Delft University of Technology
Master's Thesis in Embedded Systems

DynaLight: A Dynamic Visible Light Communication Link for Smartphones

Michail Vasilakis

embedded software

TU Delft Delft University of Technology

DynaLight: A Dynamic Visible Light Communication Link for Smartphones

Master's Thesis in Embedded Systems

Embedded Software Section
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

Michail Vasilakis
m.vasilakis@student.tudelft.nl

11th December 2015

Author
Michail Vasilakis (m.vasilakis@student.tudelft.nl)
Title
DynaLight: A Dynamic Visible Light Communication Link for Smartphones
MSc presentation
18th December 2015

Graduation Committee

Prof. Dr. K.G. Langendoen (chair)	Delft University of Technology
M.A. Zuniga Zamalloa, PhD	Delft University of Technology
Dr. Judith Redi	Delft University of Technology

Abstract

Nowadays, Visible Light Communication (VLC) has attracted the attention of the scientific community due to its great potential in creating smart communication links. Exploiting visible light modulations, could in time enable Internet connectivity via light lamps. Recent research studies have shown that modern smartphones have the ability to capture high frequency light patterns and increase the applicability of VLC links, enabling smart applications. However, creating flexible camera-based VLC links brings-up several challenges that are introduced by the diversity of the available devices.

Firstly, existing VLC systems offer inflexible setups that are designed to operate at *fixed* distances. This fact causes problems when it comes to varying the distance between the transmitter and the receiver. This thesis introduces *DynaLight*: an adaptive line-of-sight VLC system for smartphones that dynamically adjusts and maximizes its channel capacity by estimating the distance between the transmitter and the receiver.

Secondly, the wide diversity in smartphones' hardware introduces problems when it comes to implementing a generic VLC link for market smartphones. In order to increase the applicability of our system, we chose to utilize inexpensive hardware, that introduce performance limitations, such as limited camera control. We present an image processing pipeline that identifies and overcomes effects that are caused by off-the-shelf hardware, and we further increase the amount of information, that can be extracted, by 40%.

Last but not least, we develop a smartphone application that implements our enhancements and draws attention to synchronization challenges. Our conclusions indicate that the applicability of smartphone VLC links will be further extended due to the rapid evolution of modern smartphones.

Preface

The current Master thesis concerns the final part of my Master of Science in Embedded Systems at Delft University of Technology. I had always been passionate about smartphones and their capabilities to improve our everyday lives. During the first year of my master studies, I had the chance to participate in the Smart Phone Sensing course offered by the Embedded Software group in TU Delft. This is when I realized my interest in smarphone sensing, as well as in its potential to change people's lives. I was thrilled when I first read a paper on Visible Light Communication and realized that this was the topic that I wanted to master. Furthermore, it is a field that combines communication theory along with optics, smartphone sensing and image processing which makes it challenging to work on. During these eight months I was exposed to scientific fields that I had no previous experience and had the opportunity to remarkably improve my scientific skills.

I would like to express my deepest gratitude to my supervisor, Marco Zuniga, for giving me the opportunity to work on this project. During our meetings he showed me not only how to do research and have a critical thinking, but also how to be confident and manage my stress. With his knowledge and experience he helped me improve my writing and presentation skills. Next, I would like to thank Ioannis Protonotarios for assisting me with all the hardware related issues and for being so patient with me. Many thanks also go out to all my fellow colleagues Marco, Dimitris, Platon, Roshan, Coen and people from"The 9th Floor" for their ideas and feedback. I would also like to thank Katerina and Andri for their valuable comments and support in improving the current report.

Last but not least, I would like to thank my family and especially my parents for their unselfish love and support throughout my entire life.

Michail Vasilakis

Delft, The Netherlands
11th December 2015

Contents

Chapter 1

Introduction

Wireless communication is considered, by any measure, the fastest growing segment of the communication industry. During the last decades of the 20th century, wireless communication technologies emerged and became part of people's every day life. The most well-known example is cellular phones whose growth and popularity grew exponentially. Wireless communication involves transmitting information over a distance, without utilizing cables or any other form of electrical conductors but by exploiting radio waves. The major advantage of radio waves is their ability to be transmitted though short distances, such as a few meters for television control or as far as thousands or even millions of kilometers for "deep-space" radio communications. Some example applications of radio wireless technology concern cellular phones, WiFi, GPS units, satellite television and many more.

Optical wireless communication (OWC), is a subtype of wireless communication, in which information is transferred through light and not by any physical medium or radio frequencies. Communication utilizing light is not a new concept. It was back in 1880 when Alexander Graham Bell and Charles Sumner Tainter invented and patented the photophone, a wireless telephone that conducted audio conversations wirelessly over modulated light beams. A similar concept was used in the late 19th century when signal lamps were pioneered by the British Royal Navy. During that time big sign lamps were exploited to transmit Morse codes between naval ships, without using radio signals. The operators of the lamps were turning on and off the lamp in order to create the known Morse codes, while an observer was decoding the received light pattern.

Recently, transmitting data through visible light has attracted the attention of the research community. Therefore, *Visible Light Communication* (VLC) became a very promising technology that can enable or facilitate various applications. Visible light concerns frequencies between 400 and 800 THz (780375nm) in the electromagnetic spectrum, which is 10,000 times larger than the entire radio frequency spectrum [17]. VLC has become popular

due to the recent developments in Light-emitting diodes (LEDs), that have enabled us to utilize them in creating light patterns. Compared to the previously mentioned sign lamps, LEDs can be computer controlled and turned on and off at such high rates that the emitted light "seems" constant, meaning that light patterns are not perceivable by humans. By utilizing light sensors that can identify these "invisible" light patterns, high-speed VLC links can be created by taking advantage of the properties of light. Directionality, diffusion and reflections, are some of the properties one can exploit when it comes to creating VLC enabled applications. Thus, VLC can be summarized as illumination plus communication.

We can create VLC links by utilizing several transmitters and receivers. LEDs and LCD screens can be used as transmitters while LEDs, photodiodes or cameras can be used as receivers. In this work, we focus on *LED to Camera communication.*

Nowadays, LED lighting is becoming increasingly popular in both indoor and outdoor environments, which enables creating smart applications using VLC links. VLC research focuses on creating high-speed data links, that can provide Internet connectivity via lamps and lead to Internet of Lights. Furthermore, it is proved that VLC can also provide highly accurate localization (sub-meter accuracy) compared to RF-based approaches (2-7 meters) that facilitate in accommodating location-based services. Another interesting VLC application is smart-toys, in which VLC-enabled toys and devices, such as smartphones, interact so as to trigger sound or lighting effects, making toys more fascinating [4]. Nevertheless, some the aforementioned ideas are still in research level, due to the numerous challenges which have inspired various recent research projects. Nevertheless, pureLiFi [9] demonstrated the first commercially available Li-Fi (similar to Wi-Fi) system in 2014, which indicates that more commercial VLC applications will emerge in the near future.

1.1 Motivation

Building a camera-based VLC system has several challenges that vary from data encoding schemes, modulation techniques and decoding methods, to coverage, applicability and hardware. The currently implemented VLC systems are designed to operate at fixed distances, meaning that their parameters are selected on the design phase. Having such a inflexible setup causes problems when it comes to varying the distance between the transmitter and the receiver, due to the fact that several parameters have to be modified. This fact highlights the motivation behind this project which is *the need of an adaptive camera-based VLC system, that will determine its properties based on the distance between the transmitter and the receiver.* We mainly focus on trying to understand how distance affects the feasible channel ca-

pacity, in order to maximize the amount of information that the transmitter can send with respect to distance.

Moreover, it is a well-known fact that smartphones come with different hardware and capabilities. More specifically, modern smartphones have variations in the equipped camera sensors, the processing power, as well as, the running operating system. These factors affect the quality, speed, flexibility and reliability of the created link. This wide diversity introduces problems when it comes to building a universal camera-based VLC link for market smartphones. Some of the current systems utilize high-end smartphones to overcome the aforementioned challenges.

In this thesis, we choose to address the above challenges using inexpensive hardware, which in our opinion increases the applicability of VLC links and highlights performance limitations. To sum up, the goal of this project is to *create an adaptive to distance and reliable VLC link for various types of smartphones.*

1.2 Contributions

In this work, we present *DynaLight*, a line of sight (LOS) camera-based VLC system, that tries to overcome the aforementioned challenges. This thesis delivers the following contributions.

- We present a system that is able to maximize its channel capacity based on the transmitter-receiver distance. (Section 7.1)
- We implement a decoding pipeline that enhances image processing to increase channel capacity by 40% and overcomes problems that are introduced by the limited camera control of low cost smartphones. (Section 6.2)
- We developed a smartphone application that implements a protocol based on the above. (Section 7.2)

1.3 Thesis Organization

The rest of the thesis is organized as follows: Chapter 2, introduces VLC systems, while Chapter 3 presents the related works that inspired this work. Chapter 4 presents an overview of DynaLight and its building blocks. We present a detailed description of the transmitter and receiver in Chapters 5 and 6 respectively. In Chapter 7, we present a thorough explanation of the dynamic link. The evaluation of the implemented system can be found in Chapter 8, followed by Chapter 9, which summarizes our conclusions and

ideas for future work. Finally, in the Appendix A, we give a short description of the limitations that we encountered while experimenting with different transmitters and the interaction with a company in order to make our effort a product.

Chapter 2

Background

This background section presents the basic building blocks and concepts we need to understand for this thesis. Section 2.1 briefly presents the types of VLC systems with respect to the possible receivers, discussing the advantages and disadvantages of each choice. Furthermore, Section 2.2 presents the most significant camera effect that enables smartphones to decode light patterns, the rolling shutter.

2.1 Transmitter-Receiver Alternatives

Creating a communication system involves having a transmitter and a receiver. VLC systems can exploit different light sources and light sensors, in accordance with requirements of the environment of application. More specifically, light sources have to be able to be turned on and off at high frequencies, in order to create invisible (to human) light patterns. LEDs are commonly utilized in VLC systems, due to their ability to be toggled within a few microseconds since they are semiconductor devices. Similarly, apart from LEDs, recent studies have shown that LCD-screens, can also be utilized as light sources.

The light sensors that can be used as receivers are regular LEDs, photodiodes and cameras. Each option has different advantages and disadvantages depending on the system's requirements. Regular LEDs, can operate as light sensors in reverse bias mode and convert light to current, but they have very limited coverage. On the other hand, photodiodes are much more sensitive than LEDs and can be very responsive to high frequency modulations, which facilitates in creating high rate communication links. However, both solutions offer low sensing accuracy with respect to noisy environments where there is high (ambient) light intensity. Furthermore, high frequency light patterns can be recognised by high-speed cameras or smartphone cameras. Smartphone cameras, can capture many images using the *rolling shutter* capturing method that is thoroughly explained in the

following section, which enables them to identify light patterns. However, modern smartphones have variations in the equipped camera sensors, the processing power, as well as, the running operating system. These factors affect the quality, speed, flexibility and reliability of the created link. For example, the operating system affects flexibility with respect to which camera parameters can be controlled by the user, such as the image's brightness. Moreover, recent and high-end devices are also equipped with high quality sensors offering high-resolution images. Those two features are of great importance when it comes to detecting light patterns. *Therefore, the variation in the used hardware and camera properties makes the implementation of universal camera-based VLC links, hard and challenging, especially when it comes to utilizing off-the-shelf devices.*

VLC systems are mainly categorized based on the previously mentioned transmitters and receivers. In this work we focus on LED to (smartphone) camera communication. We believe that this combination offers the highest potential with respect to building real world VLC applications. The decision is based on the fact that LEDs are low cost and commonly used solutions for lighting, and that the majority of modern smartphones are equipped with embedded Complementary Metal-Oxide Semiconductor (CMOS) cameras, that use the rolling shutter capturing method.

2.2 Rolling Shutter

Modern smartphones are equipped with CMOS sensor cameras. This specific type of cameras uses the *rolling shutter*, which is an image acquisition mechanism. In order to understand the importance of this mechanism, one should compare it with the *global shutter* mechanism. The global shutter can be found in both film-based and digital cameras. In the global shutter mechanism, the whole picture is first exposed and then captured all at once. On the other hand, in rolling shutter, the image is captured in a *row-sequential* way. This means, that each frame is divided into rows forming an array of pixels and each row is first exposed and then read by the sensor. Once all the rows are read, they are merged together to form a single image. The comparison between the two capture methods can be found in Figure 2.1. The speed that rows are captured at, is named *rolling shutter speed* and it is relative to the exposure time of each row.

Rolling shutter causes a very interesting effect when it comes to capturing fast moving objects. A simple example is shown in Figure 2.1. The cause of that effect is the fact that the depicted fan is rotating faster than the rolling shutter speed. As a result, we observe distortions, which are caused by the fact that the object is captured in overlapping positions. This means, that between two consecutive row exposures, the object has moved significantly.

Similarly, if an LED is flashing in a frequency that is faster than the rolling

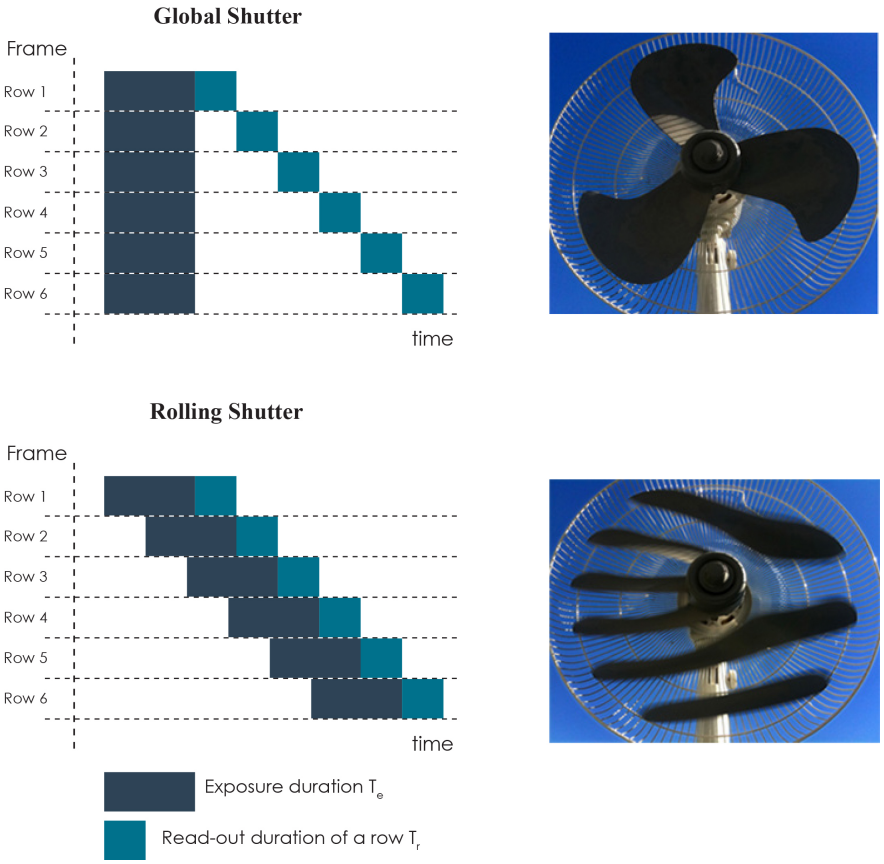


Figure 2.1: **Global vs Rolling shutter.** In the rolling shutter the image is captured in a row-sequential way and not at once, as in global shutter. Distortions appear when the fan is rotating faster than the rolling shutter speed, as can be observed in the bottom right image.

shutter speed, bands (distortions) of different light intensity appear on the captured image as can be observed in Figure 2.2. More specifically, when a row is exposed, while the LED is on, then the whole row of pixels will appear as white. On the other hand, when the LED is off then we will observe a dark row of pixels. When the duration of the LED state is longer than the scan time of each row, the pixel rows form white or dark bands. The band's intensity and width depend on the transmitter's frequency as well as the camera's properties. However, if the flashing frequency of the LED is very fast then LED states might be missed as we will present in Section 6.3.2.

