Indoor Localization Through Visible Light Communication

Group 17

***Abstract*—Visible Light Communication (VLC) has attracted many researchers and companies to investigate its potential in indoor localization. The method can be implemented in a variety of ways. This report presents an implementation of determining the location of a smartphone user through the Angle of Arrival technique which simplified the existing method. In an office-like environment accuracy of about 10 cm has been achieved.**

I. Introduction

Visible Light Communication (VLC) is a method that allows for accurate indoor positioning. A basic setup includes a set of lamps as transmitters and a smartphone as a receiver. The light transmitted from the lamps is modulated in a frequency imperceivable to the human eye. The camera of the smartphone however is able to distinguish the modulated light from unmodulated light, using the *rolling shutter* technique, and decode it. This makes it possible for the lamps to send an unique identifier to the smartphone, which can be translated into their absolute coordinates. The Angle-of-Arrival (AoA) method then uses the knowledge of the absolute lamp coordinates and the positions of the lamps captured on the screen of the smartphone, to calculate the smartphone's absolute coordinates. A more detailed description of this method can be found in [1].

The efforts in this report are a continuation on the work presented in the master thesis of a graduate student from the Delft University of Technology [2]. The VLC system, named DynaLight, makes it possible to estimate the distance between the lamps and the smartphone. The application that was written for this thesis has been modified in order to use it for indoor localization.

This report is organized as follows. Section III outlines the method to obtain a location from the light source's identifiers and proposes a novelty in the decoding scheme as well as in the localization method used. Section II contains details on the setup that was used for testing purposes. Section IV highlights the results obtained for the used setup. **??** briefly mentions the challenges that are left to solve. Finally Section VI gives a short guide on how to use the application and set up the environment for indoor localization.

II. Setup

The setup for this project included three LED lamps as transmitters with each its own Arduino microcontroller to modulate the emitted light. The room in which the lamps were installed was the Dark Room on the ninth floor of the faculty EEMCS belonging to the Delft University of Technology. The formation of the lamps can be seen in Figure 1.

Fig. 1: LED lamps positioning

For testing convenience the lamps were installed onto a wall, instead of a ceiling. This way it was possible to use the backside camera of the smartphone without having to lay down on the ground to capture images. The coordinates of the lamps were measured from a point of origin marked on the floor of the room. The coordinates in this reference frame shall be considered the "absolute coordinates" in this report.

For the modulation of the light the Arduino board were programmed identically, except for the identifier that would be transmitted. On-and-Off Keying (OOK) was chosen as the coding scheme, because it allowed for a larger range of identifiers and more accurate decoding. A scheme like Manchester encoding could be more fault tolerant, but would not be realistic when increasing the number of lamps. The codes were programmed binary into the Arduino boards, where 0 would turn off the lamp and 1 would turn on the lamp. The codes were transmitted at a frequency of 3000Hz with an easily identifiable preamble of 01110. Each lamp's id contains 3 bits which makes it possible to have 8 unique lamp ids.

The smartphone used as the receiver had the following specifications:

- Samsung Galaxy S4 (GT-I9505)
- Android 4.4.2 (KitKat)
- 13 Megapixels backside camera
- 1080 x 1920 screen resolution

The Integrated Development Environment (IDE) for this project consisted of the following software:

- Eclipse Mars (4.5.2)
- Java Native Interface (JNI)
- Android NDK R10e
- OpenCV 2.4.8.2

