# CONTENT ADAPTIVE ENCODING METHOD FOR HIGH FRAME RATE SCREEN-CAMERA COMMUNICATION

BY YAQIN TANG

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Professor Marco Gruteser

and approved by

New Brunswick, New Jersey

January, 2016

## ABSTRACT OF THE THESIS

# Content Adaptive Encoding Method for High Frame Rate Screen-Camera Communication

**by Yaqin Tang**

**Thesis Director: Professor Marco Gruteser**

As both screen displaying speed and camera capturing speed has been improved significantly over recent years, there has been increasing interest in using this technique to explore new ways of visual communication yet remain unobtrusive to human eyes.

This thesis proposes a content adaptive system that can communicate between a high speed screen and a high speed camera while hiding information from human perception. The novel content adaptive embedding approach for this visual light communication system is implemented by applying texture range selection and edge avoiding on a checkerboard pattern for the original image to embed more information in image regions that are suitable for flicker-free communication. At the receiver, the embedding regions are identified by tracking temporal signal amplitude alteration. With such techniques, the system can achieve near zero flicker perception while successfully communicating a large volume of information between the screen and camera. To evaluate the system, we test 10 static and dynamic color videos displaying at 120fps and place the camera

70cm away in a stand while capturing the video at 240fps. Results show that all the 10 videos can achieve near zero flicker and provide an average capacity of 16.52kbps and with an average bit error rate of 5%.

## Acknowledgements

I would like to express my sincere gratitude towards those who provide incountable support, guidance and help for me. It is with all their love and support that leads me to reach this successful destination.

Firstly, I would like to thank my advisor Professor Marco Gruteser for all his support and instructions. I cherish the time with him during our weekly meeting. He's always the one who trains my critical thinking ability and gives me suggestions in the right direction. And he is always the one who gives me timely advice and support whenever needed. His help and suggestions illuminate me and open my eyes. This work cannot be finished without his insightful comments. My sincere gratitude go first to him.

Secondly, I would like to give my thank to Prof. Kristin Dana, Prof. Narayan Mandayam and Dr. Wenjun Hu. They are the co-advisors for this project. I really appreciate their time and advice every week to give me insightful comments from different angles regarding my progress. Without their help, my work won't be so complete.

Last but not least, I would like to give my thank to my family and friends for their endless love. My parents are always the one to share happiness and sorrow. They stand with me no matter what happens. Their encouragement let me grow to be strong. Also, I would like to mention my gratitude to everyone in WINLAB and ECE department. They are such helpful and friendly that makes me feel like home.

This work is supported by National Science Foundation (NSF) under grant CNS-1065463, and teaching assistantship from ECE Department at Rutgers University.

# Table of Contents

**Abstract** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ii

**Acknowledgements** . . . . . . . . . . . . . . . . . . . . . . . . . . . . iv

**List of Figures** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . vii

**1. Introduction** . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1

1.1. Motivation and research objectives . . . . . . . . . . . . . . . . . . . . 1

1.2. Background . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2

1.3. Organization of the thesis . . . . . . . . . . . . . . . . . . . . . . . 4

**2. Related Works and Flicker Perception Factors** . . . . . . . . . . . . . 5

2.1. Related works . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

2.2. Factors that affect flicker perception . . . . . . . . . . . . . . . . . . 6

2.2.1. Frame rate . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

2.2.2. Modulation amplitude . . . . . . . . . . . . . . . . . . . . . . 8

2.2.3. Image content . . . . . . . . . . . . . . . . . . . . . . . . . . 8

2.2.4. Edge effect . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

2.2.5. Viewer's field of view . . . . . . . . . . . . . . . . . . . . . . 12

2.3. Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12

**3. Overview of System Design** . . . . . . . . . . . . . . . . . . . . . . 13

3.1. Content adaptive encoding method . . . . . . . . . . . . . . . . . . . 13

3.1.1. Temporal dimension embedding . . . . . . . . . . . . . . . . . . 14

3.1.2. Spatial dimension embedding . . . . . . . . . . . . . . . . . . . . 17

3.2. Synchronization issue . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21

3.3. Detect encoded pattern by the receiver . . . . . . . . . . . . . . . . . . 24

3.3.1. Algorithm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24

3.3.2. Challenges and accuracy improvement . . . . . . . . . . . . . . 25

3.4. Decoding Scheme . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28

3.4.1. Captured frame pre-processing . . . . . . . . . . . . . . . . . . 28

3.4.2. Decoding algorithm . . . . . . . . . . . . . . . . . . . . . . . 29

**4. Prototype Implementation** . . . . . . . . . . . . . . . . . . . . . . 34

4.1. System platform . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34

4.2. Experiment video selection . . . . . . . . . . . . . . . . . . . . . . 36

4.3. Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37

**5. System Performance Evaluation** . . . . . . . . . . . . . . . . . . . 40

5.1. Evaluation metrics . . . . . . . . . . . . . . . . . . . . . . . . . . . 40

5.2. System evaluation . . . . . . . . . . . . . . . . . . . . . . . . . . . 41

5.2.1. Encoded pattern detection evaluation . . . . . . . . . . . . . . . 42

5.2.2. Decoding algorithm evaluation . . . . . . . . . . . . . . . . . . 43

5.3. Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47

**6. Conclusions** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48

# List of Figures

2.1. Siganl Amplitude Experiment: the grayscale block on the left has a fixed intensity value while the right block changes color altenatively. Experiments show that $\alpha = 2$ and $\beta = 3$ is the largest pair tolerating flicker-free observation. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

2.2. Performance of flicker perception for different video samples: we divide the level of flicker into five categories, and the bar along the y-axis counts for the number of test videos that fall into its corresponding categories. It's clear that encoding into texture region has less amount of flicker than vice versa. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

3.1. Manchester code: bit 1 is represented by 10 and 0 is represented by 01 in binary. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

3.2. Temporal encoding method: The original video is 30fps and we duplicate each original frame into four frames to be displayed at 120fps on screen. The actual message is 60fps and is represented by Manchester code with 1 to be BRIGHTER/DARKER and 0 to be DARKER/BRIGHTER. The brighter and darker refers to pixel intensity value increase by 2 or decrease by 3 separately. . . . . . . . . . . . . . . . . . . . . . . . . . 17

3.3. Texture range selection algorithm: the texture value of a specific pixel is calculated by the difference of its maximum and minimum pixel intensity values surround the 3-by-3 neighborhood matrix. . . . . . . . . . . . . . 19

3.4. Synchronization issue between display and camera capturing: assume the first sequence is the video display at 120 fps, then the second sequence serves as an example of the synchronization issue if capturing at 120 fps. To solve this problem, if we double the capturing frame rate, we can ensure with a set of clean frames mixed with tainted frames. . . . . . . . 22

3.5. Histogram-based clean and tainted frame analysis: by separating even and odd frame sets and calculating the histogram of it difference value, we get the one with larger peak distance in the figure as the clean frame set. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23

3.6. Method to detect encoded pattern: from the clean frame set, get the difference matrix by subtracting two consecutive frames. Based on Manchester code, the encoded blocks should have a large difference while the non-encoded blocks' difference is zero. . . . . . . . . . . . . . . . . . . . . 26

3.7. Camera sensitivity curve: iphone6 camera's sensitivity range is around 25 to 240, for intensity value out of the range, the camera cannot capture the brightness as accurately. . . . . . . . . . . . . . . . . . . . . . . . . 28

3.8. Content adaptive encoding method: the original image can be color or grayscale image, being calculated the texture range value, which represents the level of texture for the original image, and will be placing a checkerboard on top of it as spatial embedding. In block-wise analyis, for each block, if more than 50% of pixel is good pixel and its standard deviation is smaller than 15, then mark this block as block should be encode. Do the same analysis in temporal domain; alternate brightness based on real message. . . . . . . . . . . . . . . . . . . . . . . . . . . 32

3.9. Temporal signal amplitude alteration decoding method: the captured video is extracted and classified for clean and tainted frames based on histogram analysis. For image pre-processing, we need warp the frame into correct perspective and crop only the image we want. Based on the encoded checkerboard pattern known, the camera side can do real message decoding by tracking the signal amplitude alteration, i.e. average brightness change, in temporal domain. If the trend is from brighter to darker block, then decode as bit 1, otherwise decode as bit 0. . . . . . . 33

4.1. Experiment settings: the monitor serves as the transmitter platform to display the encoded video at 120 fps using glVideoPlayer. Iphone6 is in stand hold and its default camera application is set to be in Slo-Mo mode with auto-exposure fixed. The distance between the screen and camera is 70cm for the image to fill in the camera. . . . . . . . . . . . . . . . . . 36

5.1. Error rate for encoded pattern detection evaluation: experiments result show the error rate of all 10 test videos are in the range of 2% to 6%. . 42

5.2. System performance for static videos: in this case, the system can achieve near zero flicker with an average goodput to be 15.16 kbps. And maintain the bit error rate to be within 7%. . . . . . . . . . . . . . . . . . . . . . 45

5.3. System performance for dynamic videos: in this case, the system can achieve no flicker observed for all test videos and an average goodput to be 16.52 kbps. The bit error rate is within 20%. Dynamic scene video contains plentiful information, so there's a trade-off between flicker perception and bit error rate. . . . . . . . . . . . . . . . . . . . . . . . 46

# Chapter 1

# Introduction

## 1.1 Motivation and research objectives

Screen has been widely used in our daily life to convey rich information visually. Meanwhile, with the fast progress of smartphones being capable of capturing videos at high speed, screen-camera communication emerged in diverse applications and has been a hot research topic over recent years. There are works focusing on improving the visual experience of the screen side as well as efforts to increase system capacity for the communication channel. However, there is no work that can achieve a satisfactory results on both evaluation metrics at the same time. Trade-offs exists on the design of the screen-camera communication system, and in this thesis, our objectives is to explore factors that contribute viewing experience on the screen and balance the trade-off between video quality and communication efficiency.