It can be observed from the previous descriptions that the *exposure time is the key camera property*, that determines the rolling shutter speed. The importance of this property will be further explained in Section 6.1, along with other important camera properties that enable us to capture and identify

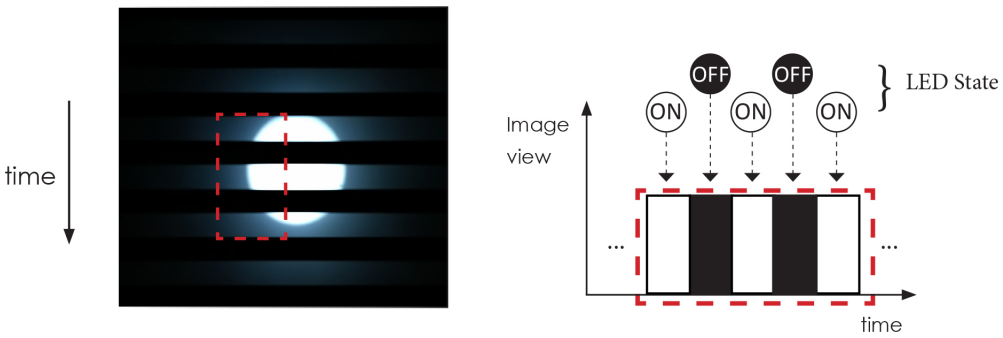


Figure 2.2: **Band formation and LED state.** The size (width) of each band, corresponds to the duration of the LED state (on or off).

the transmitted light patterns (LED states).

Modern smartphone cameras are able to capture around 20 to 30 frames per second which is called *frame rate*. As a consequence, by exploiting the rolling shutter effect, along with capturing multiple frames per second, the smartphone can obtain multiple encoded data (LED states). However, modern smartphones do not offer a stable frame rate. For example, the frame rate of Samsung Galaxy S3 fluctuates between 21 and 29 frames per second. This phenomenon introduces synchronization problems between the transmissions and receptions. Firstly, a fluctuating reception rate could lead to fragmented data between more than one frame. In addition, due to the discontinuity of the receptions there is high probability of missing data. Thus, any camera-based visible light communication link has to take into consideration these synchronization challenges, when it comes to building a reliable communication link.

2.3 DynaLight's Requirements

In this section we present the basic requirements that our system needs to meet. This thesis presents DynaLight, a camera-based line-of-sight visible light communication system. The system requirements, are mainly focused on creating a flexible, reliable, and flicker free visible light communication link for various smartphones.

Adaptive to Distance: The goal of this project is to create a flexible system that has the ability operate in different distances. Distance adaptation regards the creation of a link that can modify its properties, in accordance with the distance between the transmitter and the receiver. This means, that the channel capacity of the link should change with respect to the detected distance. Our system should be able to estimate the distance between the transmitter and the receiver and adjust the channel capacity

accordingly. The smaller the distance, the greater the channel capacity, due to the fact that the transmitter is closer to the camera. In this case, the amount of information that can be extracted increases. The system has to be able to calculate the feasible amount of information that can transmit in any distance.

Reliable: Reliability can be achieved in different levels. In this work, we relate the channel reliability with overcoming the problems, introduced by the camera-based VLC links. As has been mentioned in the previous sections, the wide diversity in the used camera hardware introduces challenges from controlling the camera's properties to synchronizing the transmissions with the receptions. In addition, our system should operate in such a way so as to avoid introducing flickering effects that can be annoying to humans.

Utilize inexpensive hardware: We aim at creating a low cost system, that can support inexpensive and resource constrained platforms both for the transmitter and the receiver. As a result, we prefer to avoid exploiting high-end hardware not only for cost reasons but also for identifying possible limitations and increase our system's applicability. It has to be clarified that in this project we do not use the flash LED of the phone or any other external flashing device. More information about this decision is provided in Appendix A.

Chapter 3

Related Work

This chapter discusses the visible light communication projects that inspired our work. In Section 3.1, we begin by presenting some of the key prior research works in the field of visible light communication. Then, in Section 3.2, we focus only camera-based research works. Both Sections 3.2.1 and 3.2.2, thoroughly discuss related projects that create data communication links, utilizing LEDs or LCD screens accordingly. Moreover, Section 3.2.3, discusses camera-based works designed for indoor localization. Finally, in Section 3.2.4, we present a short overview of the projects that influenced this thesis.

3.1 VLC projects using LEDs or Photodiodes

The majority of the early VLC research work is mainly focused on creating high-speed data links, using specialized hardware [20] and photodiodes. In these projects, RGB [20, 30] and phosphorecent white LEDs [14, 15] are utilized. Furthermore, variations of OOK modulation schemes are used, as well as more complicated schemes such as QAM or DMT/OFMD modulation [12, 20]. Photodiodes were exploited as receivers, due to their high bandwidth, that can support complicated modulation schemes.

In 2012, the IEEE, published a VLC standard known as 802.15.7 [27], which specifies the hardware, modulation and channel coding for various applications. More recent projects focus on LED to LED communication [13, 29, 31, 32] and LED to photodiodes communication [19]. The goal of these projects is to create efficient, low cost and high-rate VLC links.

3.2 Camera-based VLC

This section focuses on projects that use cameras as receivers for both creating data communication links and localization. *All the following camera-based related projects were published after 2012, which highlights that the*

present thesis deals with currently open research problems and recent engineering work.

3.2.1 LED to Camera Communication

LED to camera communication systems, have been investigated in several recent works, focusing on smartphone implementations.

In [11], the authors present how a smartphone camera can be utilized as a receiver using the rolling shutter capturing mechanism. The entire work focuses on how to transmit data between a LED and a smartphone camera. The main problems that are addressed in this work are, firstly realizing how the camera sensor operates and then how to modulate data, so that they can be captured and decoded by the receiver. The authors extensively discuss the rolling shutter effect, which is used by the CMOS cameras and explain the row-sequential capturing method. Then, they explain the utilized OOK and Manchester encoding and decoding scheme, achieving a data rate of 1-3.1kbps, using 640x480 images and 20fps. However, the proposed system introduces noticeable flickering while operating on the visible range of 40Hz. The authors overcome the flickering by imposing a DC bias on the signal, which decreases its dynamic range and the SNR at the receiver, but requires a more complex driving hardware. Another drawback is that the system operates only at close proximity (8-9cm), which reduces the link's flexibility and applicability. Despite the several drawbacks, the main contribution of this work is that it provides a proof-of-concept of camera-based VLC in mobile phones. We also take advantage of the rolling shutter effect and the OOK and Manchester encoding. The main differences with our work are:

- Our system operates above frequencies that cause direct or indirect flickering (>2 kHz).
- Our system is adaptive to distance and can operate from 20-120cm.

A similar study was conducted in [28], that uses undersampled frequency shift on-off keying (UFSOOK) to overcome the flickering effect that was presented in the previous study. However, the authors achieve a very low data rate of 0.5 bits per frame, in extremely low SNR conditions, which is considered very limited and not applicable.

RollingLight [22], is a very recent work (2015) in the field of LED to camera communication. The authors present a line of sight to camera communication system. The main goal of this study is to overcome the diversity, that is introduced by off-the-shelf smartphones that use the rolling shutter. This diversity is caused by the heterogeneous sampling rates of the phones, that cause difficulties in synchronizing the transmitter and the receiver. The authors extensively studied and found an unpredictable and varying idle time gap between two consecutive frames. This phenomenon

can cause dynamic signal losses, which results in unreliable communication. Moreover, the authors present the two problems that are caused by the sampling rate diversity, which are described as mixed symbols/packets and symbol/packets losses.

The first problem occurs when the frame duration is greater or smaller than the packet duration as can be seen in Figure 3.1. This means that a single frame may contain information from more than one symbols/packets. Thus, having an unknown time gap between the frames results in not being able to receive a complete symbol/packet.

The second problem is caused by the camera's discontinuous receiving as can be seen in Figure 3.2. If the gap's length is more than the packet duration then a packet could be entirely dropped resulting in dynamic symbol/packet losses.

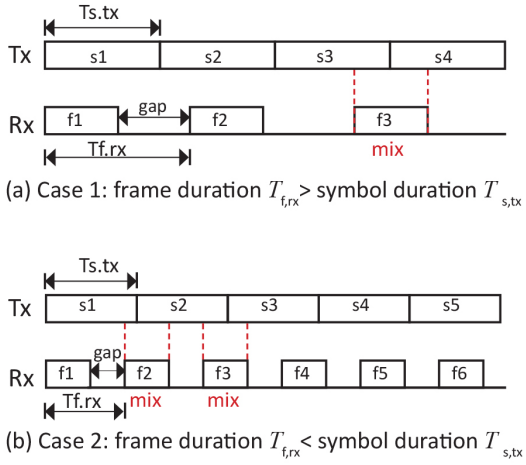


Figure 3.1: **Mixed symbols problem.** Due to the fluctuating frame rate the captured frames may contain more than one symbol or packet.

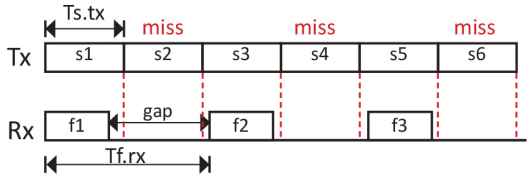


Figure 3.2: **Missing symbols.** If the time gap between the frames is too big there is a high chance of missing symbols or packets.

The authors overcome the first problem by adding splitter symbols between the data symbols. They also add parity symbols to detect, but not correct,

errors and overcome the second problem. Both solutions are facilitated by the fact that they utilize a very recent high resolution phone (iPhone 5s) and a big light source of 60cm×60cm for their experiments, which assist in achieving a high data rate and coverage (light size). This means that they can achieve a big bandwidth that enables them to have packets that include splitter symbols, sequence numbers and parity bits. Finally, the authors implement a real-time iOS application that can achieve a data rate of 11.32 bytes per second.

In our project we take advantage of:

- The unsynchronized communication problem formulation.
- The band-width analysis, but we extend it and show how the camera parameters affect the size of the detected bands.

Furthermore, our work differs in the following topics:

- We overcome the mixed packets by making sure that a complete packet can be decoded from every captured frame.
- We try to synchronize the transmission by minimizing the effect of the time gap jitter.
- We do not use high-end smartphones or big light sources.

Exploiting low cost hardware certainly limits the system's performance but proves that performance gain can be achieved by utilizing better hardware.

The previously mentioned works either use OOK or BFSK for modulating single color LEDs. Colored LEDs have also been studied. Color-shift keying (CSK) was outlined in [27] and refers to transmitting data imperceptibly through the variation of the color emitted by red, green, and blue light LEDs. There are several studies such as [24] and [25] that use CSK and but do not utilize CMOS cameras that exploit the rolling shutter.

3.2.2 Screen to Camera Communication

There are several prior works that utilize displays for delivering information bits to cameras [16, 18, 33]. These studies focus mostly on transmitting "static" information, similar to barcodes or QRCodes, rather than building communication links as done in LED to camera works. Nevertheless, these systems also face the same receiver limitations, since they are using smartphone cameras.

VRCodes [33], is the study that influenced display to camera communication. The authors present a novel active visual tag, which utilizes all

dimensions of color, time and space and show how digital information can be embedded without being obstructive to human eyes. In their setup, they utilize ordinary displays and rolling-shutter cameras similarly to our LED to camera communication systems. However, they do not address practical challenges, such as the signal losses that were explained in the previous section with respect to synchronization. In COBRA [16], the authors propose information encoding into specially designed 2D color barcodes. They use color information of pixels to improve SNR and stream them between small-size screen and low-speed camera, where most or all the data flows along a particular direction. As most of the prior works, they also do not tackle information losses but simply repeat each packet twice. Finally, in [18], the authors present Strata, a layered coding scheme for visual communication that extends the known QR codes. The study is influenced by the hierarchical modulation from traditional RF communication, resulting in information being organized in multiple layers into the same code area. The importance of the layered coding schemes is highlighted by the heterogeneity in the smartphone cameras, that cause different amount of information to be extracted with respect to distance or channel conditions.

3.2.3 Indoor Localization

Indoor localization is a field where camera-based communication can be widely applied due to the popularity of smartphones and LED lighting. However, there are still very limited real applications utilizing VLC localization. To the best of our knowledge, only ByteLight [2] offers shop location-based services, using LED lighting along with sub-meter accuracy and sub-second latency. Nevertheless, there are several research projects [21, 23, 26, 34] that present indoor localization systems.

In Luxapose [21], the authors present a system that uses slightly-modified commercial LED luminaries (light beacons), along with smartphone cameras to provide accurate positioning compared to other RF-based approaches. The authors present a new localization approach based on the promising "angle of arrival" method. This method utilizes the projections of multiple light sources (with known positions) in the camera sensor, to estimate the phone's position. The aim of the authors is to achieve sub-meter accuracy compared to other similar pre-existing systems. Indoor localization systems do not need to deal with synchronization and reliability problems as previously discussed in RollingLight, since the light sources continuously re-transmit their location ID. Furthermore, the authors assume that the system operates in fixed distances between the light sources and the smartphone device.

The authors conducted several experiments with respect to the camera parameters. In their analysis they prove that, in order to improve the SNR and boost the contrast ratio between the bright and dark bands, the para-

meters that control the speed of the rolling shutter should be minimized (i.e. exposure time). They further propose a complete image processing pipeline for transmitter detection. The author's utilize the transmitter detection for identifying the location of the transmitters in the frame, decode the transmitted light source ID and estimate position of the receiver.

In DynaLight we:

- Base the transmitter detection on the same pipeline, but we further improve the amount of information that we can fit in the detected blob by 40%.
- Implement a different decoding pipeline suitable for the used modulation technique.

Furthermore, in Luxapose, the image processing and decoding is performed entirely in a cloud server, which offloads the processing from the phone, but it adds additional communication overhead, with respect to uploading high resolution images in a server. Last but not least, the authors exploit a smartphone (Nokia Lumia 1020) that offers huge resolution (7712×5360) and extensive camera parameter control compared to Android phones. As a consequence, the authors do not face effects caused by low cost cameras and the limited camera control.

