is to repeat packet transmission until the receiver can carry out decoding to avoid the need for synchronisation across the link. The circuit used has been adapted from the circuit used in DynaLight [10]. We employ the X-Lamp MC-E by Cree, which contains 4 LED dies on an integrated circuit. The maximum current for each die is 700 mA.

Receiver: Originally, the implementation was started using a specialised camera designed for machine vision applications, the OpenMV[1]. After some testing with the camera, we realized that the built-in processor was not powerful enough for the required image processing techniques. The final solution employs a Raspberry Pi 4 Model B single board computer. This is connected to the Raspberry Pi HQ camera[2]. The Raspberry Pi contains a Broadcom Quad-core 1.5 GHz processor, making it much more powerful than the OpenMV's single core 480 MHz processor. Using the Raspberry Pi allows us to capture and process images on the same device, removing the undesired USB serial latency that was occurring. The Raspberry Pi runs its own operating system based on Debian, which provides all the necessary software that is utilised in the system.

3.1 Mapping Method

The first step for mapping the light ID to their physical location coordinates, with respect to the camera, is to know the elevation angle or the link distance between the camera and light source. In **LiTalk**, we use the following approach to calculate the link distance.

Method 1 - Conventional decoding: In rolling-shutter-based LED-to-Camera communication, the number of bright pixels that appear on the image depends on the link distance between the light source and camera. Mathematically taken from [2], the relation between distance, number of pixels and camera factors can be represented as:

$$D_{\text{link}} = f_L + \frac{S_r f_L}{S_{ip} S_p} \qquad (4)$$

where D_{link} is the link distance between the camera and LED, f_L is the focal length of the camera (equals to 0.28 mm in our case). S_r is the radius of the LED (taken as 10.9 mm in **LiTalk**), S_{ip} is the size of LED in terms of pixels and S_p is the length of a single pixel. In **LiTalk**, we use Equation 4 to determine the link distance from two light sources simultaneously.

Method 2 - LiTalk: We employ a linear regression model to calculate the D_{link} to improve the link distance estimation accuracy. We train the model for different sizes of LED

---

[1]https://openmv.io/products/openmv-cam-h7-plus
[2]https://www.raspberrypi.com/products/raspberry-pi-high-quality-camera/
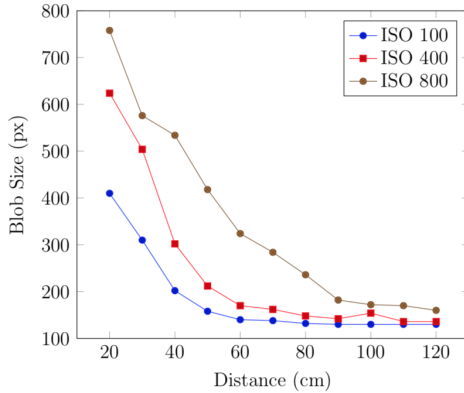
Figure 4: Blob Size vs Distance and ISO. Each line represents a different ISO (film speed) value.

received on the image sensor and for different known link distances. The training was performed in a dark room with no ambient light sources. During the evaluation, we first process the image of the received light (an example using two LEDs is shown in Figure 3), calculate the blob size, i.e. LED size appears on the image, and predict the distance.

Based on the calculated distance from two LEDs and with the known camera location and separation between the two LEDs, the location coordinates to LEDs can be calculated and assigned to map the lights to their physical location. In **LiT**alk, we have only implemented the former part, and the latter part is left for future work. The results and evaluation for distance estimation using two methods are described in Section 4.

4 EVALUATION

In this section, after a discussion of how film speed can affect transmissions over distance and how it can be best tuned, we evaluate the distance estimation of the prototype system.

4.1 Transmitter

Duty Cycle: Our preliminary experiments showed that the dark bands on the frame produced by the camera are much thinner than the bright bands due to the extra time needed to charge and discharge the LED. We addressed this issue by making the off time slightly larger than the on time, i.e., by changing the duty cycle of the signal to 40%.

4.2 Receiver

*4.2.1 **Effect of Film Speed on Distance**.* Our preliminary experiments with the complete implementation showed that the system found it more difficult to decode data at longer distances when using a fixed length packet. This is because, as the distance between the transmitter and receiver increases, the blob size provided by the detection pipeline decreases.

Therefore, there are fewer pixels to decode in an image, and using a fixed length packet imposes a lower bound on the blob size to enable decoding. The authors of Luxapose [5] state that it is best to keep film speed to a minimum to ensure reliable decoding. They mention that increasing the film speed can help the decoding process at longer distances but is conducive to images with more noise. Therefore, it is important to look into the relationship between film speed (ISO), distance and the blob size.

To this end, we carry out an experiment using three different ISO values; 100, 400 and 800. The distance between the receiver and the transmitter is measured, and the diameter of the blob is recorded. Figure 4 shows that, as the distance is increased, the blob size decreases, making size estimation less accurate. This decrease in blob size can be partially overcome by increasing the film speed of the camera. It is seen that using an ISO value of 800 results in the maximum blob size at all distances. We can also calculate a lower bound of the blob size due to the fixed packet size i.e., 276 pixels (using Equation 2). The graph shows that when using an ISO value of 100 or 400, the blob size becomes less than that value at around 40 cm, whereas when using an ISO value of 800, this point is not reached until around 70cm. As the performance of **LiT**alk for distance estimation depends on the blob size, optimising the camera parameters such as ISO evaluated in this section is vital.

4.3 Performance Analysis

*4.3.1 **Computation Time**.* Our experiments show that **LiT**alk takes between 0.3-1.3 seconds to decode a frame (containing two LEDs) with an average time of 500 ms. This is much lower than the total time taken by other systems such as Luxapose [5]. The time to decode varies based on the scheduling methods of the kernel running on Raspberry Pi.