In our efforts of trying to understand those effects and find out a balance for the system design. We conduct comprehensive experiments on different factors that might affect flicker perception, i.e. screen video viewing experience and find out some aspects would give us useful insights on the design. The first one is the video displaying frame rate on screen. Research results shown that higher frame rate would result in less flicker. Image texture as well as image contour are also important factors to consider. As a short state, the more complex the image content is, the easier it is to hide information.

Based on the insightful understanding of the affecting factors, in this thesis we first propose a content adaptive encoding algorithm to produce flicker-free screen video viewing experience. This technique is realized by image analysis of video frames and we aim at encoding those regions of the image that is good to encode. In this thesis, we use image pixel intensity range value selection method to set a reasonable threshold as a metric to determine the region of interest for each video. Secondly, our encoding method is both spatially and temporally, which helps to largely improve system capacity. Finally, the screen-camera communication system is robust enough with low bit error rate on the receiver side by using temporal signal amplitude difference comparison. Since the screen side's information is encoded into different regions based on different image content, we explore ways of automatically identifying the encoded patterns on the camera side and try to improve the accuracy in ways such as noise reduction, communication environment, camera calibration and etc.

In summary, to address the limitations and challenges of existing works, our research objective is to explore psychovisual factors that cause flicker perception and propose a novel flicker-free screen-camera communication systems modulated both spatially and temporally so that it's robust enough to achieve reliable communications with high throughput and high accuracy.

## 1.2 Background

Research works on screen-camera communication start from direct visible codes and marks on screen. PixNet [1] proposes a technique to modulate high-throughput 2D barcode and optimize high-capacity LCD-camera communication by using the orthogonal frequency-division multiplexing (OFDM) digital multi-carrier modulation method.

COBRA [2] improves the barcode system with color in a real-time phone to phone communication scenario, and aimed to improve the decoding accuracy caused by blurring during motion activities. The static barcode technique is widely used nowadays. For example, all the products in the supermarket are marked with barcode to represent their property and information. Another common scenario is the barcode on the mail serving as tracking information. Furthermore, another breakthrough for visual codes screen-camera communication is the emerging of steganography or watermarking technique [3] which is realized by embedding a spatial code (e.g., QR-code [4]) in the display image yet still visible to human eyes, such as the name card on Wechat, one of the most popular chatting applications. In this case, people just need to scan the name card in QR-code format and can get personal information immediately.

On the other hand, however, more recent studies tend to focus on establishing invisible screen-camera communication system with the emerging of the term *Visual MIMO* [5], where the spatial light spectrum serves as a multi-channel in the communication system and tend to provide with a real-time dynamic and invisible screen-camera communication system. Representative works are VR Codes [6] takes advantage of the fact that only mixed colors are perceivable to huamn eyes, they uses high-frequency red and green light to communicate with cameras on smartphone and maintains the communication process invisible to human eyes. InFrame++ proposes a concurrent, dual-mode, full-frame communication system by multiplexing data onto videos of complementary frame composition, hierarchical frame structure, and CDMA-like modulation [7]. HiLight encodes data into pixel translucency change on top of screen content by varying the color between black and white across different grids of the communication layer based on the content area color [8]. These recent works focused interest and efforts for building an unobtrusive screen-camera communicaion system with different methods for embedding invisible codes hidden from contents being displayed on the screen.

## 1.3 Organization of the thesis

The rest of the thesis is organized as follows.

In chapter 2, related works on screen-camera communication system is introduces and various factors, such as frame rate, image content, image contour and angle of view, that tend to cause flickers are explored.

In Chapter 3, the overall system design is proposed. The contend adaptive encoding method in the screen transmitter side are extensively discussed and the techniques for the camera to determine the original encoded pattern and the robust decoding scheme on the receiver side are explained in detail.

In Chapter 4, the experiments settings are given and the proposed system prototype implementation and how the experiments are conducted are fully described.

In Chapter 5, the metrics to evaluate system performance are introduced and experiments results in real video scenarios are given and explained.

Finally, chapter 6 concludes the thesis and discusses promising applications from this work and discusses some remaining issues left as future work and beyond.

# Chapter 2

# Related Works and Flicker Perception Factors

## 2.1 Related works

Screen-camera communication is initially established with visible codes like barcode technique and then improves and progresses to invisible codes embedded. In addition, later works for Video watermarking and steganography also make it imperceptible to encode messages into images and videos [9, 10]. Moreover, a more recent work, ImplicitCode [11], aims at boosting the communication capacity with a joint efforts of existing techniques of HiLight, VRCodes and Inframe. Also, there are other extensions of works include solving the frame synchroniztion issue [12], extending the supported range of operation [13], and an improvement of the robustness of the system and its communication throughtput [14]. Following this trend, it's obvious to see that researchers are trying to improve the screen-camera communication system in two aspects: visual experience on the screen side and communication performance on the camera side. Thus it's worth an effort to explore both aspects to improve the performance of screen-camera communication.

Prior knowledge and researches show that ficker perception has been well studied over the years on a single light source, where there is conclusion states that direct visual flicker perception is negligible at frequencies of 100 Hz or higher in [15–17]. Since the experiments are conducted on single light source and in our scenario we are using screen as the displaying platform, it's an important work to address for some psychovisual factors that might affect flikcer perception and as a result, leads to a better encoding scheme direction. Also, the capacitance of existing works for both visible and invisible codes are very limited. Even though there are studies of communications using temporally modulated light source. The transmitter is a light source modulated at high temporal frequencies. The specialized sensors in this system can be photo-diodes or high speed camera [18]. This idea might somehow improve the capacitance and from another side, though the photo-diodes can receive the transmitted signals, they cannot simultaneously capture images for human consumption, which leads us to think of another solution to fully exploit high speed screen-camera communications.

## 2.2 Factors that affect flicker perception

The term flicker is a perceptual attribute in the psychophysics of vision, normally defined for light source or screen displays, seen as an apparent or obvious fluctuation and change in the brightness of the displaying surface [19]. Since flicker perception are very annoying and dizzy to human eyes, which is not a positive experience. It is of high importance to learn the mechanisms behind it and find a way to best reduce flicker. This general idea has been applied to all kinds of areas such as TV, camera, home lightening and etc. However, prior psychovisual studies are mainly focused on how to reduce flicker in a single light source. Illuminated by previous work and based on the fact that screen display are actually a collection of light sources, we will follow the idea on previous research work on various effects that may contribute to perceivable flicker, such as the frame rate, image content, image contour and the viewer's field of view and etc. And aims to expand these findings to screen display.

A good communication system should be able to carry information, and the way how visual light communication system embed information is to use markers, images or videos as carrier and embed information in. However, this will introduce brightness changes on the screen . Therefore, embedded screen-camera communication can naturally generate flicker. From this motivation, we explore how to balance the conflicting goals of embedding information and avoiding flicker, thus providing guidelines for encoding method of the screen-camera communication system. Note that the following experiments are conducted in the same experimental environment discussed in Chapter 4 and the level of flicker is assessed by students in this research group, since there's no standard objective metrics to evaluate flicker perception in existing literatures.

### 2.2.1 Frame rate

It has long been known that flicker perception is prominent for luminance fluctuations below 100 Hz [15, 16]. This was determined with 2 deg circular disk on a uniform background of the same time average luminance. Although this frequency threshold was determined using a single light source, it is still applicable if we consider the modern display as a collection of LED light sources.

In our case, the fluctuation is caused by switching between bits at the same position of the video across frames. Since such bit streams are random, we are constrained by the largest differences between the codewords, the available display refresh rate (up to 144 Hz), and the camera capture rate (up to 240 fps). Given the latter two constraints, we can expect to display at 120 fps. The maximum codeword distance can then be determined accordingly.

### 2.2.2 Modulation amplitude

The system aims at hiding information both spatially and temporally. Unavoidably, there will be consecutive same modulation method in the random bits stream, thus reducing displaying frequency at the moment, which causes flicker as from conclusions in [15, 16]. To solve this problem, we propose to use Manchester encoding scheme to ensure there's only two same bits at a time, which means the minimum frame rate will be above 60fps. In order to tolerate the occasional half the signal frequency caused by temporal embedding, we conduct experiments by placing two uniform grayscale blocks side by side (Figure 2.1). In each run, the left block has a fixed intensity value $x$, while the right block's color flips between $x + \alpha$ and $x - \beta$ at 120 fps. Across runs, $x$ varies from 0 to 255 at steps of 25. Experiments show that the color deviation without inducing flicker perception is $\alpha = 2$ and $\beta = 3$. In other words, only very slight color differences between adjacent blocks can be tolerated. This suggests very limited scope for encoding bits directly using pixel intensity changes.