In DynaLight we:

- We perform the image processing locally in the smartphone.
- Do not use high-end smartphones that increases the range of market smartphones that it can support.

In [26], the authors propose a non line of sight localization system. The authors present a localization system but also overcome the unsynchronization problems in a different way than RollingLight. The authors use binary frequency shift keying and they achieve 1.25 bytes per second exploiting 720p images at 30 fps and custom made transmitters. A sliding window approach is presented after capturing and concatenating all the captured frames. They also identify the discontinuities in frame receptions as discussed in RollingLight, but use two Hanning windows to smooth the discontinuities in the captured frames. We believe that this approach is impractical with respect to real-time decoding since all the frames have to be first captured before the beginning of the decoding. In addition, they do not mention whether their approach overcome the frame rate diversity between different phones. On the other hand, the authors introduce an algorithm, that is partly used by DynaLight, for overcoming the limited camera control that is introduced by the Android operating system. Last but not least, they study the impact of the background noise and also how camera focus affects decoding. Our

system is also evaluated in different ambient light conditions and we prove that is robust to noisy environments.

Other interesting projects include Epsilon [23] and PIXEL [34]. In the former, the authors exploit LED lamps, as anchors, along with a custom made light sensor that is connected to the phones audio jack. In the latter, the authors present light-weight visible light positioning solution, that is designed to accommodate resource constrained mobile devices, such as smart glasses. Moreover, the authors present a color-based modulation scheme, that handles users mobility and a fast positioning algorithm that can execute in real-time.

3.2.4 Summary

In this section, we summarize the most important camera-based projects that influenced this thesis. Table 3.1 summarizes the key camera-based works that we based our system on. It can be observed that all the projects utilized different platforms, that offer different flexibility. Based on DynaLight's requirements that were presented in Chapter 2, we present the related features of our project. It has to be mentioned that only DynaLight has a dynamic data rate due to its adaptive behavior. All the design choices will be thoroughly explained in the following chapters.

Table 3.1: **An overview of most important projects discussed in this chapter.** An asterisk ($\star$) means the content of the cell is unknown or uncertain.

Project	Platform/ Exposure Control	Resolution	Modulation	Coverage	Data Rate
CMOS for VLC [11]	Android/ Low	480p	OOK + Manchester Encoding	8-9cm	1-3 kbps
Visual Light Landmarks [26]	iOS/ Low	720p	BFSK	1m	1.25 bytes/sec
Luxapose [21]	Windows/ High	5360p	PWM + Manchester Encoding	2-3m	*
RollingLight [22]	iOS/ High	1080p	FSK	160cm	11.32 bytes/sec
DynaLight	Android/ Low	1080p	OOK + Manchester Encoding	20-120cm	Dynamic *

Chapter 4

Overview

This chapter gives a short overview of the developed system. In Section 4.1, we present the main components of our system, while in Section 4.2 we briefly discuss the system's fundamental functionalities.

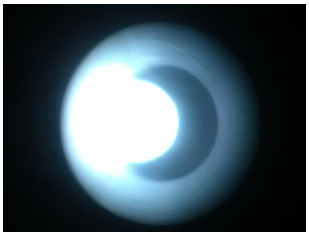
4.1 System Components

DynaLight is a VLC system designed for data communication. Therefore, it consists of a transmitter and a receiver facing one another. Both of them operate together so as to create an adaptive and reliable, visible light communication link.

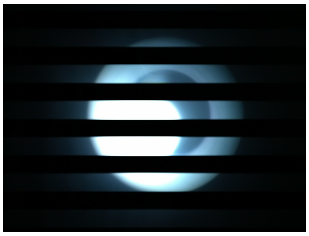
The transmitter consists of a microcontroller and a LED. The microcontroller is used to modulate the LED and encode information. Information is encoded in such a way, that the average emitted light power is constant and does not create noticeable flickering. More specifically, the LED is turned on and off at a rate that is high enough for the human eye to notice.

On the other hand, the receiver's side consists of the back-facing camera of a smartphone. The phone takes advantage of the rolling shutter capturing mechanism that was explained in Section 2.2. In order to be able to detect the transmitter light signal, we need to tune the camera sensor so that we can observe the white and dark distortions. As has been mentioned in the previous sections, by lowering the exposure time we are able to control the rolling shutter speed. Having a fast rolling shutter enables the camera to identify the white and dark bands that follow the lighting pattern of the transmitter, as can be seen in Figure 4.1. By continuously capturing images and applying image processing techniques, we are able, first to identify the light source and then decode the transmitted lighting patterns.

DynaLight has to be flexible with respect to distance and ensure reliable transmissions. Having a single transmitter and receiver facing one another is impractical due the fact that the transmitter side is not able to measure their distance and adjust the transmitter amount of information accordingly.



(a) Auto Exposure



(b) Low Exposure

(c) Encoded Signal

Figure 4.1: **Camera exposure settings**. In both pictures the LED is modulating at a frequency of 200 Hz. By lowering the exposure time we are able to observe white and dark bands (distortions), that correspond to the transmitted signal. Dark and white bands correspond to 0 and 1 bit respectively.

Additionally, it is impossible to automatically synchronize the two components because they are not connected. In DynaLight, both sides are equipped with both a transmitter and a receiver as depicted in Figure 4.2 and 4.3. With the proposed setup both sides are able to estimate their distance and maintain synchronization. In the following section, we briefly present the main functionalities of our system but the transmitter's and receiver's side will be thoroughly explained in Chapters 5 and 6 respectively.

4.2 Basic Functionalities

In this section we present the main operations of each system component. The basic functionalities of the transmitter refer to encoding, packet construction and modulation. On the receiver side, we perform the transmitter detection, distance estimation, image processing and decoding. The two transmitters and two receivers facilitate in calculating of the ideal channel capacity for each distance and for maintaining synchronization.

Encoding/Modulation: As it has been mentioned above, the transmitter consists of a microcontroller that is used for creating the lighting patterns. The data bits are first encoded then organized into packets and then transmitted using high rate light modulations. In order to create a flicker free system OOK plus Manchester encoding was utilized as will be further explained in Chapter 5.

Transmitter Detection: The transmitter detection regards identifying which areas of the captured frame contain decodable information. In DynaLight, we base the transmitter detection mechanism on the one that was

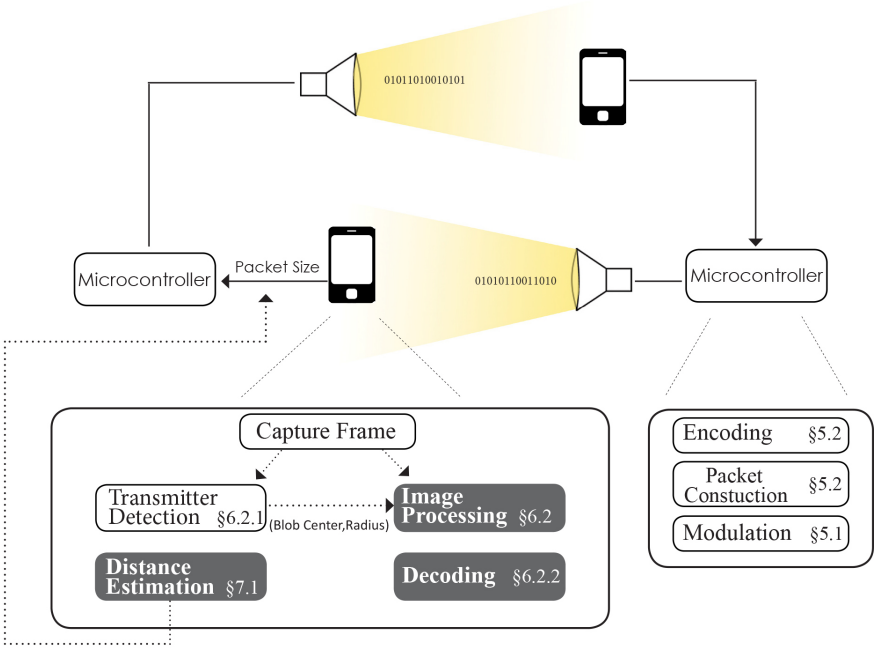


Figure 4.2: **DynaLight's overview.** The system consists of two transmitters and receivers that communicate via "invisible" light modulation. The grey boxes highlight the sections where the main contributions of this thesis can be found.

presented in Luxapose [21]. We further enhance the proposed mechanism by adding an additional image processing step that increases the detected area by 40%, meaning that we are able to decode more data. More details about our enhancement are provided in Section 6.2.1.

Distance Estimation: Since we are creating a line-of-sight visible light communication link, that could operate in multiple distances, the receiver should be able to measure the distance between the two main system components in order to adjust its channel's capacity accordingly. The distance estimation is based on the detected transmitter size of the previously mentioned operation. The bigger the detected transmitter the closer the two components are and vise versa. The system's design enable us to perform the distance estimation in both sides. More details will be presented in Section 7.1.

Image Processing/Decoding: This operation refers to all the image processing techniques, that DynaLight exploits, so as to decode the transmitted information. It has to be clarified that we did not base theses operations on Luxapose [21], as done with transmitter detection. More in-

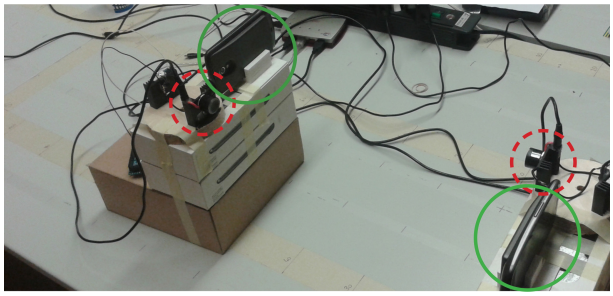


Figure 4.3: **DynaLight's setup.** The red circles show the transmitters while the green ones show the receivers.

formation about the challenges that we encountered can be found in Section 6.2.

Channel Capacity Calculation: The amount of information that the transmitter sends is related not only to the distance of the two components but also on the fact that we wish to avoid having the mixed symbol/packet problem that was mentioned in Section 3.2.1. Once the distance is estimated the receiver sets the appropriate amount of information to be sent to its connected transmitter, as shown in Figure 4.2 (arrow from phone to microcontoller). More information on the channel capacity calculation can be found in Section 7.1.2

Synchronization: As in all camera-based VLC systems, synchronization refers to the problem of missed data due to fluctuation reception rate that was discussed in Section 3.2.1. The system tries to minimize the synchronization problem in two ways. First, we synchronize the the start of transmissions and then we try to maintain a constant reception rate. More information about this problem can be found in Section 7.2.2.

Chapter 5

Transmitter

This chapter describes the first component of the developed system which is the transmitter (Tx). In Section 5.1, the selected modulation technique is presented, while Section 5.2, discusses the packet structure that is used for transmitting data and how data is encoded and transmitted. Section 5.3 presents the implementation details with respect to the selected hardware.

5.1 Modulation Technique

In order to encode data, the developed system modulates signals on the LED transmitter in such a way that data can be accurately decoded by the receiver, as well as not generate direct or indirect flicker (stroboscopic effect). As it has already been discussed in Chapter 3, recent research has shown that different modulation techniques have already been experimented such as, OOK, PWM or BFSK.

In this work, we chose to exploit On-off keying (OOK) along with Manchester encoding. OOK denotes the simplest form of amplitude-shift keying (ASK) modulation in which, digital data “0” and “1” are represented with turning the LED on or off respectively. However, utilizing just OOK introduces flickering when it comes to representing long sequences of “0” or “1”. By combining OOK with Manchester encoding, each “0” and “1” bit is represented by a sequence of “01” and “10” symbols accordingly. As a consequence, it is impossible to have more than two consecutive matching symbols, which eliminates the OOK flickering. Moreover, the number of 0s and 1s is the same regardless the encoded data. Figure 5.1 depicts the difference between plain OOK and combining it with Manchester encoding. In the first case, the long sequence of “0” bit will create noticeable flickering if the modulation frequency is not high enough, while in the second case the symbol proportion is more balanced.

OOK, plus Manchester encoding, is appealing for its simplicity. Nevertheless, there are cases where other transmitters can lower its robustness.

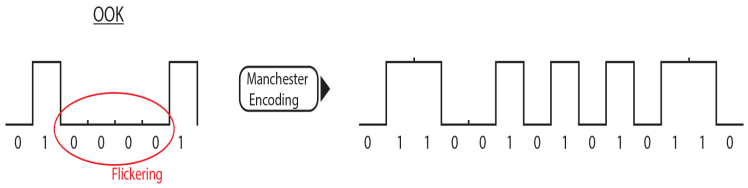


Figure 5.1: **OOK vs Manchester encoding.** Having consecutive 0s leads to flickering which is eliminated by Manchester Encoding.

As a result, it is assumed that the system operates without other interfering transmitters.

5.2 Packet Structure and Encoding

In the proposed communication protocol, symbols are organized into packets consisting of a fixed preamble and a varying payload. We set the preamble size to be 5 symbols "01110", and by having three consecutive "1" we make sure that it is distinguishable from any other symbol combination.

The size of each packet is determined by the fixed preamble size and the number of symbols in the payload. Furthermore, the duration of each packet is given by Equation 5.1, where T_{symbol} is the period of each symbol and $n_{symbols}$ is the number of symbols in the payload.

$$T_{packet} = 5 \cdot T_{symbol} + n_{symbols} \cdot T_{symbol} \tag{5.1}$$

The payload can be of different sizes based on the distance between the transmitter and the receiver, as we will explain in Chapter 7. As a result, the preamble overhead varies. The more symbols in the payload, the less overhead is introduced by the preamble. For example, having 4-symbol payloads means that we have an overhead of 62%, while having a payload of 16 symbols results in 15% overhead.

Our goal is to build a system that can transmit any bit sequence but more specifically ASCII codes. Every ASCII code consists of 8 bits. Each ASCII code is firstly transformed in to its binary representation and then to the correspondent Manchester encoding symbol sequence. Then, depending on the channel's capacity, we split and construct the equivalent amount of packets. Each packet is transmitted for a specific amount of time with respect to the receiver's reception rate, as it will be discussed in Chapter 7.