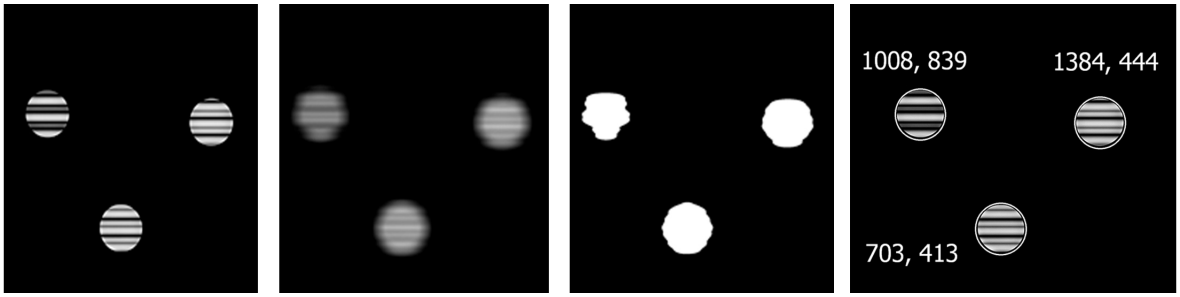


Fig. 2: An overview of the blob detection algorithm. From left to right: original (cropped) image, blurred, threshold filter, circle detection

The Android NDK toolset allowed the OpenCV library to be operable, which made it possible to use the wide range of image processing functions available in OpenCV. The JNI was used as a communication link between native code and Java code.

III. Method and improvements

In this section the method that was used in the project will be considered. Throughout the section the rest of the report term ”blob” will be used for a lamp that was recorded on the screen of the smartphone. The term ”id” will be used as the shorthand notation for the identifier of a lamp.

A. Pipeline

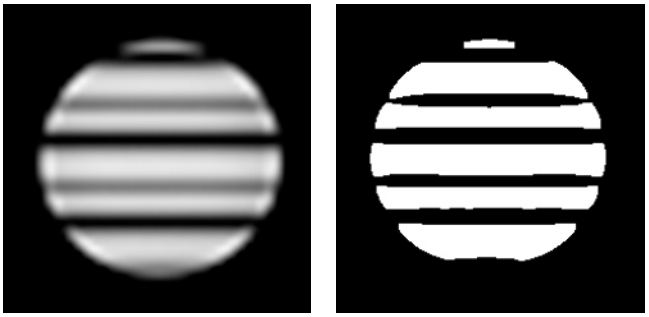
In order to produce a set of absolute coordinates determining the smartphone's location from an image of multiple blobs, several steps need to be undertaken. The following pipeline will have as input an image of at least three blobs, and outputs the absolute location of the smartphone:

1) Detect blob positions on image
2) Segment blobs into separate images
3) Decode the id of each blob
4) Link id to real location of lamps
5) Use AoA-method to determine location

Step 1: Detecting the blob positions on the image can be done by convolving the original image with a high-intensity blurring kernel, followed by making a binary image using a threshold filter. This results in white circular objects, which can be detected by a contour-finding algorithm. The algorithm will pinpoint the centers of the blobs as well as calculate their radii. This step is illustrated in Figure 2.

Step 2: Knowing the center points and the radii of the blobs it is possible to segment rectangular images of individual blobs from the original image of multiple blobs. Special attention has been given to blobs near or on the edge of the image. It is possible that these blobs can still contain useable information and are therefore also segmented.

Step 3: Each segmented image undergoes a series of image processing techniques before the code is read out. This is done to create a clear distinction between when the lamps was on and when the lamp was off. Figure 3 illustrates this (see [1] on how an image is obtained through the rolling shutter effect).



(a) Before image processing (b) After image processing

Fig. 3: Example of processing a segmented image of a blob

Trying to decode the image in Figure 3a directly would be very difficult, since the thin black bars are almost completely covered by the white bars, due to overexposure. Figure 3b on the other hand shows a clear OOK pattern that can be used for decoding.

After Figure 3b has been obtained, the middle column of pixels is extracted from the image and used as the ”decoding line”. This line holds the necessary information to decode the blob (see Figure 4 for an example of a decoding line). A black bar means that the lamp was off when capturing the image with the rolling shutter effect, and a white bar means that the lamp was on. The size of a bar is determined by the time the lamp was on or off. The smallest period of being on or off is determined by the frequency in which the light is modulated. For example, in this report a frequency of 3000Hz was used, which yields a single bar size of 8 pixels in an image. This value can easily be determined experimentally by encoding a block wave in the lamps (i.e. 01 as the code).