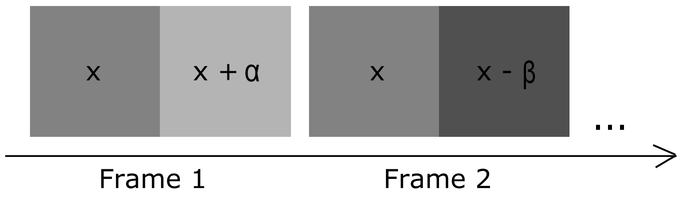


Figure 2.1 Siganl Amplitude Experiment: the grayscale block on the left has a fixed intensity value while the right block changes color altenatively. Experiments show that $\alpha = 2$ and $\beta = 3$ is the largest pair tolerating flicker-free observation.

### 2.2.3 Image content

Images of natural scenes often contain many textured regions that we can use in our coding method. It is well known that human vision is sensitive to even small intensity edges [20, 21] and that texture affects the perception of intensity transitions [22]. As a practical consequence of these perception traits, intensity modifications in smooth regions are more likely to cause flicker than textured regions. To take advantage of this flicker reduction, our method adapts to image content by detecting textured regions and embedding message bits within this space. To qualitatively evaluate the intuition of texture-based embedding, we experiment with 20 videos of varied content. We divide each video frame into smooth and textured regions and embed bits into the smooth regions only, the textured regions only, or all regions to compare the flicker perception. Fig. 2.2 shows that embedding into textured regions exhibits the least amount of flicker.

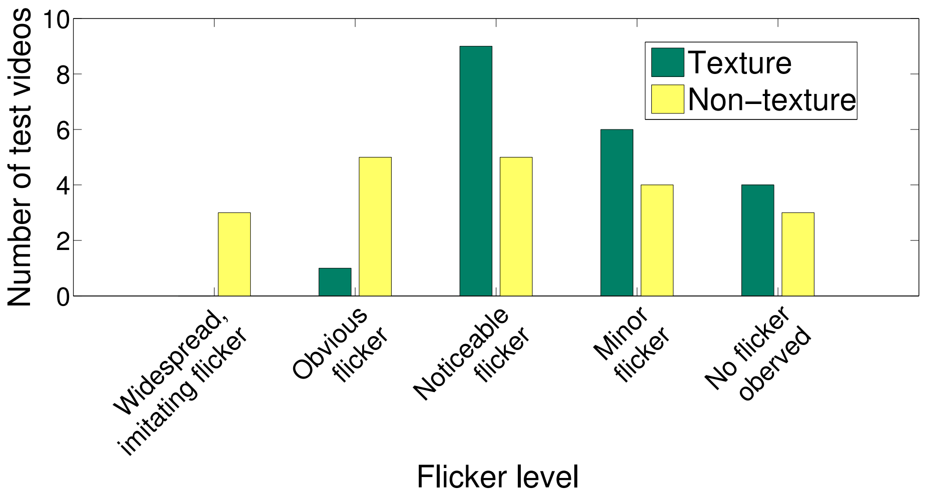


Figure 2.2 Performance of flicker perception for different video samples: we divide the level of flicker into five categories, and the bar along the y-axis counts for the number of test videos that fall into its corresponding categories. It's clear that encoding into texture region has less amount of flicker than vice versa.

Besides image texture, image brightness and image contrast is also a point of interest to explore. Experiments are conducted with the same set of videos in the same settings. For image brightness (our discussion below is in the range of pixel intensity value from

0 to 255), we create videos side by side comparing with encoding the whole image, encoding only dark regions (average brightness smaller than 85), encoding only bright regions (average brightness greater than 170) and encoding mid-tone regions with average brightness in between. We repeat the comparison for 20 videos with different image content and results show that the level of flicker perception is quite similar regarding difference brightness encoding regions. Therefore, we conclude that image brightness is not as a large factor as to affect flicker perception.

For image contrast, it is a visual concept that is determined by the difference in the color and brightness of the object. We measure the contrast of an image by Michelson contrast [23], where it is defined as the following equation and $I_{\max}$, $I_{\min}$ represent the highest and lowest luminance separately.

$$\text{Michelson contrast} = \frac{I_{\max} - I_{\min}}{I_{\max} + I_{\min}} \tag{2.1}$$

In such case, contrast equals zero means the image has no contrast. In our experiments, we analyze the same 20 videos and adjust its contrast range from 0 to 1 (normalized) with a step of 0.1. Table 2.1 is an example of the image after flipping its contrast from the original image. We do this for all the 20 videos available and compare the flicker perception for each video. We conclude from the results that image contrast is not a major factor to influence flicker perception but might help the video fall into the camera sensitivity range, thus improving accuracy (We will discuss this in Chapter 3).

### 2.2.4 Edge effect

From the grayscale block experiment illustrated in Figure 2.1, we also observe that when two neighboring blocks modulated with the *same phase* i.e. both the left and right block in one frame is x+$\alpha$ would cause no flicker at the edge, while those modulated

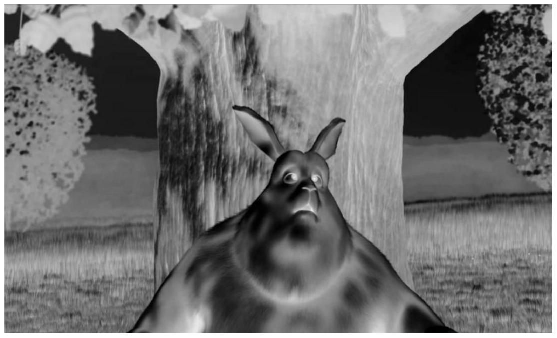
| | |
|---|---|
| |  |
| Original image | Image after change of contrast |

Table 2.1 Comparison of original image and image after change of contrast: the left side image is the original image and the right side image is the one after flipping its contrast.

with *different phases* i.e. left block is x+$\alpha$ while right block is x-$\beta$ would introduce flicker. Furthermore, if separate the two blocks with larger distance, this effect will be minimized. In trying to understand this edge effect, we find literature stating that this is the so-called *saccades effect*, which are defined as rapid, ballistic movements of the eyes that abruptly change the point of fixation. The authors in [24] discusses edges introduced by a white and black part each stands in half of a frame, and their results demonstrate that even at a switching frequency of up to 500Hz, we can still observe flicker by alternating these two halves rapidly. It has also been observed that when the modulated light source contains a spatial frequency edge, human subjects can still see flicker with frequency over 200 Hz.

Since it is common to use a block of pixels to encode bit stream, we also encounter edges between adjacent blocks of different bits. When the two neighboring blocks are modulated with *different phases*, flicker is noticeable. However, based on the observation, we can still try to separate the blocks with some distance or apply edge blending technique to reduce or minimize the flicker caused by edge effect.

### 2.2.5 Viewer's field of view

In the course of experiments, we also observe that the level of flicker perception depends on the size of the encoded regions in the video and the distance of the viewer from the video displayed. We capture both effects with a single metric, the size of the "viewer's field of view". To measure this size, we use a square block of different sizes for encoding without changing other parameters and view the video from difference distances. Results show the smaller area fell into viewer's retina, i.e., the smaller block size or further distance, the less flicker the viewer perceives. This suggests using only small code blocks for encoding and avoiding parts of the image scene that might attract attention.

## 2.3 Conclusion

Chapter 2 aims to explore factors that affect flicker perception. From the discussion, we know that higher frame rate tends to produce less flicker, resulting in choosing Manchester code as our basic encoding scheme. The advantage of Manchester code is to ensure displaying frequency to be above 60fps. To minimize the flicker caused by the 60 fps part, we conduct experiment to find out the most suitable brightness alteration pair is to increase by 2 and decrease by 3. To further improve the capacity of the system, we introduce checkerboard pattern on the spatial domain. In this case, image texture is another important factor that should be taken into account. Since we know that if the image has more complex content, i.e. textured, there would be less flicker, that's the reason to choose more textured region to encode.

# Chapter 3

# Overview of System Design

## 3.1 Content adaptive encoding method

As a restatement of our research objective, we are trying to propose a robust flicker-free screen-camera communication system with high capacitance and low bit error rate. Therefore, the above explorations give a good guidance of where to strive for a flicker-free screen encoding method. The first two points, frame rate and modulation amplitude, suggest opportunities for modulating bits in a small range of alteration if being applied properly. And image texture analysis gives us a good hint to explore methods of reducing flicker, while the image brightness, image contrast and the field of view cannot be leveraged easily since they either don't seem to have direct impact on flicker perception or because the encoder side or the human observer has no control. However, the former factors are orthogonal, indicating possible solutions for combining these factors to give the most satisfied scheme. To clarify, frame rate is a temporal property of the video, whereas modulation amplitude and image texture mostly affect the spatial domain of the video. Based on these insights, we design our content adaptive flicker-free encoding method to achieve high capacity at negligible flicker, which compromises the constraints and limitations of previous work like HiLight [8] and Inframe++ [7] where capacitance and flicker perception cannot be achieved satisfactory at the same time, thus distinguishing us from previous work with better system performance.

### 3.1.1 Temporal dimension embedding

Since we are displaying videos on the screen as our carrier of information, and videos are a collection of frames being displayed consecutively as time goes, we are trying to embed different code into every frame so that the system can achieve highest capacity as possible, that is what we called temporal dimension embedding. In trying to get our system robust enough for any scenario, we put our analysis for random bits assigned in the temporal domain as a more general case.