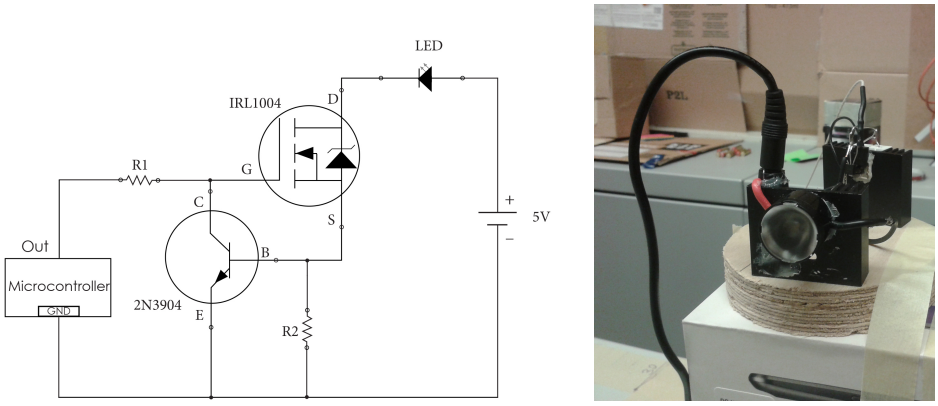
5.3 Implementation Details

The transmitters of our system consist of a microcontroller, the LED and a reflector. We chose not to use the smartphone's LED due to the limitations

of the Android operating system. More information can be found in Sections A.1 and A.2 about this decision.

The LED modulation, control and data encoding is performed completely on the microcontroller, which in our case is an Arduino MEGA ADK [1]. In order to achieve the required modulation frequencies and accuracy, we used the microntroller's timer interrupts. More specifically, we turn on and off the LED based on the timer ticks using an interrupt handler. Arduino has a system clock of 16MHz and the timer clock frequency is set by the prescale factor. The desired output frequency for the LED is around 3 kHz so we choose a prescaler value of 256. More information about the selected frequency can be found in Chapter 6.

For the LED we use the XLamp MC-E White by Cree [10]. It is a multi-chip LED, that provides high lumen output in a small package. It consists of four individually addressable LED dies (integrated circuit). The maximum drive current per LED die is 700mA which results in 751 lm at 9.5 W. The size of the whole package is 7×9 mm and the viewing angle is $110\,^{\circ}$. The implemented transmitter circuit can be found in Figure 5.2(a). In order to drive all four LED dies we used a constant current driver that outputs a current of 1.5 A. The circuit uses a MOSFETs transistor IRL1004 [7] and a NPN Switching transistor 2N3904 [8]. The circuit acts as a current feedback loop which continuously monitors the LED current and keeps it exactly at the set point at all times. The resistor values for R1 and R2 are 100 Ω and 0.34 Ω accordingly. R2 also has a power rating value of 5 Watts. The output power of the LED is 5.4 Watts.



(a) Driving Circuit (b) Transmitter

Figure 5.2: **Transmitter setup**. The circuit outputs a current of 1.5 A to the LED. In the right image we can also observe the diffuser lens.

Apart from the microcontroller and the actual LED, a lens was used in order to reduce the viewing angle of the LED. Different lens were examined

in order to decide on the most effective one. We decided to use the diffuser [3], which offers the brightest beam and a 20 ° viewing angle. The size of the lens is 2 × 2cm. The advantages and effects of the diffuser can be found in Chapter 8.

An additional reason for selecting the aforementioned microcontroller is, the offered flexibility when it comes to connecting it with an Android phone. It is equipped with a USB host interface, given by MAX3421E IC [6], which allows the Arduino to connect and interact with any type of device, that has a USB port. We use this interface for sending control commands to the microcontroller, as well as the data to be transmitted, which in our case are the ASCII codes.

Chapter 6

Receiver

This chapter describes the second fundamental component of the developed system which is the receiver (Rx). In Section 6.1, all the important camera parameters are discussed along with their importance in receiving the transmitted signal. Then, in Section 6.2, a detailed description of the image processing pipeline is presented, followed by Section 6.3 which includes important implementation details and limitations.

6.1 Camera Control

As it has been mentioned several times in the previous sections, smartphones use the rolling shutter capturing mechanism in which, the frames are captured in a row sequential way, as shown in Figure 2.2. In order to get the depicted bands in the captured frame, there are several camera parameters that have to be tuned appropriately.

Exposure time & ISO: The most important parameter is the *exposure time*. Exposure time, determines the duration that each pixel collects photons. In other words, exposure time affects the amount of light in an image. In practice, short exposure values increase the ability to distinguish between the dark and white bands. A closely related parameter is *film speed (ISO setting)*, which affects the amount of photons that are required to saturate a pixel. In [21], the authors argue that for the best performance and in order to improve the SNR, the exposure time and film speed have to be minimized.

Scan rate & Resolution: As has been mentioned above, the frames are captured in a row sequential way, so we can consider each individual frame as our signal, and the pixel-rows can be regarded as the samples. The number of rows in a frame is determined by the camera resolution. As a result, the time to scan a new frame refers to the time that the sensor requires to scan

all the rows of the frame. This property is named *camera scan rate* and is the rolling shutter speed. The camera scan rate can also be considered as the sampling rate of our pixels (samples). Certainly, the scan rate depends also on the time that is needed to scan a single row.

Each "pixel-row" is first exposed and then read. Due to the rolling shutter effect, as shown in Figure 2.1, the time to scan a single frame is given from Equation 6.1, where T_e is the exposure time and T_r is the time to read a single row.

$$T_{frame} = T_e + T_r \cdot Rows \tag{6.1}$$

All the previously mentioned parameters affect the width of each captured band. The width of each band is expressed in pixels and represents how long the light was on or off in the transmitter and as a result it affects the packet decoding. In [22], it is proved that the width W of each band can be given by Equation 6.2 where f is the modulation frequency of the transmitter.

$$W = \frac{1}{2fT_r} \tag{6.2}$$

Combining Equations 6.1 and 6.2, we present Equation 6.3, where we relate width of each band with the most important camera parameters, meaning the exposure time, the scan rate and the number of rows (resolution).

$$W = \frac{1}{2f\frac{T_{frame}-T_e}{Rows}} \tag{6.3}$$

As it can be observed in Equation 6.3, the width of each band *does not depend on the orientation of the camera, the size of the LED or the distance between the LED and the smartphone.*

Frames per second: Finally, the number of frames that a camera can obtain per second is significantly important, when it comes to specifying the VLC link's data rate. Nevertheless, the actual bandwidth is also determined by the ability to synchronize the two components. This challenge will be further explained in Chapter 7.

Other Camera Functions

Cameras usually have additional functions that affect the captured images such as antibanding, video stabilization, white balance and auto-focus. Antibanding is a function that decreases fluctuations in brightness of frames or images, caused by a light source oscillations. Video stabilization is a family of techniques utilized to reduce blurring caused by motion of a camera. Furthermore, white balance refers to methods that correct the colour balance of

the lighting in an image. Finally, auto-focus concerns methods for adjusting the camera lens so as to obtain focus on a subject.

Challenges

In practice, smartphones are limited by the operating system and do not allow applications to modify some of the previously mentioned parameters. The scan rate and the time to scan a single row are *fixed* parameters of the camera sensor and cannot be tuned. On the other hand, exposure time, film speed (ISO), resolution and the frames per second can be set by a developed application. However, the value range is very limited and varies from phone to phone. As a result, Android phones do not offer much flexibility compared to iOS or Windows phones. Moreover, the exposure range may also be limited by the camera sensor due to the limited hardware capabilities. More information about the selected values for each parameter can be found in Section 6.3

6.2 Image Processing

Once the camera sensor has been tuned appropriately (low exposure time), and has obtained a frame with recognisable white and dark bands, the following steps regard, the image processing methods that are required, in order to convert bands to useful data bits. DynaLight's image processing pipeline exploits methods that are offered by the OpenCV image processing library.

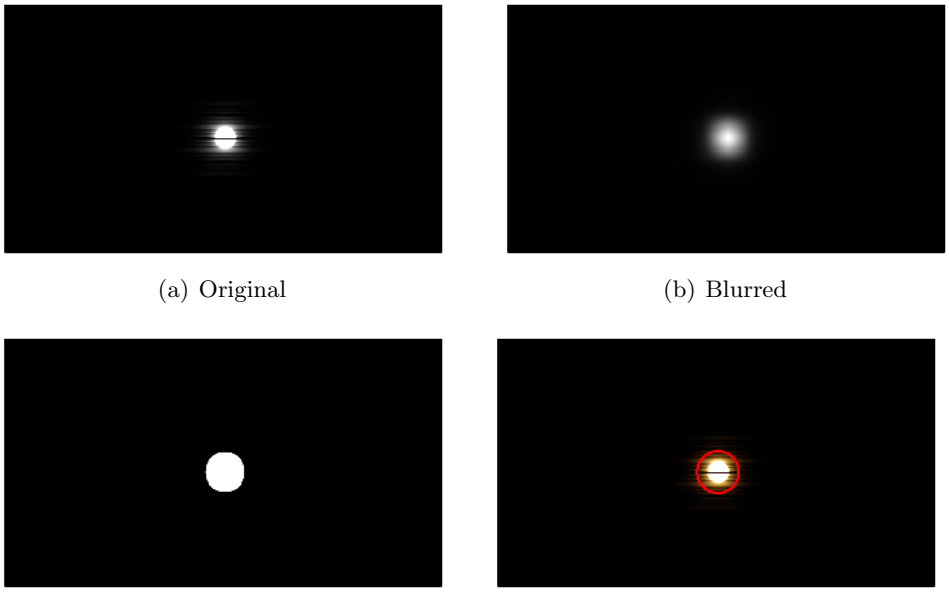
Before discussing the decoding method we first need to highlight the importance of the transmitter's detection. DynaLight is a LoS system and is designed to operate in close distances from 20 cm to 120 cm. As a result, the transmitter light will not be spread out in the whole frame as can be observed from Figure 6.2(a). Furthermore, we assume that the transmitter does not have to be always in the center of the frame, as in the previously mentioned figure. This fact gives prominence to the need of a transmitter detection mechanism, that will identify, which part of the image contains decodable information.

6.2.1 Transmitter Detection

As has been mentioned above, by selecting the minimum exposure settings, the image will be brighter in the area of the transmitter plus, creating a visible "aura" around it with respect to its intensity. The greater that area is, the more bands are visible, which results in the extraction of further useful information. We base transmitter detection to the one that was presented in Luxapose [21]. We further improve the proposed transmitter detection

method so as to take advantage of as many bands as possible.

Existing Method: We use the same transmitter detection steps that were presented in Lucapose [21]. The basic sub-steps are: 1) blurring the image using a 100×100 kernel, 2) pass it through a binary OTSU filter, 3) find the contour of each blob and 4) find the minimum enclosing circle for each contour. In the first step a big kernel is utilized so as to create a "shadowy" image. The next steps concern detecting the blob area where the transmitter light is diffused, meaning the center and the radius of the transmitter blob. The different steps can be found in Figure 6.1. After experimenting with the proposed transmitter detection method, we noticed the method underestimates the detected transmitter area. This means that there are more bands that remain "hidden" due to the low exposure settings. As a result, we looked for a method that could highlight all the information that is hidden in the frame.



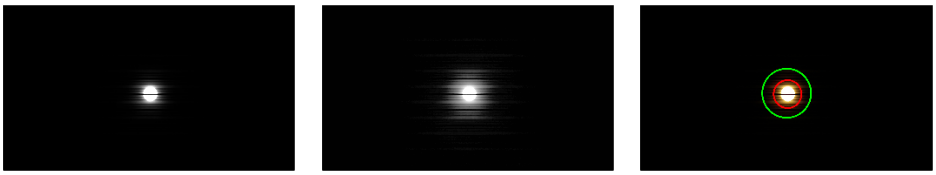
(a) Original (b) Blurred

(c) OTSU Filter (d) Min Enclosing Circle

Figure 6.1: **Transmitter detection pipeline of Luxapose [21].** The image is first blurred (b) and then passed by the OTSU filter (c) that creates a binary image. Finally, the minimum enclosing circle (red circle) is calculated, which represents the detected transmitter (d).

Contrast Increase: To increase the visible bands, we add a contrast increase step before the existing ones. The image processing method is called *histogram equalization* (HE), and it is used for contrast adjustment using

the image's histogram. Histogram equalization is mainly useful for increasing the global contrast of an image. Contrary to that, adaptive histogram equalization (AHE) is used increasing the local contrast of an image but it over-amplifies noise. In our image processing pipeline, Contrast Limited AHE (CLAHE) is utilized, which is used for global HE but also limits the noise of over-amplification. The effect of contrast increase can be observed in Figure 6.2(b). After increasing the image's contrast, the rest of the transmitter detection steps. The difference in the detected blob size can be seen in Figure 6.2(c). We compare our blob detection pipeline with Luxapose in Section 8.1.



(a) Original frame (b) After Constrast Increase (c) Detected blob with (green) and without (red) contrast increase

Figure 6.2: **Impact of contrast increase.** After increasing the contrast of the image we can observe that more bands can be identified (b). In the right image (c) the red circle represents the detected blob of Luxapose [21]. The green circle shows the improvement of the contrast increase.

6.2.2 Data Decoding

Once the transmitter is located in the image, we continue by examining only the corresponding blob sub-region independently to decode data. In Luxapose and RollingLight, different modulation techniques were exploited meaning that we had to implement our own decoding pipeline. Furthermore, by utilizing different hardware, we were able to identify and overcome additional limitation that the other works did not encountered. It has to be clarified that we do not crop the transmitter area before processing it.

The complete decoding pipeline is depicted in Figure 6.5. It has to be clarified that we take advantage of the same contrast increase method also in the decoding pipeline. The additional steps are presented bellow.

Blurring: We observed that the contrast increase introduces some noise that can be smoothed exploiting the OpenCV blurring method. This blurring step is different from the aforementioned in the transmitter detection. In this phase a very small kernel (3×3) is used, for averaging the pixel values

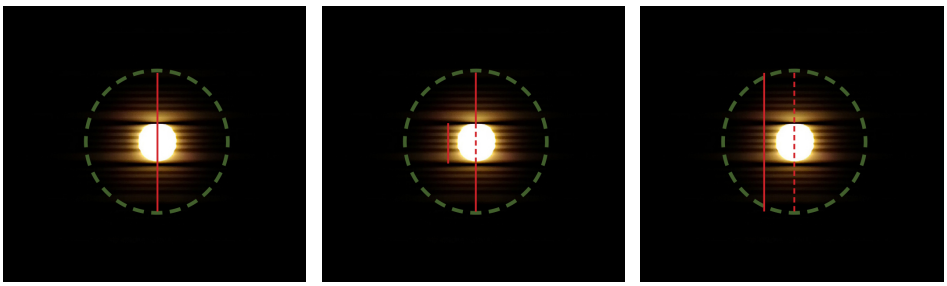
and reduce noise.

Thresholding: The current step concerns the transformation of the image to a binary image. Thresholding, is the simplest method of image segmentation. The most trivial way would be to select a global thresholding value in order to distinguish between the white and dark bands. However, the light intensity of the image is not normalized, meaning that different areas in the image, have different intensities (Figure 6.3), which does not facilitate in finding a single threshold value. OpenCV, offers a method for adaptive thresholding in which the algorithm calculates the threshold for a small regions of the image. As a consequence, we get different thresholds for different regions of the same image, which result in better results for images with varying illumination. The result of this method is a binary image, where dark pixels have a value of 0 and white a value of 255. The result of this method can be observed in Figure 6.5(c). The red circle shows the detected transmitter blob.