*4.3.2 **Distance estimation**.* Using conventional decoding methods (Method 1 as described in Section 3.1) [2], our analysis shows that the error is minimized only at a specific distance, i.e. when the received LED image has no blooming effect and no blurriness. Further, the error is inconsistent and increases with distance. For shorter link distances, the average error is more than 2 cm and the pattern in the error plot for distance estimation is not linear, as illustrated in Figure 5a. The mean squared error calculated with this approach is greater than 10cm.

In Section 4.2.1, we have seen how the camera parameters and link distance affect the blob's size. This implies that it is possible to estimate the link distance based on the blob size. We employ our regression model (Method 2 as described in Section 3.1) to perform the analysis at various distances and predict the link distance based on the received blob size while simultaneously decoding the light ID. The pink
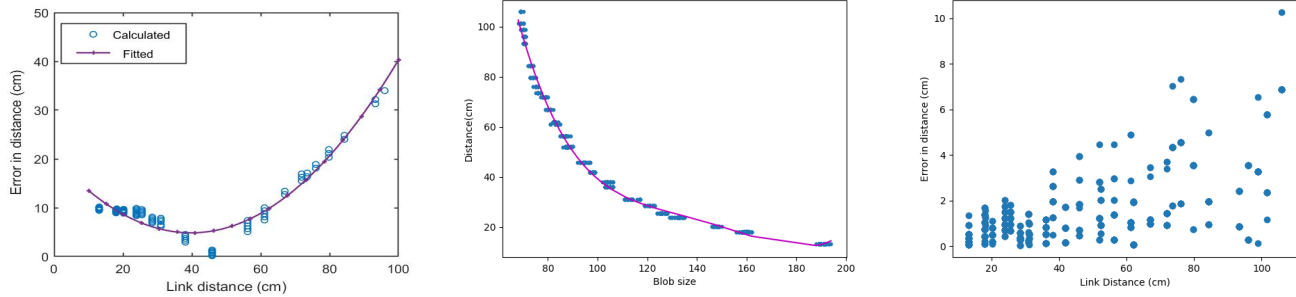
(a) Error in link distance using conventional mathematical method.

(b) ML predicted distances for given LED size.

(c) Measured error in link distance using regression.

Figure 5: LiTalk link distance analysis.

curve in Figure 5b shows the predicted distance, and the error in the predicted distance for various positions is shown in Figure 5c. The mean squared error calculated using ML model is 1.9734 cm, resulting in a five-fold improvement compared to the existing method (see Figure 5a & 5c). Further, performance comparison with the SOA work can only be made after full implementation of **LiT**alk for localization service as the estimation distance is the initial step. As part of our future work, we plan to train the model to decode the LED IDs, identify the link distance within a frame, and provide location services.

5 CONCLUSION

With an increased demand for wireless LED control, camera-based VLC proves to be a potential solution to the ID matching problem. This is defined as the requirement to match hardware identifiers of light fittings to their locations. In this paper, we have proposed a way to address this problem using off-the-shelf available cameras and a link distance estimation regression model. Our results show a five-fold improvement in the distance estimation mean squared error compared to existing methods. Furthermore, we have developed a fast decoding pipeline to reduce the frame decoding time to an average of 500 ms.

ACKNOWLEDGEMENT

REFERENCES

[1] Bloomberg. 2022. Smart Lighting Market size. https://www.bloomberg.com/press-releases/2022-07-05/smart-lighting-market-size-worth-18131-31-million-globally-by-2028-at-6-32-cagr-verified-market-research

[2] Omer Dalgic, Daniele Puccinelli, and Marco Zuniga. 2022. Enabling Body-Centric Computing Applications with LED-to-Camera Communication. In *Proceedings of the 2022 Workshop on Body-centric Computing Systems*. 7–12.

[3] Christos Danakis, Mostafa Afgani, Gordon Povey, Ian Underwood, and Harald Haas. 2012. Using a CMOS camera sensor for visible light communication. In *2012 IEEE Globecom Workshops*. IEEE, 1244–1248.

[4] Hyunwoo Jung and Sung-Man Kim. 2020. A Full-Duplex LED-to-LED Visible Light Communication System. *Electronics* 9, 10 (2020), 1713.

[5] Ye-Sheng Kuo, Pat Pannuto, Ko-Jen Hsiao, and Prabal Dutta. 2014. Luxapose: Indoor positioning with mobile phones and visible light. In *Proceedings of the 20th annual international conference on Mobile computing and networking*. 447–458.

[6] Hui-Yu Lee, Hao-Min Lin, Yu-Lin Wei, Hsin-I Wu, Hsin-Mu Tsai, and Kate Ching-Ju Lin. 2015. Rollinglight: Enabling line-of-sight light-to-camera communications. In *Proceedings of the 13th annual international conference on mobile systems, applications, and services*. 167–180.

[7] Lingkun Li, Pengjin Xie, and Jiliang Wang. 2018. Rainbowlight: Towards low cost ambient light positioning with mobile phones. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. 445–457.

[8] Tianxing Li, Chuankai An, Xinran Xiao, Andrew T Campbell, and Xia Zhou. 2015. Real-time screen-camera communication behind any scene. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. 197–211.

[9] Jagdeep Singh and Usman Raza. 2020. Passive visible light positioning systems: An overview. In *Proceedings of the Workshop on Light Up the IoT*. 48–53.

[10] M. Vasilakis. 2015. DynaLight: A Dynamic Visible Light Communication Link for Smartphones. https://doi.org/uuid:6dc20955-c2ff-48b4-aa3c-1f91dffccba0