The first thing to decide from the insights is the frame rate. From what we know from previous work, we will not prefer frame rate to be below 100 Hz, as it will cause obvious flicker with luminance fluctuations. Therefore, based on the available display refresh rate to be up to 144 Hz, we choose to display our video at 120 fps, which is double the refresh rate of what we usually have to be 60 Hz in our daily life. The reason why we not go with the highest frame rate is based on the consideration of the receiver side, where the maximum capturing rate for smartphone is 240 fps up to date. Therefore, if we want to capturing the video at double the frame rate, we are limited to 120 fps at the transmitter side. Based on all these facts and considerations, we set our video displaying frame rate to be 120 fps. However, it is not the whole story as if we are ensured to get a flicker-free video once we display it at high refresh rate. We can still observe flicker with random bits being assigned to the video and this phenomenon is caused by the random bits, where there might be cases with consecutive same bits leading to a lower frame rate. For example, For a normal ideal case, we consider the video is always maintaining 120 fps when the bits stream is alternating every other bits. However, for random bits stream, we cannot ensure the bits sequence, there would be cases that two or more same consecutive bits exist side by side, at this point, it will reduce the displaying frequency and cause obvious flicker in return.

Illuminated by a traditional communication encoding scheme called Manchester code, where code 1 is represented by 10 and 0 is represented by 01 in binary code system (Figure 3.1), we can keep the frequency components of the encoded video signal above 60 Hz, where in the worst case, there's only two consecutive bits.

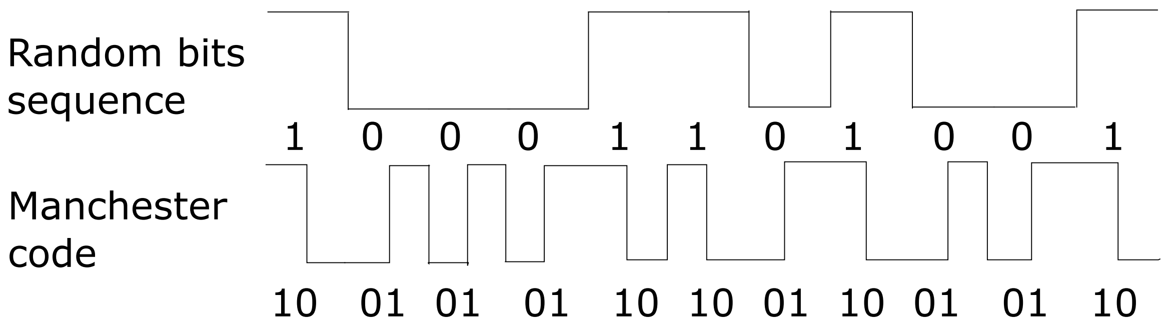


Figure 3.1 Manchester code: bit 1 is represented by 10 and 0 is represented by 01 in binary.

Taking the advantage of Manchester code ensures a transition on every bit, it generates less low frequency components in the modulated signal when multiple consecutive bits are identical, even though we compromise half of the system capacity. It is worth the effort because flicker perception is such a key factor for this high frame rate screen-camera communication system.

Once the frame rate issue is settled, we need to fix the low frequency displaying part producing noticeable flickers. One of the key factors to consider is to adjust the modulation amplitude. Before we go deep into this question, we need to clarify how the 0, 1 bits in binary system work in our case. Technically, these bits are another name of image brightness alteration, where we would define 0 as slightly lower the original pixel intensity value and 1 as adjust it to a little bit higher intensity value. The brightness adjustment and fast alternation is the source of causing flicker perceptions, whereas we need to come up with a largest number we can reach to maintain flicker perception to human eyes to the minimum.

To find out the best modulation amplitude, we have experiments designed as shown in Figure 2.1. The left block is a fixed pixel intensity value evaluated from 0 to 255 with a step of 25 and the right block is alternating between a brighter block $x+\alpha$ and a darker block $x-\beta$. We evaluate $\alpha$ and $\beta$ for values from 1 to 10 and find out the best pair with the least flicker perception is $\alpha = 2$ and $\beta = 3$. Therefore, with the best modulation signal amplitude, the random bits sequence generated is combined with the sequence of carrier image frames. For the carrier pixel values with bit 1, the intensity values will be increased by $\alpha$, which makes these pixels brighter. Similarly, for the carrier pixel values with bit 0, the intensity values will be decreased by $\beta$, which makes these pixels darker. And note that these pixel intensity value changes are applied to the Y channel in each frame that contains Y, U and V channels.

As a conclusion for the temporal domain embedding, we use Manchester code to carry our randomly generated information, which will expose the system to display at 60 Hz in some cases and we try to minimize the flicker caused by 60 Hz display by finding the most suitable modulation amplitude of alternating the original pixel intensity value with either increase by 2 or decrease by 3 depending on the bits sequence. As from the implementation side, due to the fact that the videos available in standard library are only 30 fps no matter the video is static or motion, we need to duplicate each frame in the original video as four frames in the video that will be displayed on our 120 fps screen. Therefore, these four frames are containing exactly the same image content in both static and motion videos. We use these four frames to embed two bits of information. That is, the first two frames in these four frame represents 1 bit of information (since we are using Manchester code, so we need two frames to represent one bit), and the following two frames for another bit and so on. The process described above can be further illustrated in Figure 3.2.

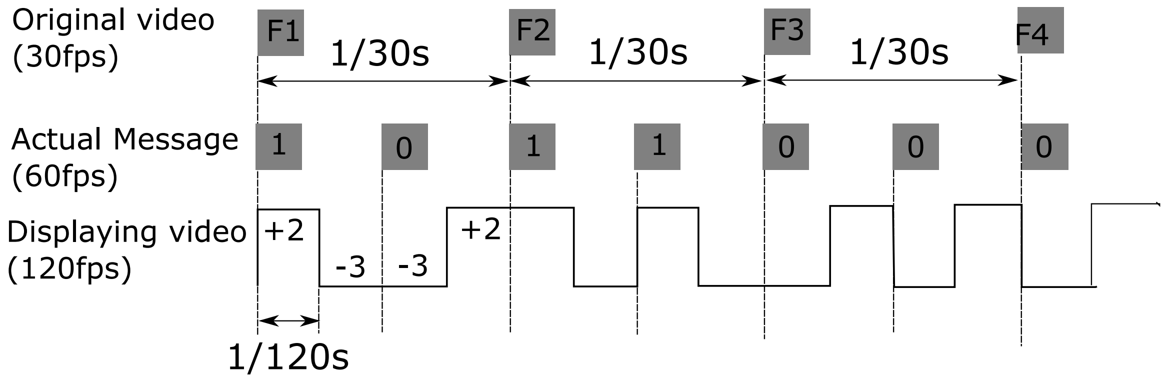


Figure 3.2 Temporal encoding method: The original video is 30fps and we duplicate each original frame into four frames to be displayed at 120fps on screen. The actual message is 60fps and is represented by Manchester code with 1 to be BRIGHTER/DARKER and 0 to be DARKER/BRIGHTER. The brighter and darker refers to pixel intensity value increase by 2 or decrease by 3 separately.

### 3.1.2 Spatial dimension embedding

In order to further increase the capacity of the system, we propose to place a checkerboard pattern on top of each frame, from which the throughput of the system will be increased significantly depending on the size of the checkerboard. We conducted experiment with difference checkerboard size,and find out that with 32×32 pixel size, we can achieve best results in terms of flicker and decoding accuracy. However, based on the flicker perception exploration, we know that image texture and edge effect will have an influence on flicker, and the introduction of checkerboard is also another source of edge besides the image contour. Therefore, image texture and edge effect are the two challenges for spatial domain embedding design.

Image texture analysis has been studied over the years. Some researchers are looking in a small neighborhood through an image with local brightness variations from pixel to pixel [25]. And there are others looking at a different perspective stating that texture can be described as an attribute representing the spatial arrangement of the gray levels of the pixels in a region of a digital image [26]. Traditionally, there are three ways of analyzing image texture, that is, structural approach, Fourier approach and statistical approach. The structural approach to describe texture is defining texture as a set of texture elements or texon occurring in some regular or repeated pattern, like sand on the beach or bricks on the wall. In trying to understand this texton analysis, we apply an algorithm with a training stage and a classification stage. The training stage is to use a large amount of raw data to build a model with different textures. For example, to distinguish leaves, sand or cars. Then use this model to do classification for the images we want to embed. However, there are too many varieties of texture in the universe and we don't have such a large database to train a comprehensive model, which is time consuming and resource limited. So this structural approach is not applicable in our situation. The second approach, Fourier approach, is by calculating the power of the image and classify fine, regular, and directional texture based on its different frequency band. However, this approach is not so commonly used. Last but not least, the statistic approach is to characterize texture with grayscale intensities alone, which is applicable to all types of images and is computationally efficient. As a result, we choose this statistical approach to analyze our image texture.

In this thesis, we call the statistical texture analysis approach as *texture range selection* method. The way we calculate the texture of an image is illustrated in Figure 3.3. We know that an image is a matrix containing a collection of pixel intensity values, and the texture of the image is defined as the local variability of the intensity values of pixels. That is, as shown in Figure 3.3, we consider a 3-by-3 neighborhood for a pixel, say 9 in the figure. It's corresponding texture range value is evaluated by the difference of the maximum and minimum value in the neighborhood matrix, and in this case, the texture range value for pixel marked as 9 is 12. And if apply this algorithm in the whole image, we have a complete texture value matrix. From the local variety, we know that in areas with smooth texture, the range of values in the neighborhood around a pixel will be a small value and in areas of rough texture, the range will be larger.