Avoid Over-exposure: As discussed in Section 6.1, due to the limited exposure time control, we observe the over-exposure effect in which, white bands appear wider and overlap other bands. This effect is found mostly in the center area of the detected blob where the light has the maximum intensity. We can observe this phenomenon in Figure 6.3 and compare it with Figure 4.1, where white blob is absent. This phenomenon causes problems when it comes to selecting which pixel values we should decode, from the detected blob area that was mentioned in the previous section.

Figure 6.3: **Over-exposure effect.** The white bands in the center of the light blob overlap due to limited exposure settings.

In Luxapose, the authors did not face over-exposure effects, so they used the middle column of pixels (vertical radius of the detected transmitter blob) for decoding information, as shown in Figure 6.4(a). The most easy solution would be to use a fixed column offset to avoid the big white blob in the previously mentioned figure. However, having a predefined offset is impractical in different distances where the blob size is varying. This fact



(a) Middle Column (b) Middle Column and Offset (c) Offset

Figure 6.4: **Pixel selection method (over-exposure avoidance).** The first method (a) uses the pixels of the middle column as Luxapose [21]. The other two methods use an offset in different ways, so as to avoid the white over-exposure effect. The red vertical continuous line represents the pixel column that each method uses for decoding.

highlights the need of an *automatic* mechanism that avoids the over-exposure effect and selectes the pixels to be decoded. During the experiments, we tried two different avoiding methods.

Recall that the expected symbol's width can be calculated from Equation 6.3. We present a few width examples in Section 6.3. The wider expected band is the preamble, which consists of three "1" symbols as explained in Section 5.2. As a result, the maximum width of a white band (preamble) follows Equation 6.4, where W is given by Equation 6.3.

$$W_{white} \leq 3 \cdot W \tag{6.4}$$

In the first method, we begin by examining the pixel values of the middle column. As we move closer to the center we detect big white bands that are bigger than the expected preamble, corresponding to the over-exposure white blob. By comparing each band width with the expected one we can identify the length of the over-exposed blob in pixels. The method selects the pixel values that are in the middle column, but uses a column offset only for the rows where the over-exposed blob was found as depicted in Figure 6.4(b). The offset value is equal to the size of the detected over-exposure white blob. *With this method we avoid the over-exposure effect but also use the brightest pixel values (of the middle column).*

The second method, uses a constant column offset as depicted in Figure 6.4(c). The offset is calculated as follows. We begin by examining the middle blob column. If there is a band that is wider than the expected, we continue by examining the previous column (one to the left). If there are no wider bands in the examined column we select all its pixels, as shown in previously mentioned figure. *With this method we select the closest column*

to the center of the blob radius that does not contain over-exposed bands.

We evaluate the effect of each method in Section 8.2.1. The result of this step is an array of pixel values that need to be decoded as shown in Figure 6.5(d).

Decode: The final step regards decoding the array of pixel values as shown in Figure 6.6. The decoding is based on the packet structure that was described in Section 5.2.

Initially, we look for the preamble which is used to separate the different packets. As has been mentioned above, we can identify the preambles by checking the width of each band. Every band whose width is more than two times the expected symbol width, is considered a preamble as shown in Equation 6.5.

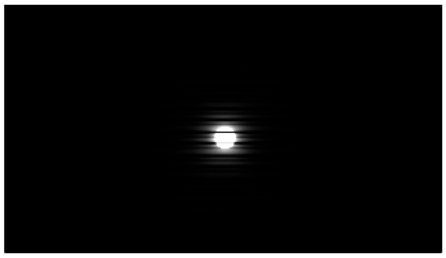
$$W_{preamble} > 2 \cdot W \tag{6.5}$$

We need to identify at least two preambles so that we can be sure that a whole packet can be decoded. This issue will be further explained in Chapter 7. Once the preambles are found, we examine the number of pixels of each dark or white band in order to get the encoded symbol. The symbol combinations can be found in Equation 6.6, and follow the Manchester encoding.

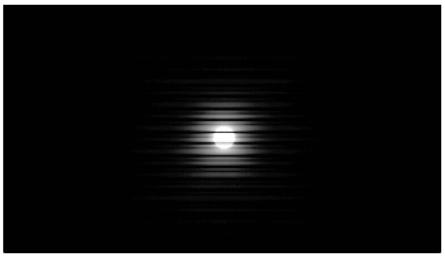
$$symbol = \begin{cases} 0, & \text{if } W_{dark} < 1.3 \cdot W \\ 1, & \text{if } W_{white} < 1.5 \cdot W \\ 00, & \text{if } W_{dark} > 1.3 \cdot W \\ 11, & \text{if } W_{white} > 1.5 \cdot W \end{cases} \tag{6.6}$$

It has to be mentioned that, due to over-exposure effect the dark bands tend to become shorter and that is why in the conditions of the dark bands we used a smaller multiplier.

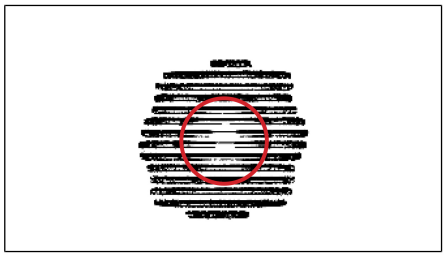
The result of the decoding is a binary array with the result symbols that represent the Manchester encoding of our transmitted data, as shown in Figure 6.6.



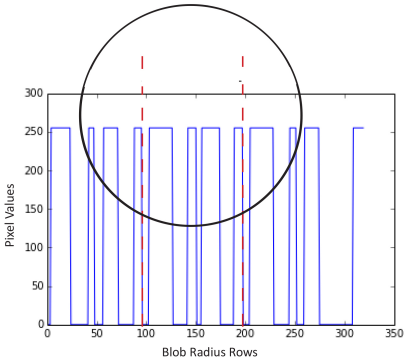
(a) Original



(b) Contrast increase + Blurred



(c) Adaptive Threshold



(d) Pixel values within detected blob

Figure 6.5: **Image processing pipeline.** First we increase the contrast of the original image and then we blur it (b). We create a binary image by using a adaptive threshold (c). Finally, we select the pixels to be decoded based on the detected blob radius (red circle in (c)) and the pixel selection method. The black "circle" is (d) is enlarged in Figure 6.6

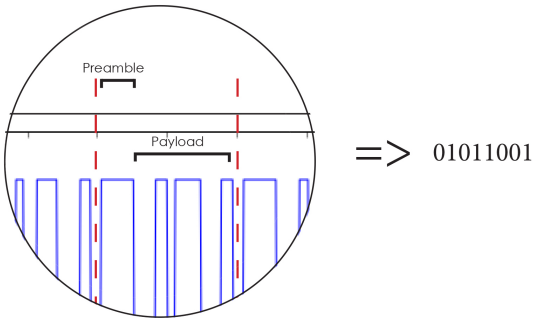


Figure 6.6: **Focus and decoding of Figure 6.5(d).** We process the pixel values to identify the preamble and payload so as to decode the data sequence.

6.3 Implementation Details

In our work the Samsung Galaxy Nexus was utilized as a receiver. The phone provides a 5 MP, as well as a 1.3 MP front-facing camera. DynaLight uses the back-facing camera which can capture full high-definition video of 1080p with maximum of 24fps. In terms of processing power, it is equipped with a Dual-core 1.2 GHz Cortex-A9 processor. As for the operating system, we installed the Cyanogenmod 11 which is based on Android 4.4 KitKat.

6.3.1 Parameter Values

In the current section we present how did we calculate or set the camera's parameters. In order to calculate the expected width of each band we need to know the exposure time, the camera scan rate and the camera resolution (Equation 6.3). In Table 6.1, we summarize the available camera parameters for the used phone.

Table 6.1: **Camera parameters of Samsung Galaxy Nexus.**

Parameter	**Value**
Scan Rate	46956 rows/sec
Resolution	1920×1080
Exposure Time	1/4000s

As has been mentioned above the camera scan rate is a fixed parameter of the camera sensor. In order to measure the camera's scan rate we set the transmitter at a fixed distance transmitting a square wave at a known frequency. Then we count the number of bands on the captured frame. More specifically, at resolution of 1920x×1080, when the transmitter is modulating at frequency of 200Hz we observe 9 white and dark bands. By multiplying the observed number of bands with the period of each on and off pulse (2.5ms), we conclude that the camera needs around 23ms to scan 1080 rows. As a result, the scan rate is 46956 rows per second.

As has been mentioned in Section 6.1, the Android operating system is not very flexible when it comes to setting the exposure time. More specifically, it does not provide an API so that the user can set the exposure time directly. However, it provides methods for locking the exposure time and setting the exposure compensation. The exposure compensation modifies the exposure time or film speed without the user knowing the exact values. In order to overcome this limitation, we first set the exposure compensation to the minimum available value. Then, by pointing a bright source close to the camera we force the exposure time to reach the lowest value. Finally, we lock the exposure time exploiting the provided method by Android camera API. To the best of our knowledge, only the very recent versions of the Android (>Android 5) provide a full camera API with methods that enable

application to set the exposure time directly. We found out that the lowest exposure time value for the used phone is approximately 1/4000s. Apart from the exposure time the current phone does not allow us to modify the film speed (ISO), meaning that we cannot know its value.

Last but not least, we disable a few parameters that we believe that could affect our decoding. These are, the antibanding, video stabilization, white balance and autofocus.

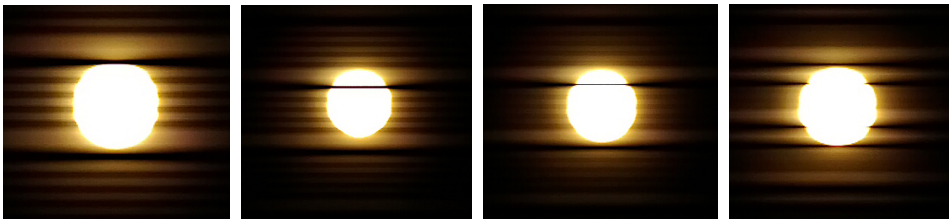
6.3.2 Maximum Modulation Frequency

In order to maximize the amount of information that we can send from the transmitter side we first need to know what is the maximum modulation frequency that our receiver can decode. Assuming that the width of each band is one pixel and based on the Nyquist frequency, we get the theoretical upper bound, which is half the camera's scan rate. However, having one pixel long bands is impractical when it comes to decoding the images. The bands should be a few pixels long to ensure that they can be correctly decoded. We experimented with several modulation frequencies but in the current section we will present the effects of the modulation frequencies of 3000Hz, 3250Hz, 3500Hz and 4000Hz. We captured a single frame per frequency and we observed the formation of the bands in the images.

As it can be seen from Figures 6.7, as the modulation frequency increases above 3250Hz we are not able to differentiate between the white and the dark bands. This is mainly caused by the camera's scanning rate. Following Equation 6.3, we can calculate that for the frequency of 4000Hz we get 6 pixels per band, which makes the bands become invisible. On the other hand, when we modulate at a frequency of 3000Hz we get 8 pixels per band, which makes the bands visible enough to be decoded. However, when we have around 7 pixels per band (3250Hz and 3500Hz), then the bands start to become blur as can be seen from Figure 6.7.

We conclude that we need more than 7 pixels so that each band can be separated. In our system we prefer to exploit a modulation technique that will facilitate our implemented image processing algorithms. By having a modulation frequency of 3000 Hz (8 pixels) fulfils the requirements of not causing flickering, as well as, having distinguishable bands. It can be also calculated that the biggest expected band (preamble) will have a width of 24 pixels.

It has to be mentioned that the number of pixels per bands are not affected by the distance between the transmitter and the receiver, as well as, the size or intensity of the transmitter.



(a) 3000Hz=8 pixels (b) 3250Hz=7 pixels (c) 3500Hz=6.8 pixels (d) 4000Hz=6 pixels

Figure 6.7: **Frequency and pixels per band.** As the frequency increases the white and dark bands get mixed up and cannot be separated. The short bands are the symbol bands, while the bigger are the preamble bands. The width of each band is independent of the distance between the transmitter and the receiver.

Chapter 7

Dynamic Link

This chapter discusses the adaptive behavior of DynaLight. In Section 7.1, we present our distance estimation method and how the channel capacity calculated in accordance with the detected distance. In Section 7.2, we discuss how the implemented smartphone application operates, along with the limitations that we encountered with respect to synchronizing transmissions.

7.1 Dynamic Channel Capacity

7.1.1 Distance and Blob Size

As it has been mentioned in the previous chapters, the aim of this project is to create a VLC system that can maximize its channel capacity based on the distance between the transmitter and the receiver. First, we need to identify what is the relation between the distance and the channel capacity. It can be seen in Figure 7.1 that the closer the receiver to light source the greater the visible light area ("aura") that the light creates. As a result, after applying the blob detection method, the blob in closer distances will be noticeably larger, meaning that we can fit more data, therefore have a greater amount of information per frame. This fact is significantly important and defines the link's channel capacity.

We relate the distance of the transmitter and receiver to the detected blob size. As the distance increases, between the transmitter and the receiver, the received energy at each pixel drops due to the line of sight path loss. As a result the projected transmitter area in the imager plane also decreases. Generally, the received light intensity of a light source follows the inverse-square law. This means that if you double the distance between the light source and the receiver, the received intensity drops to a quarter. In our case we are not measuring the light power of the transmitter, but the light "area" (in pixels) that it creates and will contain the light pattern. The measurement uses the transmitter detection method that was explained in the previous chapter. This means that we have performed the blob detection

(a) 20cm (b) 100cm

Figure 7.1: **Distance and blob size.** The bigger the light blob, the closer to the light source.

in several distances and saved their relation. As a result, when the receiver measures a blob size it can correlate it with the relevant distance. More information about the observed relation can be found in Section 8.1.

In order to maximize the amount of information in our channel we chose to apply the blob detection when the transmitter is fully on and not when is modulating. The difference is that when the light is modulated the average emitted power is reduced because the light turns on and off. However, we observed that the change in the intensity is negligible and this is mainly caused by the addition of the preamble, which introduces larger white bands. Finally, we see a significant increase in the detected blob size after applying the contrast increase that was presented in Section 6.2.1, which will be further evaluated in Section 8.1.

7.1.2 Packet Size Calculation

As it has already been discussed in Chapters 2 and 6, due to the fluctuation of the reception rate and inter-frame gap between the frames, there is a high probability that a packet can be contained in more than one frames or missed completely. As a result, the packet size is a very important characteristic of a visible light communication system. Having big packets might cause the mixed-packet problem, meaning that a complete packet might be contained in more than one frames. On the other hand, having very short packets may result in having a frame containing parts of more than one packets or underestimate the channel's capacity. Due to the discontinuity of capturing the frames it is difficult to estimate which part of the packet gets mixed up, as can be seen in Figure 3.1.

DynaLight should be flexible and adapt its channel capacity to different distances. That is why we vary the packet size based on the detected distance between the transmitter and the receiver. Recall that in DynaLight's setup, both sides have a transmitter and a receiver and can calculate their distance. First, we detect the transmitter blob size and then we calculate how many symbols we can fit in the blob radius. *In order to overcome the mixed packet*

problem, we make sure that a whole packet can fit at least one time within the detected blob sub-region. This means, that in every captured frame the receiver will be able to capture and decode a complete packet.