The original method to decode the ids of the lamps in [2] was to determine the size of each bar of a blob and determine how many single bars could fit in. For example, if a white bar of 24 pixels was detected and a single bar is 8 pixels wide, then this would be decoded as 111 (since a 1 turns the lamp on and gives a white bar). This methods works correctly when considering just a single lamp from a close distance, but when trying to decode an image with three lamps, this method is not accurate enough. The reason being that in order to fit three blobs onto a single image, the distance between the

camera and the lamps must increase as well. This inevitably will mean that one blob will be composed of less pixels (i.e. lower resolution) and the overexposure of the white bars will "eat away" several pixels of the black bars (see Figure 3a). Image processing alone would not give stable outcomes for the decoded ids. This was the main reason to change the decoding method.

THE DECODING SCHEME

Instead of determining how many multiples of a single bar size (S_b) call this fits in the width of a detected bar, the improved method relies on the transitions that occur between two preambles. A preamble indicates the beginning of a package, so the bars between two preambles represent the id of a lamp. A preamble should have a distinctive pattern, so that the starting and ending point on the image is clear. The current implementation requires the preamble to end on a singular black bar, however this could be modified to become more general.

The decoding of the id starts 1 pixel after the last transition in the preamble occurs (going from white to black). The reason for this will become clear shortly. This point will be called P_0 with a state 0 (i.e. it is a black bar). S_b pixels are added to P_0, which shall be called P_1. P_1 is checked for its state. If its state is 0 that means that no transitions have occurred between P_0 and P_1 and the first bit of the id is a 0. When P_1's state is 1 then a transition did occur then the first bit of the id is a 0. See Figure 4 for an example.

This process repeats itself until point P_f is reached, which is the starting point of the second preamble. The reason for having chosen P_0 as explained above is because of the fact that the white bars "eat away" pixels from the black bars as mentioned earlier. Setting P_0 before the first preamble ends, will include even black bars that are smaller than S_b, in contrast to the method in [2], where these bars would be ignored.

Step 4: The blobs of which the ids are now decoded are linked to their priorly known absolute locations. In case one of the blobs was not decoded correctly the pipeline is flushed and a new image will be processed.

Step 5: The Angle-of-Arrival method is used to calculate the location of the smartphone. In the implementation of [1] a quadratic system of equations was solved to get the proportionality factors between the absolute coordinate system and the smartphone's coordinate system. Also a constant value was used for Z_f, the focal length of the smartphones camera in pixels. Section III-D however explains how the AoA method was improved in order to calculate the absolute location and how Z_f can be calculated to generalize the application for other smartphones.

B. Image Processing Improvements: More Accuracy

In addition to the techniques used in [2], several other techniques were applied in order to improve the decoding accuracy. After implementing the improvements, the application processes a single segmented blob image in the following way:

i) Increase the contrast
ii) Apply gentle blur kernel
iii) Remove noise through mask
iv) Apply threshold filter (OTSU)
v) Erode and dilate the image

Figure 3 illustrates the result of the image processing.

Step iii and v involve two techniques that were not implemented in the image processing pipeline of [2], but were needed to successfully decode the image. The problem with the original implementation was that it did not generate the black and white bars as in Figure 3 as true to reality as needed to decode the ids. Usually this was a matter of single pixels deviations. For example, a bar size would be 3 pixels wide and would be ignored even with the improved decoding technique. Therefore it was crucial for the stability of the decoding performance to improve the image processing pipeline.

The first novel technique that was implemented is the use of a circular mask before determining the threshold value in step iv. The threshold algorithm in step iv determines the threshold value depending on the grayscale values of the entire segmented blob image. The threshold value is what will determine which gray values will become white and which values will become black. After increasing the contrast in step ii, significant amount of noise around the blob would get an increase in gray value and thus a lower threshold value. In step iv the pixels that in reality should be black would be converted into white pixels, resulting in the small bar sizes mentioned earlier. Applying a circular mask that selects the pixels outside of the blob removed the noise and gave more realistic bar sizes due to a higher threshold value. This improved the decoding performance significantly.