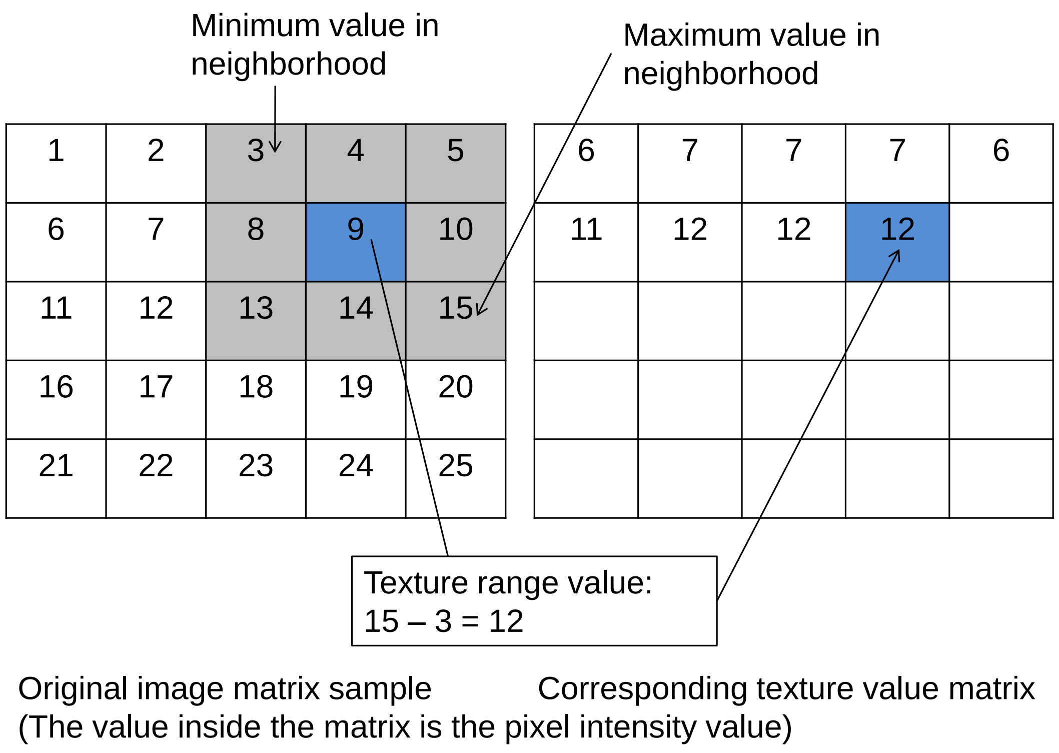

Minimum value in neighborhood
Maximum value in neighborhood
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
6
7
7
7
6
11
12
12
12
Texture range value:
15 – 3 = 12
Original image matrix sample
Corresponding texture value matrix
(The value inside the matrix is the pixel intensity value)

Figure 3.3 Texture range selection algorithm: the texture value of a specific pixel is calculated by the difference of its maximum and minimum pixel intensity values surround the 3-by-3 neighborhood matrix.

Table 3.1 gives us a visualizing perspective of the transformation from original image to texture value image. We can see that the texture value image calculated from texture range algorithm depicts the variety of the pixel intensity value pretty accurately, which formed as the foundation for our later analysis. With the texture range value matrix, the next step is to determine a threshold for the level of texture we want to use. To play around with this texture range value from 0 to 20 and look at the areas being

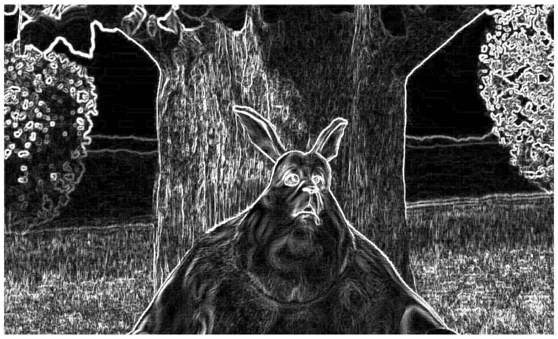
| | |
|---|---|
| |  |
| Original image | Texture value image |

Table 3.1 Comparison of original image and texture value image: the left side image is the original image and the right side image is the one representing the texture range of the image.

covered above the threshold, we conclude that when this value equals to 8 gives best results in terms of flicker perception and image capacity.

On the other hand, from Figure 3.1 we notice that there are sharp edges in the texture value image. Applying the texture range threshold cannot eliminate the sharp edges, since the texture range value is extremely large along those sharp edges. In order to refine the method, we need to come up with a solution to find out the uniform textured region and ensure those regions don't contain large areas of sharp edges. Take the advantage that the sharp edges usually happens at where there's immediate intensity value change surrounding the neighborhood. Therefore, to combine the checkerboard pattern with texture analysis, let's take one block of the checkerboard as an example. The block contains 32×32 pixels represented by its texture range value. If we calculate the deviation of the block, for those blocks with sharp edges, the standard deviation of the block should be relative large. And from our experiment, we conclude that if the standard deviation of the block is larger than 15 pixel intensity value, then mark this block as not good to encode since it is highly possible that the block contains sharp edges. Furthermore, if we define the texture range value of a pixel to be above the threshold as "good pixel", then we also need to mark out the blocks which the number of "good pixels" are less than or equal to half the number of pixels in the block.

To summarize the temporal and spatial dimension embedding scheme discussed above, we illustrate the overall design in Figure 3.8 and in algorithm 1, where as a whole picture, we will do temporal domain embedding with alternating the brightness with increase 2 or decrease 3 in pixel intensity value over time. And the code is represented with Manchester code. On the other hand, for spatial domain embedding, we apply texture range selection and edge avoiding techniques in each frame to decide which block of the checkerboard pattern is good to embed message. And we will only embed those "good" pixels of the marked block. With this content adaptive encoding method, we are expected to have a near zero flicker video being displayed at 120 fps.

## 3.2 Synchronization issue

Considering the video is displaying at 120 fps, we need a camera capable of capturing videos at least the same frame rate on the receiver side. However, if the camera is recording the video at 120 fps, there exists a synchronization issue that if the camera starts capturing not exactly the same time of a specific frame is on display, then the received frames are representing the transition state of two frames. However, Manchester code ensures us that two frames are either alternating from low to high or high to low, the transition state might be any value in the range of 0 to 1. Obviously, these type of what we call *tainted frames* (refer to Figure 3.4) cannot convey the real message accurately. In Figure 3.4, let the white block denotes bit 1 and black block denotes bit 0. We can see that there's possibility if capturing at 120 fps shown in the second sequence. Therefore, capturing at double the frame rate can solve this problem, where we are ensured that even though we have tainted frames, we still have clean frames in the captured sequence. Once we decide to record at 240 fps, our next problem to solve is how to distinguish between *clean* and *tainted* frames.

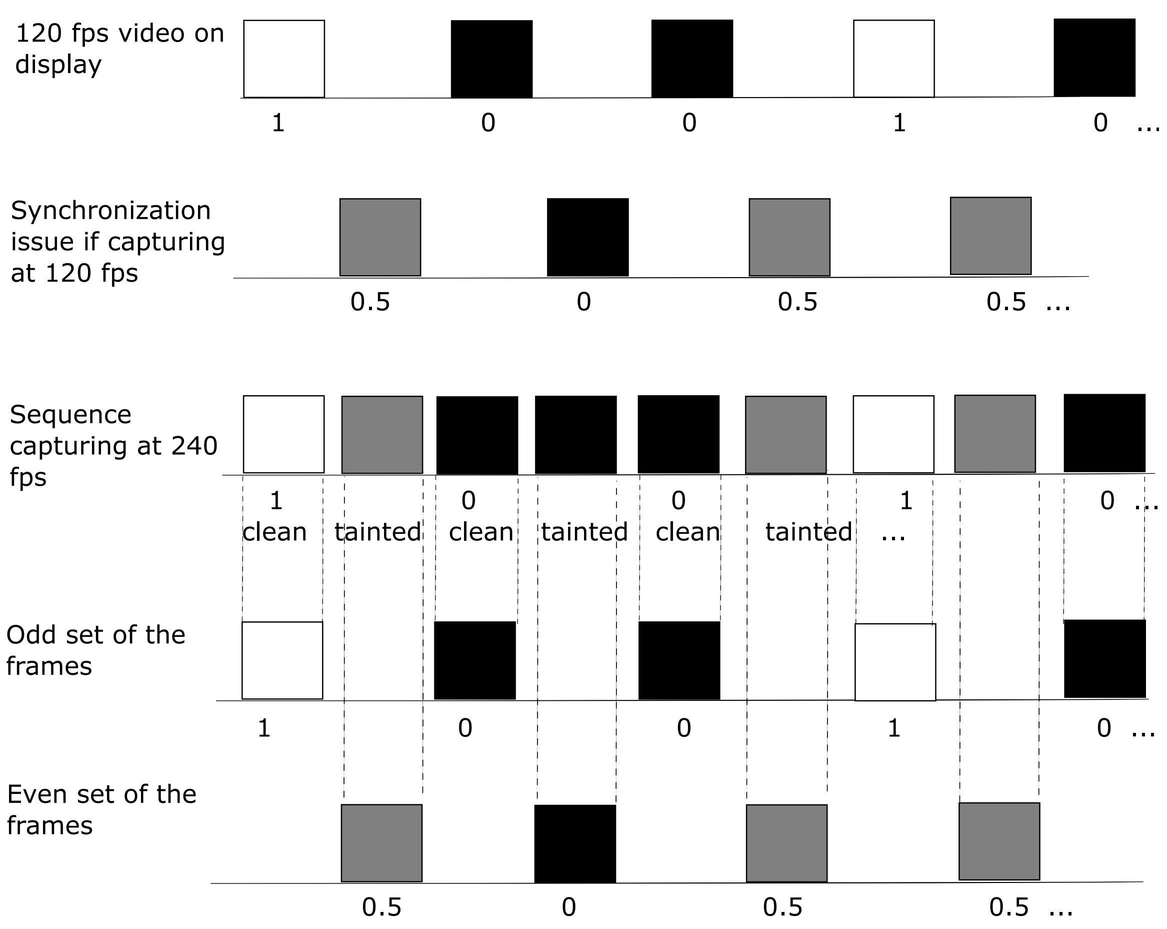


Figure 3.4 Synchronization issue between display and camera capturing: assume the first sequence is the video display at 120 fps, then the second sequence serves as an example of the synchronization issue if capturing at 120 fps. To solve this problem, if we double the capturing frame rate, we can ensure with a set of clean frames mixed with tainted frames.