To do so, we base our calculations assuming that within the detected blob size we can fit at least two complete packets as shown in Equation 7.1 and Figure 7.2.

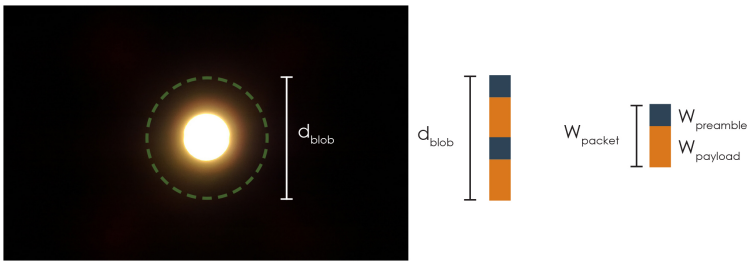


Figure 7.2: **Packet size and blob's radius.** The blob radius should fit two complete packets to avoid mixed packets.

$$d_{blob} \geq 2 \cdot W_{packet} \tag{7.1}$$

Following the analysis in Chapters 5 and 6, we can extend Equation 5.1 to get the packet size in pixels, where W_{symbol} is the number of pixels per symbol and can be calculated using Equation 6.3.

$$W_{packet} = 5 \cdot W_{symbol} + n_{symbols} \cdot W_{symbol} \tag{7.2}$$

In the proposed system the channel capacity is calculated after detecting the transmitter's blob size. By combining Equations 7.1 and 7.2, we can calculate the number of symbols we can fit in the blob radius, which is given by equation 7.3.

$$n_{symbols} \leq \frac{d_{blob} - 10 \cdot W_{symbol}}{2 \cdot W_{symbol}} \tag{7.3}$$

After calculating the number of symbols per packet, we send it to transmitter so that it can organize the packet construction accordingly.

It has to be mentioned that the result number of symbols of Equation 7.3 should always be even. Having an odd number of symbols per packet introduces flickering between different packets. For example, if we alternate between packets "01101" and "01100" then we will observe noticeable flickering due to the different number of 1s in each packet, which affects the average emitted power. By having an even number of symbols along with Manchester encoding, avoids flickering.

The above analysis ensures that, *in every captured frame and within the detected blob, we will be able to decode a complete packet* and overcome the

mixed packet problem. We believe that with our solution, having reliable transmissions is affected only by the fluctuating reception rate of modern smartphones. Once we maintain synchronization between the transmitter and the receiver, the captured frames will contain *all* the transmitted data.

7.2 Smartphone Application

One of the goals of this thesis is to implement DynaLight in a smartphone application. This means developing all the aforementioned image processing algorithms to the smartphone. Since we are building a camera-based VLC link we encountered the problems that were thoroughly explained in RollingLight [22] and refer to synchronizing transmissions. In Section 7.2.1, we present a short overview of the developed application. Then, in Section 7.2.2, we explain the main synchronization problems that we encountered while building our system.

7.2.1 Overview

In order to overcome the limitations of Java with respect to memory management and performance, we developed the processing pipeline and the decoding directly into the Android native interface as shown in Figure 7.3. Besides the Android Software Development Kit (Android SDK), which supports Java, Android provides Native Development Kit (NDK) to support native development in C/C++. It is generally believed that NDK programming is suitable for CPU-intensive applications such as game engines, signal processing, and physics simulation. Another benefit of implementing the processing pipeline in the native interface, is to take advantage of the implemented C++ OpenCV image processing methods.

The designed application is created following the producer-consumer multithread model as shown in Figure 7.3. The main thread captures the frames and stores them in a blocking queue (FIFO). Then, the processing thread processes each frame, one after the other, using the native code implementation of our image processing pipeline and decoding. Finally, it combines the decoded information to a message that is displayed in the application's screen.

The implemented application uses the back-facing camera of the Galaxy Nexus phone. A preview of the implemented application can be found in Figure 7.4. The phone can communicate with the transmitter using the USB interface that was explained in Section 5.3.

The application has buttons for controlling the exposure time as well as initiating communication and decoding. The two left buttons are used to lower the exposure time of the camera sensor. Due to the limitation that were explained in Section 6.3, we first need to manually lower the exposure

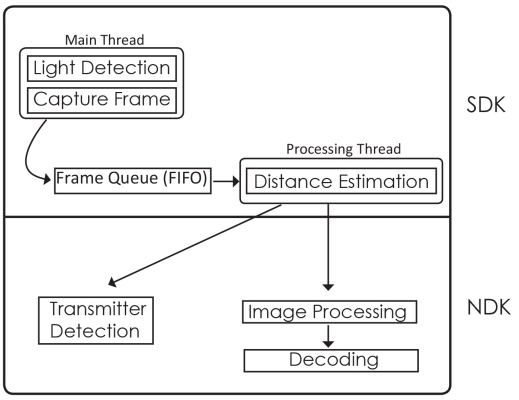


Figure 7.3: **Implementation Structure.** The image shows the structure of the implemented application and the mapping of the most important methods.

Figure 7.4: **Application screenshot.** The left buttons are used for lowering the exposure time, while the right ones are used for initiating communication and debugging.

compensation (top left button), then place a bright light source in front of the camera and then lock the exposure time (bottom left).

In the implemented application we have introduced a simple communication protocol that has to be executed before the start of the transmissions so that both sides can estimate their distance and set the channel's capacity, as mentioned in Section 7.1.2. In addition, the protocol synchronizes the first frame reception so as to maintain synchronization as will be explained in the following section.

7.2.2 Synchronization

As has been extensively discussed in [22], in the context of camera-based visible light communication links, synchronization regards the problem of receiving all the transmitted packets. Smartphone cameras have different and fluctuating frame rates, which makes synchronization between the two

components a major challenge when it comes to exploiting the full reception rate of the phones. There are several techniques that can ensure that a packet has been received by the receiver, such as acknowledgements. We believe that by having each packet to be acknowledged by the receiver, it adds a lot of overhead and also requires a two-way communication.

The major problem is that the captured frames are not received at fixed time intervals. This means that between two consecutive frame receptions there is a time gap that we call jitter. This jitter can be observed from Figure 7.5. It can be seen that instead of receiving the frames every T (black arrows), the reception is delayed by ΔT (red arrows), which is random. The reception time of frame i is given by Equation 7.4, where T is the expected reception period and ΔT is the jitter.

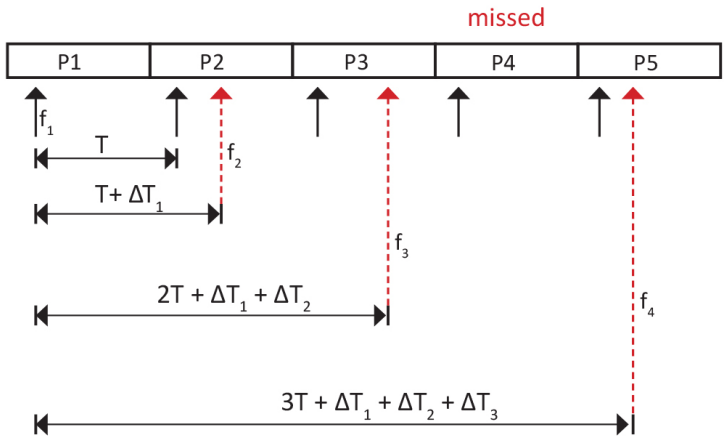


Figure 7.5: **Packet synchronization problem.** The black arrows indicate the expected reception time of each frame. The red arrows represent the delayed receptions due to jitter.

$$frame_i = (i-1) \cdot T + \sum_{n=1}^{i-1} \Delta T_n \tag{7.4}$$

It can be seen that the jitter of each frame is the accumulative value of the previous jitters. This effect causes packet 4 to be missed in Figure 7.5. Having a big jitter value increases the frequency of missed packets.

To highlight the importance of the jitter values we measured the jitter values in different frame rates. It has to be mentioned that the Android phones do not offer methods for setting the exact desired frame rate. However, they provide methods for setting the frame range that the camera will capture frames, such as 15-25fps. As a result, the frame rate is not guaranteed. In order to get the desired frame rate we set the frame rate to the highest available frame rate range, but our application introduces the desired delay T

between the frame receptions. For example, in order to get 2fps we manually delay each reception for 500ms. The results are summarized in Table 7.1.

Table 7.1: **Jitter measurements for different fps.** For each frame rate we measured the jitter of 100 consecutive frames.

Frame Rate (fps)	**Mean Jitter (ms)**	**Max Jitter (ms)**
1	7	22
2	6	19
5	7.5	63
10	58	154

It can be observed that as we increase the frame rate the jitter increases. This means that as the frame rate increases, we will experience more missed packets. As a consequence, we need a way to reduce the packet misses in order to receive all the transmitted packets. In other words, we need to make the red arrows, of Figure 7.5, go as close to black ones, that indicate the expected reception rate. For example, for a frame rate of 2fps, instead of receiving each packet at a period of 500ms we can reduce it by X, as shown in Equation 7.5.

$$delay = T - X \tag{7.5}$$

The problem that still remains is selecting the value of X. We assume that the jitter values follow a random distribution. Therefore, we choose to calculate X as the expected value of the jitter values. Equation 7.6 shows the calculation of the expected value for a given distribution of jitter values v_i, where p_i is the probability of value i.

$$E[\Delta T] = \sum_{n=1}^{k} p_n \cdot v_n \tag{7.6}$$

After applying the above analysis we expect that the frame reception times will oscillate around the value of period T. As a result, the reception time jitter will not keep increasing in every new frame reception. Each packet is repeated by the transmitter for a specific amount of time which is the period T of the reception rate. Furthermore, we make sure that each frame capture occurs approximately at the middle of the transmitter packet. Based on that and the fact that a whole packet can be decoded from a single frame we can estimate the channel's data rate. For example, if the packet size is 8 symbols and the reception rate is 1fps, then the data rate is 8 symbols per second, which corresponds to 0.5 bytes per second.

In order to synchronize the initial transmission, so that all the rest will come "around" period T, we utilize the light sensor of the phone. The main reason is that we want the two components to have a common time

reference point. The light sensor is sensitive enough to guarantee that when the transmitter has turned on its LED, the receiver will know the timestamp of the on operation. The whole procedure is initiated by the "initiator" side that turns on its LED first. The rest operations are executed as shown in Figure 7.6.

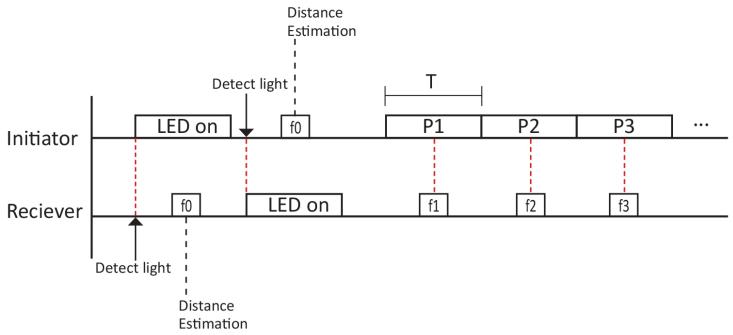


Figure 7.6: **Synchronization protocol overview.** The initiator side initiates the procedure by turning on its LED. Then both sides calculate their distance and set the channel capacity. The packets are repeated for period T that corresponds to the frame rate period. The frames are captured "around" the middle of each packet.

The transmissions begin right after both sides has finished calculating the channel's capacity by executing the transmitter detection. The first reception is delayed in order occur "around" the middle of first packet. In that way, we reduce the effect of the increasing jitter that causes packet misses. In the context of this thesis, we do not evaluate the effect of the above solution, so we cannot estimate DynaLight's reliable data rate.

Chapter 8

Evaluation

In this chapter we present the evaluation of the developed system. In Section 8.1, we present all the experiments that refer to way that DynaLight adjusts its channel capacity based on the detected distance as explained in Section 7.1. We continue our evaluation on the implemented decoding image processing pipeline that was presented in Section 6.2.2. The evaluation of the decoding regards the avoidance of the over-exposure effect, the impact of ambient light, as well as, its real-time performance.

For evaluation we utilized smartphone application that was explained in Section 7.2. The application was mainly used in order capture and save the frames to be evaluated. In order to be able to repeat our experiments we created Python scripts that execute the same image processing code as in the smartphone. In that way we could evaluate and improve our implemented methods offline. For example, we could capture several frames and process them offline and get the same results as running on the smartphone device.

8.1 Adapting to Variable Distance

In this section we compare our transmitter detection method, that was presented in Section 6.2.1, with the one that was presented in Luxapose [21]. Recall that we improved the existing methods by adding a contrast increase step. We show the impact on the detected blob size with respect to the amount of information we can fit in every frame. Moreover, we evaluate the impact of the selected diffuser lens.

8.1.1 Blob Size vs Distance

Methodology

We begin by evaluating the importance of using a diffuser lens. We used the diffuser lens to reduce the viewing angle of the LED from 110 ° to 20 °. This results in having a more narrow and directed light beam but also a

brighter light source, due to the fact that the light is diffused. We compare the performance of the Luxapose transmitter detection in the case of using or not the lens.

Furthermore, we evaluate our own improvement in the detection mechanism that was presented in Section 6.2.1. The evaluation is done in 9 different positions of varying distance from 20cm to 120cm. For each position, we captured 50 frames and averaged the detected transmitter radius.

It has to be clarified that when we refer to the Luxapose, we refer to the transmitter detection method that was presented in [21], *without* our addition. In DynaLight, we use both the diffuser lens and the contrast increase.

As has been mentioned above we perform the blob detection when the light is turned fully on and without modulating.

Results

The results of our experiments can be found in Figure 8.1.

Observation 1 *Using the diffuser lens increases the detected blob by 50% in the close distances, while the contrast increase further increases the detected blob by 40%.*

The results of these experiments highlight the importance of using a lens. It can be observed that light diffusion increases the brightness of the transmitted light resulting in greater blob.

Furthermore, we can observe that using the contrast increase step before performing the transmitter detection increases the detected light approximately 40% on average. Increasing the image's contrast proves that it can reveal information that could be "hidden" due to the low exposure settings. Furthermore, by using both the lens and the additional image processing step also results in being able to detect a 20% bigger blob in the distance of 120cm. This means that our additions have increased the coverage of our system.

Observation 2 *The blob size is not proportional to the square of the distance between the transmitter and receiver.*

The relation of the blob size and the distance does not follow the inverse square law. However, the behavior is similar, meaning that it drops exponentially. This is because we are not directly measuring the emitted power from the light source. We believe that this phenomenon is caused by several parameters of the image processing that measures the blob size. More specifically, the used transmitter detection method blurs the image before measuring the transmitter blob. As a result, we are not measuring the transmitter's size but the light "aura" that it creates. The created light "aura"

is more noticeable in the closer distances and that is why the improvement of the contrast increase is higher. As the distance increases and due to the low exposure time the "aura" fades away. This phenomenon in the closer distances may also be amplified by the camera's protective lens that may introduce additional reflections.