The second novel technique that was implemented is eroding and dilating the blob images in step v. Essentially what these two OpenCV functions do, is to remove loose pixels that are floating around the bars and straightening the edges of the bars. Again, this would result in more realistic bar sizes and thus a more stable decoding performance.

C. Decoding Improvement: Transition Detection Method

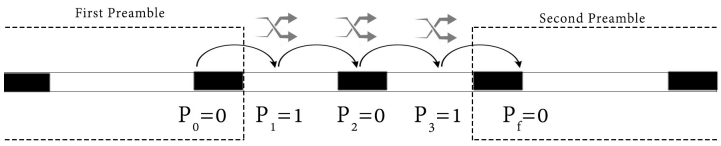


Fig. 4: Transition Decoding Scheme. The crossing arrows icon represents a transition. The id in this decode line is 101.

D. Localization Improvement: Reduced Redundancy

There are three types of coordinate systems involved in this method which are shown in Figure 5. See [1] for more information on the coordinate systems.

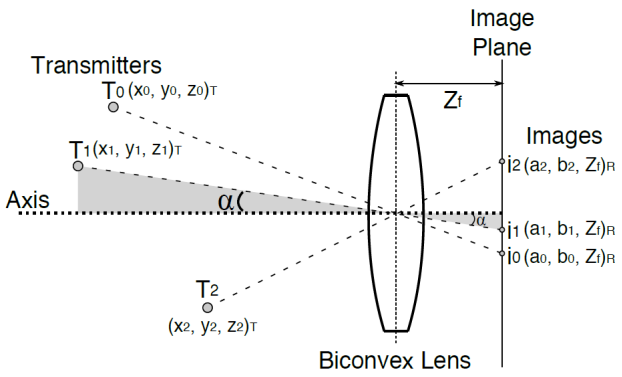


Fig. 5: Location in different coordinate systems [1]

1) Absolute coordinates: x, y, z (cm, known beforehand)
2) Relative coordinates: u, v, w (cm, lens reference point)
3) Blob image coordinates: $a, b,\ Z_f$ (pixels, lens reference point)

The AoA method is based on the theory that the distance between two lamps is same in different coordinate systems. According to the trigonometric theory, the set of coordinates $(a, b,\ Z_f)$ and (u, v, w) are proportional, since they use the same reference. Assume the scaling factor is called K_i for each lamp i. In [1] the K_i values are calculated by using the absolute distance between two lamps, which involves solving a quadratic system of equations.

The improved method makes use of the vector theory for the distance calculations. Each two of the three lamps can make an vector and those vectors should same at different coordinate system as well. If two vectors are equal, they are equal at each direction. In addition, the relationship between $(a, b,\ Z_f)$ and (u, v, w) is still exist. Then list the equation groups as follows:

$$\begin{cases} u_i = K_i \cdot a_i \\ v_i = K_i \cdot b_i \\ w_i = K_i \cdot Z_f \end{cases}, \begin{cases} u_1 - u_2 = x_1 - x_2 \\ u_2 - u_3 = x_2 - x_3 \\ v_1 - v_3 = y_1 - y_2 \end{cases}$$

Combing those two equation group can get an equation group as follows that is an linear system of equations:

$$\begin{cases} K_1 \cdot a_1 - K_2 \cdot a_2 = x_1 - x_2 \\ K_2 \cdot a_2 - K_3 \cdot a_3 = x_2 - x_3 \\ K_1 \cdot b_1 - K_2 \cdot b_2 = y_1 - y_2 \end{cases}$$

After the scaling factor solved, the location of lens can be calculated.