The algorithm we propose here is *histogram-based* clean and tainted frame analysis. Refer to the third sequence in Figure 3.4, what we first do is to separate odd and even indexed frames into two different set as in the 4th and 5th sequences. Then calculate the difference in two consecutive frames for a large enough number of frames, let's say one thousand. Theoretically, if we get the histogram of the differences between two frames, we are expected to get a larger difference for the clean set of the frames and a smaller difference for the tainted set. Since Manchester code ensures a instinct change for one bit, therefore, the maximum difference is 1 for clean frame set and maximum for tainted frame set is 0.5. Therefore, if we go through one thousand frames in the recorded video, after doing the histogram-based clean and tainted frame analysis, we will be able to get the clean set of frames for decoding. As can be seen from Figure 3.5, once we get the histogram of the difference of two sets, we can easily determine the clean frame set is the one where two peaks on each side of y-axis is larger, and tainted frame set is the one with a smaller peak distance. Also note that the absolute value of the peak difference is around 5 is because we do pixel intensity value change with +2/-3 in the encoding algorithms. Once we get the clean frame set, the next step before doing decoding is to detect the encoded pattern on the original frame.

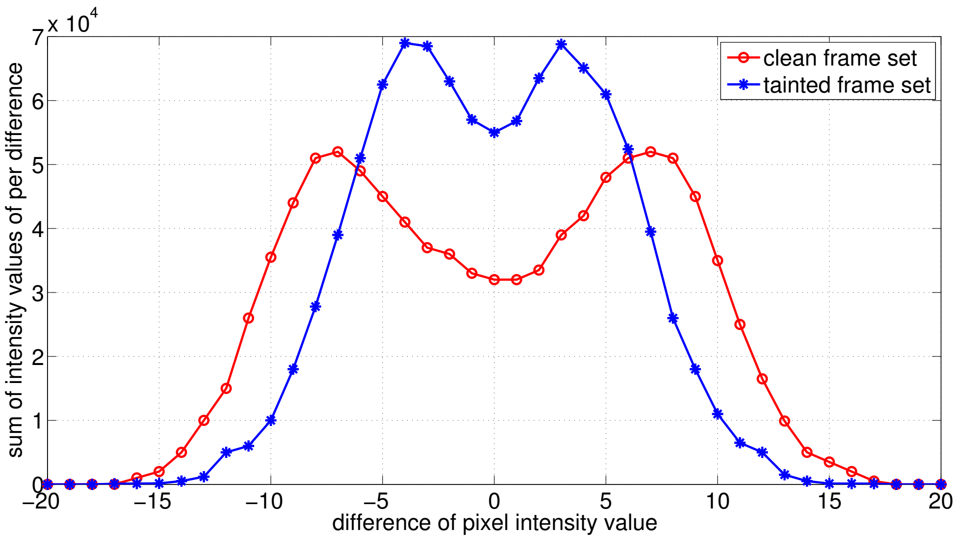


Figure 3.5 Histogram-based clean and tainted frame analysis: by separating even and odd frame sets and calculating the histogram of it difference value, we get the one with larger peak distance in the figure as the clean frame set.

## 3.3 Detect encoded pattern by the receiver

The near zero flicker perception encoding method discussed in previous chapter is a content adaptive embedding, which means different source images will result in different encoded regions. We name those encoded regions as encoded pattern. In order to decode the message embedded, we need to know the encoded patterns of the video, i.e, the checkerboard matrix representing the markers of each block if it's good to encode or not. In the area of network communication, this problem is solved by using a header which contains useful communication information such as address, package size, package type and etc. However, for this screen-camera communication system, real time performance is our priority consideration, so to use a header as the preamble is not a good choice in our case. Hence it is a major challenge for the receiver to know where the information is embedded before it can do actual message decoding. And we define decoding method as for the receiver to find out the exact information hidden in the received frames.

### 3.3.1 Algorithm

The first intuition to determine the encoded pattern is to regard the received frames as the source images and follow the same encoding routine on the those frames. However, this spatial analysis is not as promising as it's ideal assumption. In real experiment scenario, there are some major issues remaining that makes this method impractical. For example, the brightness of the received frame is different from the original frame in the sense that it will be affected by the experiment light condition, camera recording angle, camera saturation and etc. As these factors are almost uncontrollable during the communication process, it's hard to find out the source of error and analyze the texture range value distribution of the received frames and set a suitable threshold exactly the same as what is in the transmitter side. Therefore, this spatial domain analysis intuition is not considered as our priority algorithm.

The algorithm proposed here is to analyze the frames in temporal domain. Since the information is embedded by alternating the brightness of consecutive frames, it's reasonable to detect brightness changes between consecutive frames. Based on the fact that the displaying video at 120fps is get from 30fps original video, the two consecutive frame will have same image content once synchronized, we can get the difference image by subtracting those two consecutive frames and then mark the areas with large difference as encoded regions and near zero difference as non-encoded regions. See Figure 3.6 as an example, the green block marked as -1 is a non-encoded block, therefore, from the difference matrix, we are expected to get a zero difference. However, for the blue blocks, either change from 1 to 0 or 0 to 1, their absolute value of difference is 1, which means a large difference depending on the modulation signal, and 5 pixel intensity value in our case. Therefore, once we get a large amount of data representing the difference in time domain for the same image content, we can achieve this binary classification by using support vector machine, SVM algorithm [27], a computer algorithm to label objects by trained models from examples [28], to get a good model to classify the *encoded and non-encoded* blocks.

### 3.3.2 Challenges and accuracy improvement

We always want an accurate enough encoded pattern before doing real message decoding, since one error here will be accumulated for information lost for that particular block. However, in experiments and real applications, there are always challenges and limitations. Here we will discuss aspects that might affect the accuracy of the detection, such as frame pre-processing, camera settings and its sensitivity range etc.

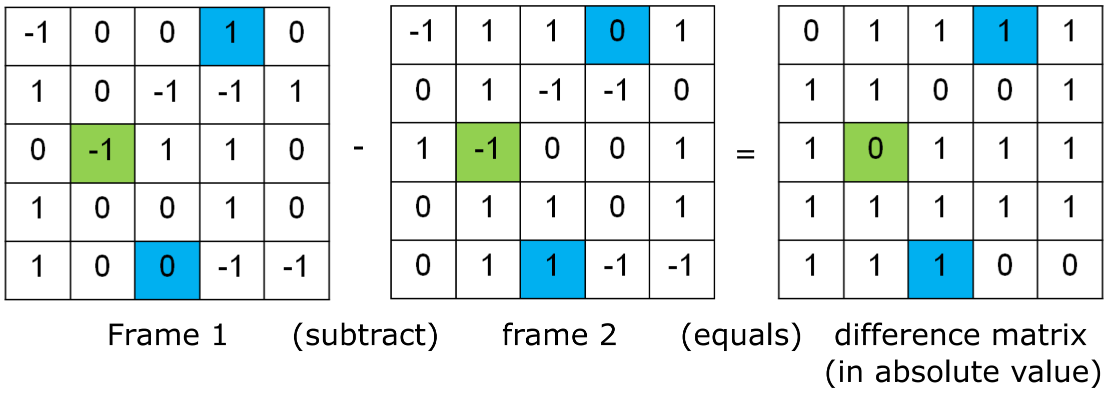


Figure 3.6 Method to detect encoded pattern: from the clean frame set, get the difference matrix by subtracting two consecutive frames. Based on Manchester code, the encoded blocks should have a large difference while the non-encoded blocks' difference is zero.

One of the first observation is that there are lots of noise even at the background region of the recorded video and the light communication is not ideal, which will introduce noise in a uncontrollable way, thus the first step to improve accuracy is to reduce those noise as much as possible. Consider first, there would be some distortion in the process, and second, by analyzing the difference image, we found that it mainly depicts the edges in the image, we would let the received raw image pass a median filter to eliminate these effects. A median filter is a nonlinear filter usually used to remove noise [29] on a image or signal and is implemented by running through the image pixel one by one and replace each entry with the brightness median of neighboring pixels. Furthermore, since the embedding side has to maintain to be flicker free, the modulated signal is small, so we will apply a low-pass filter to remove the large noises. After the received frame has pass the two filters, we can ensure the difference image is quite clean and smooth.