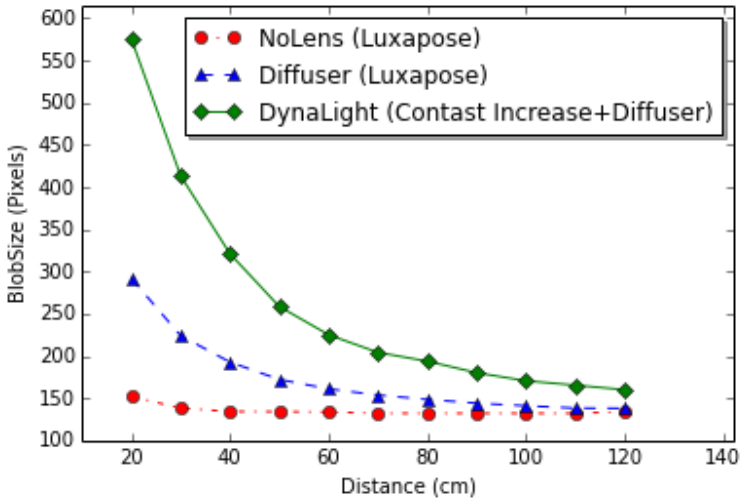


Figure 8.1: **Blob size vs Distance.** The red and blue lines refer to the transmitter detection method presented in Luxapose [21]. The green line regards DynaLight, that uses both the diffuser and the contrast increase.

8.1.2 Number of Symbols vs Distance

Methodology

After showing the impact of the diffurer and contrast increase in the size blob size we continue our evaluation on the number of symbols that can fit in the detected blob for both DynaLight and Luxapose. The calculation of the number of symbols is done as explained in Section 7.1.2, excluding the preamble symbols.

For the evaluation we used the same captured frames as in the previous experiment. After detecting the blob size we calculate the number of symbols that can fit in the detected blob. It has to be mentioned that we perform this evaluation using the diffuser lens in both Luxapose and DynaLight.

Results

The behavior in Figure 8.2 is similar to the previously explained figure as expected. The increase in the number of symbols is more noticeable in the closer distances where the detected blob is bigger. The positive observation

is that at the distance of 120cm we can still transmit 2 symbols per packet. Moreover, at the distance of 90cm we can transmit 50% more data compared to Luxapose. Recall that the number of symbols in the figure refers to the number of symbols that we can definitely decode from a single frame, as explained in Section 7.1.2.

However, it is noticeable that the amount of symbols does not decrease linearly after the distance of 60cm. This is mainly caused by the fact that the detected blob decreases but is still "large" enough to include the same amount of symbols.

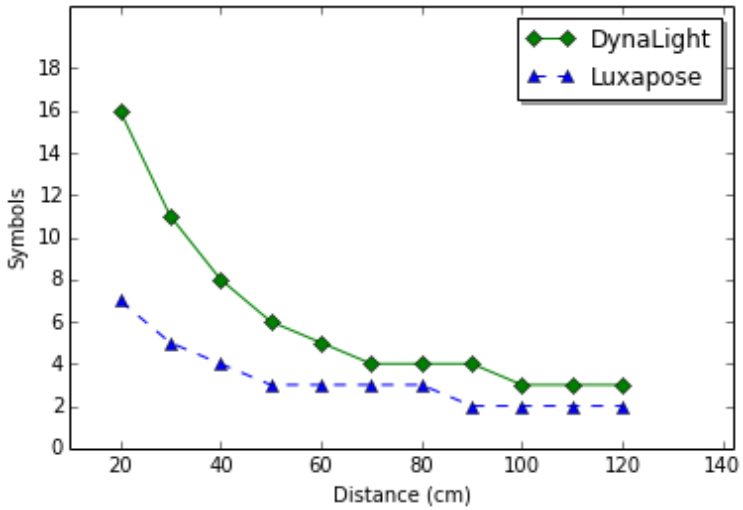


Figure 8.2: **Number of symbols vs Distance.** We get 40% more symbols on average, compared to Luxapose.

8.1.3 DynaLight's Adaptive Channel Capacity

In this section we evaluate the influence of the proposed channel capacity adaptation that was presented in Section 7.1.2. Recall that in the state-of-the-art systems the channel capacity is fixed. Having a dynamic channel capacity increases the flexibility of the implemented system when it comes to operating in different distances.

Methodology

The goal of the three evaluation experiments that we conducted is to highlight the importance of adapting the channel capacity based on the distance between the transmitter and the receiver. In order to achieve that, we evaluate our system in 3 different distances, 20cm, 60cm and 90cm. The distances were chosen so that we fit 16, 8 and 4 symbols accordingly (excluding the

preamble symbols). In the first two experiments we followed the state-of-the-art way of using a fixed packet size of 4 and 16 symbols, that are the optimal for the high and short distance accordingly. In the last experiment we present how DynaLight modifies the channel capacity, dynamically, based on the detected distance between the transmitter and the receiver. For each distance we processed 50 frames of the same packet.

Results

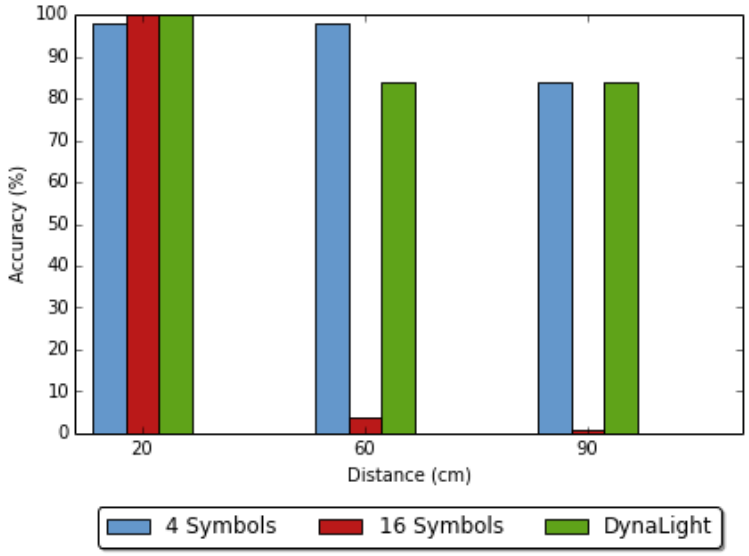


Figure 8.3: **Accuracy vs number of symbols in 3 distances.** Dynamic channel capacity is preferred for more efficient link exploitation.

Observation 3 *Choosing a small packet size underestimates the channel capacity but ensures that packets can be decoded in all distances.*

As can be observed in Figure 8.3, by selecting a packet size of 4 symbols we have an accurate packet detection of above 85%, in all the different positions. This happens because the 4 symbols are the number of symbols that can fit in the greatest evaluated distance (90cm). In all the shorter positions the detected blob is bigger, meaning that it can fit more that 4 symbols. As a consequence, 4 symbols per frame may offer high reliability but is not the most efficient for all distances.

Observation 4 *Choosing the biggest packet size reduces reliability.*

On the other hand, when we select the greatest packet size and we increase the distance, then the accurate transmissions are expected to drop. For

example in the 60cm we cannot fit more than 8 symbols, which reduces the probability of correct reception. However, we see that very few packets were decoded properly. This can be explained by the fact that in the 60cm distance the detected blob is big enough to fit 8 symbols. Recall that our calculations are based on including each packet twice in the detected blob (Figure 7.2). As a result, there is a small chance that the 16bits are perfectly aligned and fit the detected blob radius. Furthermore, decoding is impossible in the highest evaluating distance because the blob size is too small to fit the total of 16 symbols.

Observation 5 *DynaLight's channel capacity adaptation ensures efficient channel utilization along with an accuracy above 85%.*

We show that by modifying the channel capacity we can achieve an accuracy above 85% in all individual distances. As we can observe from the results, as we move away from light source the accuracy drops. We believe that having an accuracy of 100% is impossible due to numerous factors that influence our system. The basic reason is the image processing pipeline that introduces noise in high distances (mostly by thresholding method). Moreover, as we move away from light source there is a high probability that the camera sensor will introduce additional noise. This means that some pixels might not be saturated the same way as in the closer distances, which may affect our decoding performance.

8.2 Decoding

In this section we evaluate the decoding pipeline that was presented in Section 6.2.2. We begin by evaluating the which methods offers the best results in avoiding the over-exposure effects. Then, we show how robust DynaLight is to ambient light. Finally, we discuss its real time performance.

8.2.1 Overcoming Over-exposure Effects

Methodology

We begin by comparing the decoding methods (pixels selection) with respect to the one used in Luxapose (middle column), in order to see which method has the highest decoding accuracy. We chose to evaluate the decoding method in the distance of 40cm where we can fit 8 symbols. We evaluate our methods using 3 different packets 1) "01010101", 2) "010**11**001" and 3) "101**00**101". Packets 2 and 3 have two consecutive identical symbols compared to packet 1. Evaluating packets 2 and 3 will highlight possible limitations that are introduced by the over-exposure effect.

For each packet we captured and decoded 50 frames. We relate the accuracy of each method in terms of how many times the packet was accurately

decoded. If the decoded symbol sequence is different or has a different length then we consider a false decoding.

Results

Observation 6 *The over-exposure effect is more observable in the center of the blob, therefore, we cannot accurately decode data from the middle column of the blob.*

It can be clearly seen from Figure 8.4 that the pixel selection method used in Luxapose cannot be applied in our case. The over-exposure effect introduces too much noise in the signal. When it comes to the second pixel selection method, we can observe that it dramatically increases the accurate decoding but the percentage differs between the packets. *We conclude that the middle column of the blob may contain the most bright signal but this fact increases the over-exposure effect.* This can be clearly observed in the case of packet 3 where the two consecutive 0 symbols can not be accurately decoded. Recall that the over-exposure effect tends to enlarge the size of the white bands. In the case of packet 3, there are cases where the white bands appear bigger than the expected ones, resulting in false decoding. The third pixel selection shows that it offers the best option with an accuracy of more than 90%. Moreover, these experiments showed the importance of having an automatic offset selection mechanism, as in the second method. The closer to the vertical radius column, the more over-exposure noise the signal will experience. On the other hand, a high offset value may also lead to false decoding, due to the fact that the farther away we go from the center of light source, the noisy the signal can get.

8.2.2 Influence of Ambient Light

In this section we evaluate the impact of ambient light with respect to accurate decoding. Since we did not have a light sensor, we created three different experiment conditions by changing the amount of light in the experiments room. In the first case there is no ambient light in the room. In the second case we turned on half of the room's ceiling lights, as depicted in Figure 8.5. In the third case, we turned on all the room's ceiling lights and used an additional light source (lamp) right above the VLC link, as shown in Figure 8.6. As in the previous experiments we capture 50 frames for every individual experiment.

Methodology

We evaluate the impact of ambient light in 40cm and 90cm. The two distances were selected based on the number of symbols that can be fitted,

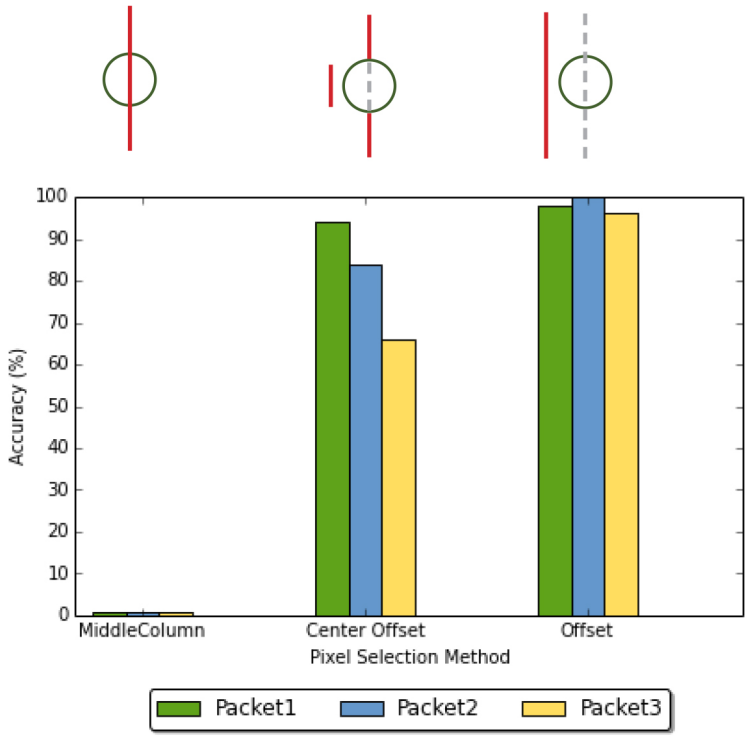


Figure 8.4: **Accuracy of different pixel selection methods using 3 different symbol combinations.** Packet1= "01010101", Packet2= "01011001" and Packet3= "10100101". Above the figure we show a simple representation of each method, similar to Figure 6.4.

namely 8 and 4 accordingly (excluding preamble symbols). In the first experiment we wanted to show the impact of ambient light in the least accurate packet of the previous section, which was packet 3, believing that ambient light will amplify the over-exposure effects. In the second experiment we used two slightly different packets 4) "0110" and 5) "1001" (similar to previous section), that can show the impact of ambient light as well as the over-exposure effect on the decoding.

Results

Observation 7 *Ambient light does not affect our system significantly in both short and great distances, due to low exposure settings.*

As can be seen from Figures 8.7 and 8.8, there is no significant impact on the decoding accuracy introduced by the ambient light in both cases. We believe that this is caused by the low exposure settings that enable only very bright light to be visible in the captured image. This observation highlights

Figure 8.5: **Low ambient light setup.** Half of the ceiling lights are on.

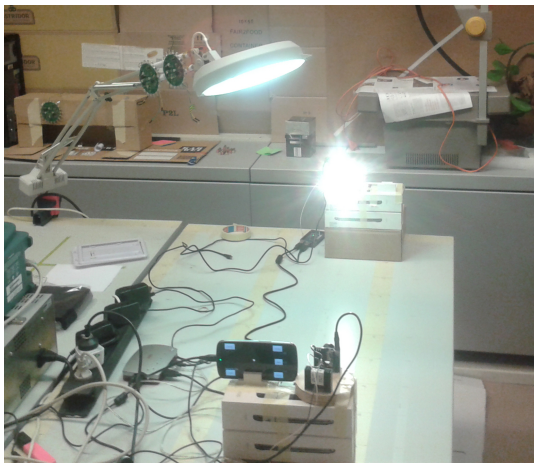


Figure 8.6: **High ambient light setup.** All of the ceiling lights are on plus a lamp pointing above the VLC link.

the benefit of using camera sensors instead of photodiodes or LEDs, where external light sources introduce unavoidable noise. Nevertheless, we believe that the ambient light that we introduced is not that bright compared to the light of the transmitter. However, having multiple concurrent transmitters operating at the same light intensity will definitely affect the systems performance with respect to introducing noise.