$$\begin{cases} x_c = x_i - K_i \cdot a_i \\ y_c = y_i - K_i \cdot b_i \\ z_c = z_i - K_i \cdot Z_f \end{cases}$$

In this report the parameter Z_f do not need to search for each phone anymore which can get through calculate.

$$Z_f = \frac{z_1 - z_2}{K_1 - K_2} = \frac{z_1 - z_3}{K_1 - K_3} = \frac{z_3 - z_2}{K_3 - K_2}$$

TABLE I: Decode Test Results

Success Rate	Original Length Method	Improved Transition Method
Position 1 (20 images)	25%	75%
Position 2 (13 images)	7.7%	61.5%

IV. Results and Discussion

This section shows the lamp id decode results for the original decoding method [2] and improved decoding method. In addition, the localization results, which use the simplified AoA method, are also shown.

A. Decode Method Result

20 images were captured at one position and 13 images were captured at another position. Using both the decoding method, Table 1 displays the comparative results.

According to the results, the decode success rate was improved by 50%.

B. Localization

In this report the localization results was tested at two locations and output the localized errors at each direction that unit is centimeter. One of them is at location (0, 157.5, 173), with an average error in the three direction of respectively (3.6, 11.4 23.8). The other one is at location (44, 170, 203) which the errors are respectively (5.3, 6.0, 77.7).

V. Challenges

There are three main challenges remaining in the current implementation of our VLC system for indoor localization. The first one is to set exposure time manually, in order to reduce the overexposure of the white bars in the decoding line as mentioned in Section III-C.

The second challenge is to stabilize the location accuracy in Z direction. The current method for calculating the Z direction depends on the locations of the blobs with respect to the middle of the image (i.e. the a, b, Z_f coordinate system). Whenever one or more of the blobs are positioned in such a way that the a values will be negative, the Z coordinate will also be negative. This can be solved by accounting for the blobs being positioned in the negative space of the a, b, Z_f coordinate system.

The third challenge is to make the app function correctly when holding the smartphone at an angle with respect to the lamps. Currently an angle will shift the blobs on the screen accordingly, which in the Angle of Arrival method would result in a shift in the absolute position of the smartphone. This is unwanted behavior, since a rotation of the camera should not cause the absolute position to change. There are two ways to take on this challenge. The first one is to use the smartphone gyroscopic sensor to determine the angle at which the phone is being held by the user. In the Angle of Arrival method an adjustment can be made to account for the angle when determining the absolute position. Another solution could be to use the fact that the blobs on the screen will change their shape when the user hold the camera at an angle. There are

two parameters that can be used for this purpose: the size and the ellipticity of the blobs. This method still needs to be investigated, but could potentially be a supporting measure for the first-mentioned solution.

VI. Using the VLC system

In this section a brief overview on how to set up the lamps and use the applications for indoor localization shall be outlined.

A. Setting up the lamps

The lamps are driven by Arduino micro controllers that encode the ids in the lamps. In order to change the package or ids that are sent, one needs to adjust this in the .INO files that program the boards. Also the frequency can be changed to increase or reduce time of a single on or off event of the lamp. These files can be found in the public repository of the authors. The current implementation of the application does not allow for another preamble than 01110. This can however changed by modifying the JNI source code.

B. Using the app

After executing the app, the user needs to lock the camera's exposure time by holding the camera close to the lamp and pressing the "LockExp" button. After that, the user should step back to get at least three blobs on the screen and press the "Process" button to start the localization process. The source code has been well-documented and one can find more details on the workings of the localization process in the header files.

References

[1] Y.-S. Kuo, P. Pannuto, K.-J. Hsiao, and P. Dutta, "Luxapose: Indoor positioning with mobile phones and visible light," in *Proceedings of the 20th annual international conference on Mobile computing and networking*. ACM, 2014, pp. 447–458.

[2] M. Vasilakis, "Dynalight: A dynamic visible light communication link for smartphones," Ph.D. dissertation, TU Delft, Delft University of Technology, 2015.