The second step is to calibrate the orientation of the received frames. In the spatial domain, we place a checkerboard pattern on top of it. Even though we know the size of the checkerboard in the system, we still need to resize the received frame and correct its orientation before dividing the checkerboard pattern. Therefore, to solve this problem, we need to warp the frames. Image warping, usually serves as a method to remove optical distortions, is a transformation which maps all positions in one image plane to positions in a second plane [30]. Ideally, the intensity of the warped image is the same as the original image at corresponding points. Consider an image $I_1$, image warping produces a new image $I_2$ such that in Equation 3.1.

$$I_2(T_\theta(x)) = I_1(x) \tag{3.1}$$

To get the warped image $I_2$, we need to define a model for rotation, scaling and translation with four degrees of freedom. Therefore, we have $T_\theta(x)$ with the following relationship shown in Equation 3.2.

$$T_\theta(x) = sMx + [t_x \ t_y]' \tag{3.2}$$

$$M = \begin{bmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{bmatrix}$$

Once we have the smooth and warped frames, we can ideally detect the encoded pattern by tracking temporal amplitude alteration. However, there are two aspects we need to consider during experiment designs. The first one is the experiment light condition. In trying to understand how environment light will affect experiment accuracy, we conduct experiments at indoor conference room with main lights, side lights on and all lights off. Results show that with environment lights off gives best results. And for those lights on, there will be some noise even in the static background scene. This insight gives us a hint that those noises come from environment light should either be removed by filters introduced above or try to eliminate environment light influence.

Furthermore, camera settings and its sensitivity range is also a major factor in detect encoded pattern. Figure 3.7 depicts the iphone6 camera sensitivity curve, where we can get that the camera can only capture pixel intensity value in the range of 25 to 240. For brightness out of the range might suffer from camera saturation.

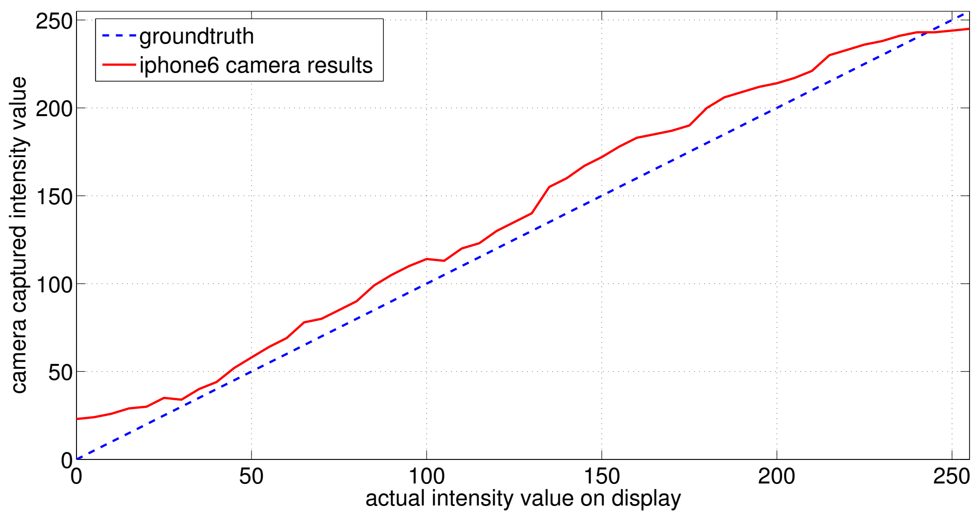


Figure 3.7 Camera sensitivity curve: iphone6 camera's sensitivity range is around 25 to 240, for intensity value out of the range, the camera cannot capture the brightness as accurately.

## 3.4 Decoding Scheme

### 3.4.1 Captured frame pre-processing

In real applications, the receiver side is uncontrollable. Due to difference viewing or capturing angle, image distortion will always exists in the captured videos. This distortion won't affect temporal sequence if we assume the camera is in static stand. However, in spatial domain, distortion makes it very difficult to recover the correct location of those blocks in the checkerboard pattern. Therefore, after we get the clean frame sets, we need to warp all those frames similar to what we refer in Equation 3.1 and 3.2. Once we get all frames warped into correct perspective by the projective transformation method, we can easily divide the frame into checkerboard patterns, as image warping will maintain the pixel intensity value of its corresponding points to be the same. With these checkerboard pattern frames, we are ready to do real message decoding.

### 3.4.2 Decoding algorithm

Our decoding algorithm is also based on the temporal signal amplitude alternation. In the encoded pattern detection part, we check the temporal difference of the frame sequence to determine the blocks that's been encoded. Similarly, once we are confirmed with which block is encoded with real message, we just need to look into that particular block with a synchronized two consecutive frames, if the brightness of that block is changing from high to low, then we decode it as a bit 1 messages, otherwise we decode it as bit 0 message. As a complementary, for motion videos, we need to check the start of the eight frames. We can easily detect motion by calculating the pixel by pixel difference between frames, because once there's motion in the video, there must be brightness change for some of the pixels inside that frame. We can conclude the decoding algorithm described in Algorithm 2 and Figure 3.9.

**Algorithm 1** Encoding algorithm

```
**Input**: original video at 30 fps.
**Output**: a flicker-free video encoded with messages at 120 fps.
Extract frames from the original video.
for each frame in the extracted sequence of frames do
    Calculate the texture range value matrix.
    Quadruple each frame.
end for
for each frame in the quadrupled set do
    for each block inside each frame do
        Count the number of pixels above threshold 8 (pixel intensity value) as Num.
        Calculate the block's standard deviation of texture range value matrix as Dev.
        for $Num > blocksize^2/2$ do
            if $Dev < 15$ then
                $Mask = 1$ ($should\ encode$)
            else
                $Mask = -1$ ($not\ encode$)
            end if
            Save mask into buffer.
            for every bit in real message sequence do
                if $bit_i = 1$ then
                    $frame_i = intensity_i + 2$;
                    $frame_{i+1} = intensity_{i+1}$ - 3;
                else
                    $frame_i = intensity_i$ - 3;
                    $frame_{i+1} = intensity_{i+1} + 2$;
                end if
            end for
        end for
    end for
end for
Output frames as video at 120 fps.
```

**Algorithm 2** Decoding algorithm

```
Input: recorded video at 240 fps.
Output: decoded bits stream.
Extract frames from the original video.
for each frame in the extracted sequence of frames do
    Calculate histogram for odd and even frame sets.
    Buffer only the clean frames.
end for
for each frame in the clean frame set do
    Resize and warp the frame into correct perspective.
    Crop only the image region inside each frame.
    Detect the starting frame with pixel-by-pixel analysis for consecutive frame.
    for each block inside each frame do
        Calculate the average brightness of the block as $Avg_i$, i is count in temporal domain.
        if $Avg_i > Avg_{i+1}$ then
            $OutBit = 1$
        else
            $OutBit = 0$
        end if
        Save OutBit into buffer.
    end for
end for
OutBit is the decoded message from recorded videos.
```

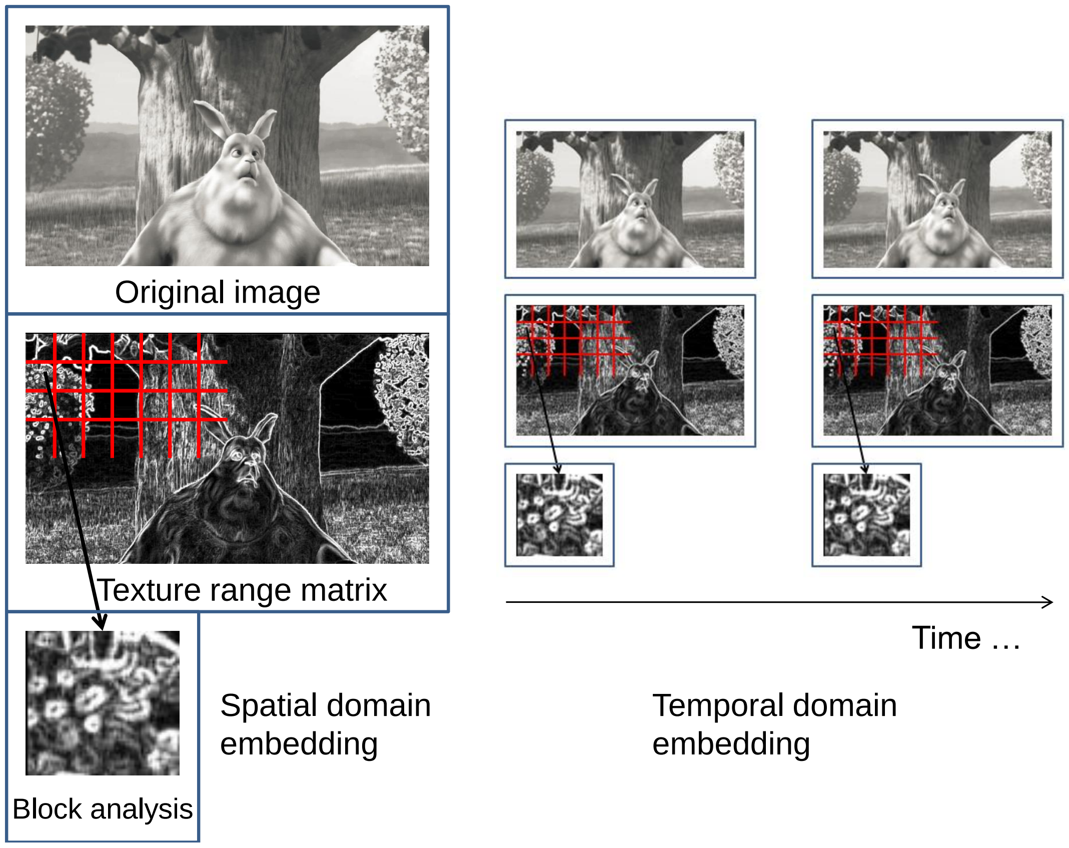


Figure 3.8 Content adaptive encoding method: the original image can be color or grayscale image, being calculated the texture range value, which represents the level of texture for the original image, and will be placing a checkerboard on top of it as spatial embedding. In block-wise analyis, for each block, if more than 50% of pixel is good pixel and its standard deviation is smaller than 15, then mark this block as block should be encode. Do the same analysis in temporal domain; alternate brightness based on real message.