Finally, we observe a slight drop in the accuracy as the distance increases. This can be observed in packets 4 and 5 that are evaluated in a higher distance. This fact is similar to Figure 8.3, and is mostly caused by noise that is introduced by the image processing methods.

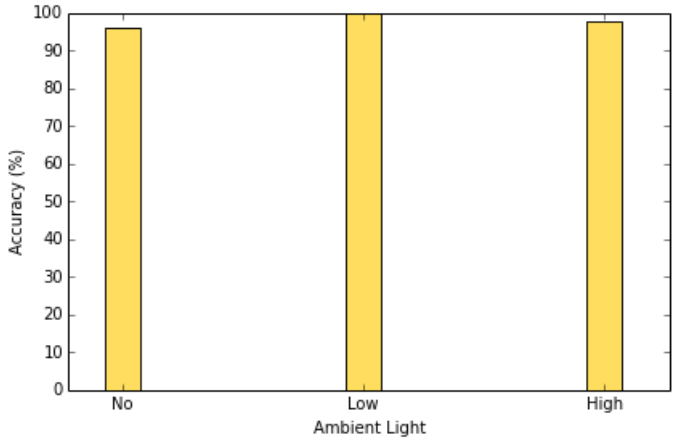


Figure 8.7: **Accuracy vs ambient light for Packet3 in 40cm.** No impact is observed in close distances.

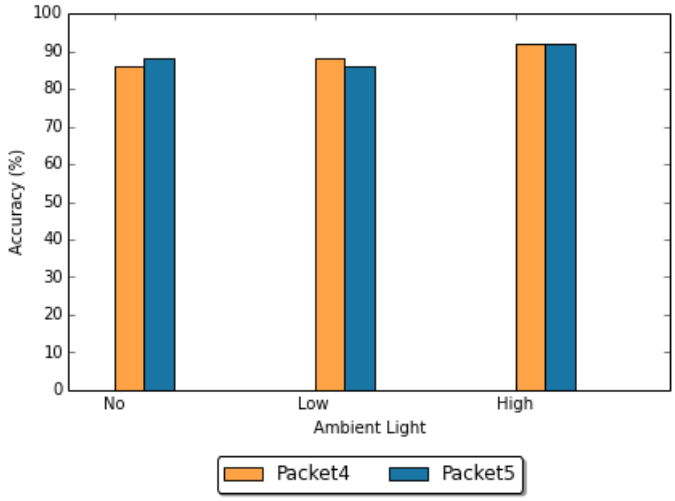


Figure 8.8: **Accuracy vs ambient light in 90cm.** Packet4= "0110" and Packet5= "1001". No significant impact is observed even in the large distance.

8.2.3 Execution Time

Compared to the Luxapose project, all the decoding algorithms of Dyna-Light are executed locally in the device. We measured the real-time performance of our decoding algorithm, so as to identify which are the most time consuming methods. However, since we are not using a high-end device we have noticed that the decoding time varies.

The average decoding time for decoding a single frame varies from 1 to 2 seconds. However, there are times that the decoding time drops to 600ms, which may be caused by the scheduling mechanisms of the Android OS. We further experimented with changing the scheduling priority of the processing thread to the maximum available by the operating system. However, increasing a thread's priority proved to reduce the processing time of the decoding slightly, but it introduced additional jitter to the reception of the frames. We believe that the additional delay is caused by the fact that the processing thread gets a similar priority with the camera's capturing thread, that provides the captured frames. As a result, we believe that the scheduling mechanism of the operating system is sharing the CPU between the high priority threads, including the processing thread. We believe that we can obtain better performance by using a more powerful device with respect to processing power.

In addition, we noticed that the OpenCV methods require lots of processing power and memory. The most time consuming methods are the contrast increase and the adaptive thresholding. As a result, we believe that there is room for improvement when it comes to improving the efficiency of the image processing techniques.

Chapter 9

Conclusions

9.1 Conclusions

Recent research has focused on creating high-rate visible light communication links. Camera-based VLC has shown that it offers a great potential with respect to applicability, due to the fact that LED lighting and smartphones are widely utilized. However, the existing VLC systems assume a fixed distance between the transmitters and the receivers. Moreover, building a universal camera-based VLC link is not considered an easy task due to wide diversity of the hardware capabilities of the smartphones. The most demanding challenge is synchronizing the transmissions and the receptions. The problem is caused by the unstable frame rate of the smartphone cameras. Therefore, any camera-based communication link may suffer from dynamic losses.

In this thesis, we highlight the need for an flexible visible light communication link that will dynamically adjust its channels capacity based on the distance between the transmitter and the receiver. Based on that, we present DynaLight, an adaptive VLC system for smartphones. Our system is able to estimate the distance between the transmitter and the receiver and modify its channel capacity. This results in an efficient channel utilization in different distances.

Furthermore, by applying an image processing enhancement we managed to improve the transmitter detection mechanism, that was presented in a recent related work, by 40%, meaning that we increased the feasible channel capacity of our link. In addition, we identified and avoided effects that are caused by the limited camera control of low cost smartphones.

Finally, we developed a smartphone application that implements our solutions. During our work we experienced the two main problems of synchronization, which refer to the mixed and missed packet problems, that are introduced by the fluctuating reception rate. We avoid the mixed packet problem by making sure that from each captured frame a complete packet can be de-

coded. As a result, the system's reliability depends on synchronizing the packet's reception. We believe that our observations and contributions will influence other related projects that will take advantage of the potentials that the visible light offers in creating communication links.

9.2 Future Work

In the context of this thesis we did not fully evaluate our proposed solution for the reception synchronization. We observed that our solution has the ability to reduce the effect of the reception jitter and enable us to further improve the link's data rate. By further studying the behavior of the jitter we may be able to predict the next frame reception or better understand the causes of reception rate fluctuation. Furthermore, the phone can continuously monitor the distribution of the jitter values, so as to recalculate the expected values that are used to reduce the influence of the jitter increase. In that way, the phone can dynamically adjust to the fluctuation of the frame rate.

Our system is robust to ambient light and this caused by the low exposure settings. We believe that utilizing more than one transmitters will introduce errors in the decoding because the different bands will collide. We believe that by utilizing different light colors might be a decent alternative for separating transmissions. However, this implies that the whole image processing pipeline has to be reconsidered and modified so as to support "colored" VLC transmissions.

When it comes to further improvements of the current setup, we believe that DynaLight's performance can increase notably by utilizing a more recent smartphone. Being able to execute the image processing pipeline faster will also enable DynaLight to cope with mobility. This means that transmitter detection could be executed real-time, along with the decoding, so as to monitor if the transmitter has changed its position.

With respect to improvements to the detection and decoding process, we believe that OpenCV methods introduce overhead that could be avoided by implementing methods such as thresholding. A further improvement would be to also crop the detected transmitter area and process it individually in order to reduce the computation overhead and memory requirements.

Appendix A

Transmitter Experiments

A.1 Android Limitations

One of the initial goals of this project was to create an application that could use commercial hardware to create simple VLC links. Smartphones are already equipped with LEDs that could be used in creating VLC links. In the first phase of this project we experimented with the phone's flash light to check if it can be toggled in such a high rate that could be used in our project. We concluded that this is not an option due to the fact that the Android operating system does not offer much flexibility when it comes to modulating the LED in high frequencies. More specifically, we observed an erratic behavior on the on off pattern when the LED was modulating faster than 5-7 Hz. We believe that this is mainly caused by the Android Java layer that does not enable real-time operations. There are several real-time patches that could be applied to the kernel of the Android OS but it is rather impractical and out of the scope of the current thesis. We also investigated using native code to access the LED but to the best of our knowledge, there is no existing API that could support such operation. Thus, utilizing the device's LED is not as trivial as one might consider.

A.2 External Flash Light

One of the initial motivations of the project was to create a commercial application that will enable smartphones to create flexible visible light communication links. Therefore, we created collaboration with the company that produces the external flash Iblazr [5] (version 1). This device operates via the audio jack of the smartphone. By using stereophonic audio signal we were able to control the behavior of the light. Nevertheless, we could not reach more than a few hundreds of Hz which again is too limited. We discussed with the engineers, that created the device, and came to the conclusion that the problem was caused by the existing firmware that was

running on the device, since it was designed as an external flash light and not as a VLC component. However, the new version of Iblazr extends its capabilities in terms of connectivity and programmability. It can be controlled via Bluetooth and supports firmware flash. We believe it could assist in creating commercial multi-transmitter VLC links but is something that further needs to be investigated.

Based on the above facts, we decided to offload the LED modulation to a microcontroller (Arduino).

Bibliography

[1] Arduino MEGA ADK. http://www.arduino.cc/en/Main/ArduinoBoardMegaADK. [Online, Accessed May-2015].

[2] ByteLight. http://www.bytelight.com/. [Online, Accessed November-2015].

[3] Diffuser Lens LM1 for Cree MC-E Datasheet. https://www.maritex.com.pl/media/uploads/products/op/LEDIL-LM1.pdf. [Online, Accessed June-2015].

[4] Disney Research on Visible Light Communication. http://www.disneyresearch.com/project/visible-light-communication/. [Online, Accessed April-2015].

[5] Iblazr. http://iblazr.com/. [Online, Accessed April-2015].

[6] MAX3421E. https://www.maximintegrated.com/en/products/interface/controllers-expanders/MAX3421E.html. [Online, Accessed April-2015].

[7] MOSFET IRL1004 Datasheet. http://www.irf.com/product-info/datasheets/data/irl1004.pdf. [Online, Accessed October-2015].

[8] NPN Switching Transistor 2N3904 Datasheet. https://www.sparkfun.com/datasheets/Components/2N3904.pdf. [Online, Accessed October-2015].

[9] pureLiFi. http://purelifi.com/. [Online, Accessed November-2015].

[10] XLamp MC-E. http://www.cree.com/LED-Components-and-Modules/Products/XLamp/Arrays-Directional/XLamp-MCE. [Online, Accessed May-2015].

[11] Christos Danakis, Mostafa Afgani, Gordon Povey, Ian Underwood, and Harald Haas. Using a cmos camera sensor for visible light communication. In *Globecom Workshops (GC Wkshps), 2012 IEEE*, pages 1244–1248. IEEE, 2012.

[12] Hany Elgala, Raed Mesleh, and Harald Haas. Indoor broadcasting via white leds and ofdm. *Consumer Electronics, IEEE Transactions on*, 55(3):1127–1134, 2009.

[13] Domenico Giustiniano, Nils Ole Tippenhauer, and Stefan Mangold. Low-complexity visible light networking with led-to-led communication. In *Wireless Days (WD), 2012 IFIP*, pages 1–8. IEEE, 2012.

[14] J Grubor, OC Jamett, JW Walewski, S Randel, and K-D Langer. High-speed wireless indoor communication via visible light. *ITG-Fachbericht-Breitbandversorgung in Deutschland-Vielfalt für alle?*, 2007.

[15] Jelena Grubor, Sian Chong Jeffrey Lee, Klaus-Dieter Langer, Ton Koonen, and Joachim W Walewski. Wireless high-speed data transmission with phosphorescent white-light leds. *ECOC 2007*, 2007.

[16] Tian Hao, Ruogu Zhou, and Guoliang Xing. Cobra: color barcode streaming for smartphone systems. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 85–98. ACM, 2012.

[17] Harald Haas. High-speed wireless networking using visible light. `https://spie.org/x93593.xml`. [Online, Accessed November-2015].

[18] Wenjun Hu, Jingshu Mao, Zihui Huang, Yiqing Xue, Junfeng She, Kaigui Bian, and Guobin Shen. Strata: layered coding for scalable visual communication. In *Proceedings of the 20th annual international conference on Mobile computing and networking*, pages 79–90. ACM, 2014.

[19] Lennart Pieter Klaver. Design of a network stack for directional visible light communication. *ratio*, 3:4, 2014.

[20] Toshihiko Komine and Masao Nakagawa. Fundamental analysis for visible-light communication system using led lights. *Consumer Electronics, IEEE Transactions on*, 50(1):100–107, 2004.

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

[22] Hui-Yu Lee, Hao-Min Lin, Yu-Lin Wei, Hsin-I Wu, Hsin-Mu Tsai, and Kate Ching-Ju Lin. Rollinglight: Enabling line-of-sight light-to-camera communications. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 167–180. ACM, 2015.

[23] Liqun Li, Pan Hu, Chunyi Peng, Guobin Shen, and Feng Zhao. Epsilon: A visible light based positioning system. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pages 331–343. USENIX Association, 2014.

[24] Edmundo Monteiro and Steve Hranilovic. Constellation design for color-shift keying using interior point methods. In *Globecom Workshops (GC Wkshps), 2012 IEEE*, pages 1224–1228. IEEE, 2012.

[25] Edmundo Monteiro and Steve Hranilovic. Design and implementation of color-shift keying for visible light communications. *Lightwave Technology, Journal of*, 32(10):2053–2060, 2014.

[26] Niranjini Rajagopal, Patrick Lazik, and Anthony Rowe. Visual light landmarks for mobile devices. In *Proceedings of the 13th international symposium on Information processing in sensor networks*, pages 249–260. IEEE Press, 2014.

[27] Sridhar Rajagopal, Richard D Roberts, and Sang-Kyu Lim. Ieee 802.15. 7 visible light communication: modulation schemes and dimming support. *Communications Magazine, IEEE*, 50(3):72–82, 2012.

[28] Richard D Roberts. Undersampled frequency shift on-off keying (ufsook) for camera communications (camcom). In *Wireless and Optical Communication Conference (WOCC), 2013 22nd*, pages 645–648. IEEE, 2013.

[29] Stefan Schmid, Giorgio Corbellini, Stefan Mangold, and Thomas R Gross. Led-to-led visible light communication networks. In *Proceedings of the fourteenth ACM international symposium on Mobile ad hoc networking and computing*, pages 1–10. ACM, 2013.

[30] Yuichi TANAKA, Toshihiko Komine, Shinichiro Haruyama, and Masao Nakagawa. Indoor visible communication utilizing plural white leds as lighting. In *Personal, Indoor and Mobile Radio Communications, 2001 12th IEEE International Symposium on*, volume 2, pages F–81. IEEE, 2001.

[31] Qing Wang and Domenico Giustiniano. Communication networks of visible light emitting diodes with intra-frame bidirectional transmission. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, pages 21–28. ACM, 2014.

[32] Qing Wang, Domenico Giustiniano, and Daniele Puccinelli. An open source research platform for embedded visible light networking. *Wireless Communications, IEEE*, 22(2):94–100, 2015.

[33] Grace Woo, Andrew Lippman, and Ramesh Raskar. Vrcodes: Unobtrusive and active visual codes for interaction by exploiting rolling shutter. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 59–64. IEEE, 2012.

[34] Zhice Yang, Zeyu Wang, Jiansong Zhang, Chenyu Huang, and Qian Zhang. Wearables can afford: Light-weight indoor positioning with visible light. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 317–330. ACM, 2015.