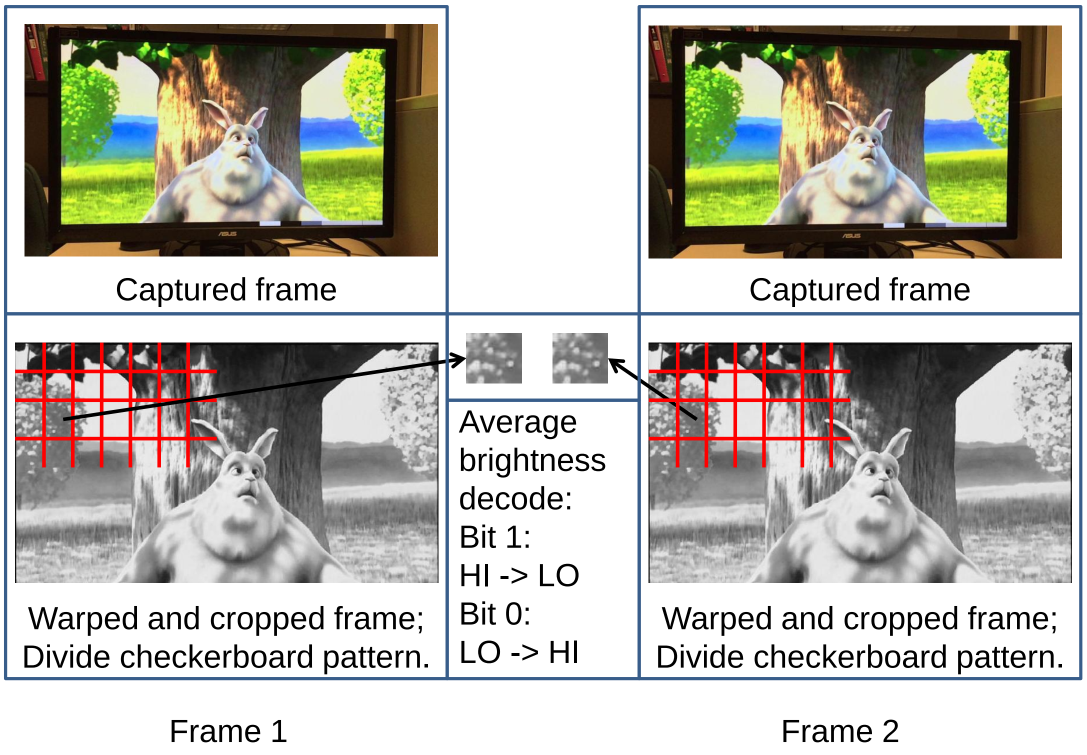

Captured frame
Captured frame
Warped and cropped frame; Divide checkerboard pattern.
Average brightness decode:
Bit 1:
HI -> LO
Bit 0:
LO -> HI
Warped and cropped frame; Divide checkerboard pattern.
Frame 1
Frame 2

Figure 3.9 Temporal signal amplitude alteration decoding method: the captured video is extracted and classified for clean and tainted frames based on histogram analysis. For image pre-processing, we need warp the frame into correct perspective and crop only the image we want. Based on the encoded checkerboard pattern known, the camera side can do real message decoding by tracking the signal amplitude alteration, i.e. average brightness change, in temporal domain. If the trend is from brighter to darker block, then decode as bit 1, otherwise decode as bit 0.

# Chapter 4

# Prototype Implementation

## 4.1 System platform

The implementation of the proposed content adaptive screen-camera communication system consists of a transmitter and a receiver component. For the transmitter, we take an original image or a video stream of 30 fps and a data bits stream of real message as our input. The original 30 fps video is quadrupled and made into video of 120 fps. The real message is encoded using Manchester code and then embedded into each frame's "good block" with all the pixels' intensity value either being increased or decreased with a pre-defined signal amplitude. The "good blocks" are selected based on image content analysis. The 120 fps video with message is a YUV sequence to be displayed at 120 fps on a computer screen, where the refresh reate is set to be 120 Hz, using *glvideoplayer* [31]. We choose an uncompressed YUV format to avoid any artifacts caused by video compression schemes. The receiver is a smartphone camera with high frame rate video recording capability. We choose to use iPhone6 since it allows for 240 fps capture. It captures the video sequence displayed on the display screen and detects the message embedded inside the video sequence.

Currently, both the transmitter and the receiver work offline. We use Matlab to multiplex the original video sequence with the data stream to create encoded version of the video. We use glVideoPlayer and a machine equiped with GPU to precisely control the frame rate of the displayed video. For the receiver, it also works in offline mode - we use a smartphone camera to capture the video frames and use Matlab to run post-processing of the recorded video.

We implement the prototypes proposed in chapter 3 and 4. For the transmitter side, we implement the whole encoding system discussed in Figure 3.8. We analyze the input frames and do content adaptive embedding with random bits stream inserted as a more generalized evaluation. For the receiver side, we implement detect encoded patterns in Figure 3.6 and real message decoding in Figure 3.9 separately, as we notice the detect encoded patterns should be 100% accurate to maintain the performance of the decoding algorithm, so at this point, we implement them as individual module for results analysis. Also, there are some minor assumptions for the decoder. We assume that the decoder knows the checkerboard size, the original video resolution and the encoded patterns for each frame for evaluation purpose of our decoding algorithm. This eliminates pixel offsets and error for texture analysis on the receiver side introduced from several factors, including video distortion, ambient light change and camera exposure setting. In a full protocol design, these parameters can be included in packet headers or inferred through additional receiver processing.

The system evaluation experiments setting is shown in Figure 4.1. We conduct experiments in a well-lit indoor office room environment using a display monitor screen as the transmitter and a smartphone camera as the receiver. We use an ASUS VG248QE 24-inch monitor to display a set of test videos at a rate of 120 Hz. The screen resolution is 1360×760 while video resolution is 1280×720. The displayed videos are recorded as video streams at 240 fps with an iPhone6 using its built-in camera application in the *Slo-Mo* mode. The default distance between the screen and the camera is set to 70 cm, where the screen fills the camera image. The iPhone is mounted on a tripod.

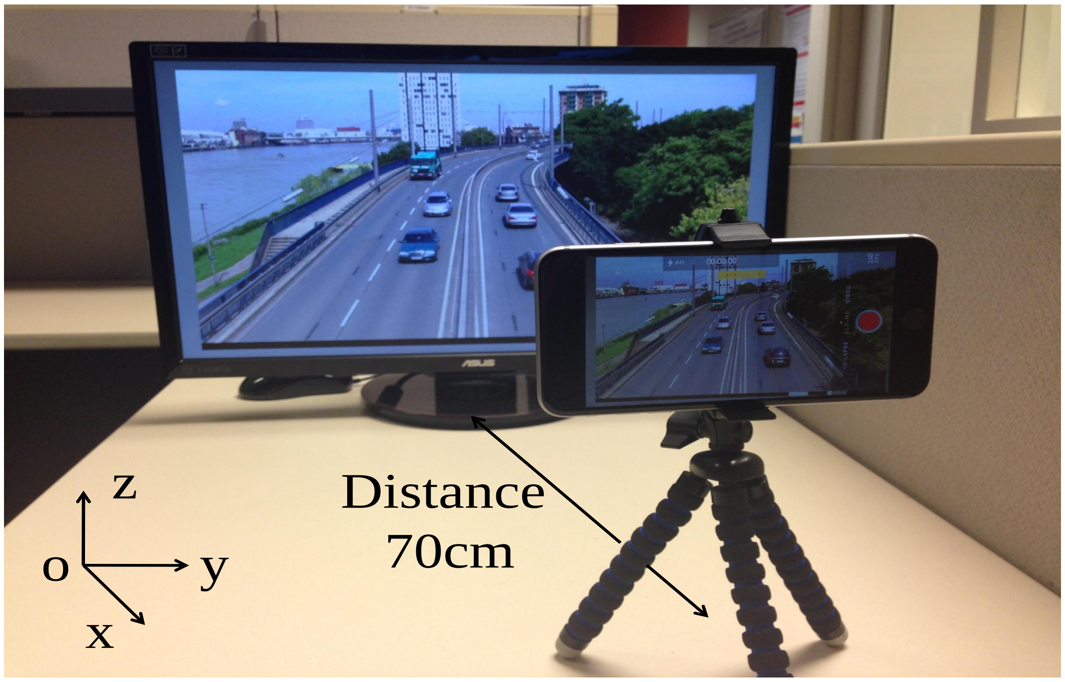


Figure 4.1 Experiment settings: the monitor serves as the transmitter platform to display the encoded video at 120 fps using glVideoPlayer. Iphone6 is in stand hold and its default camera application is set to be in Slo-Mo mode with auto-exposure fixed. The distance between the screen and camera is 70cm for the image to fill in the camera.

## 4.2 Experiment video selection

We selected a set of 10 videos from two publicly available standard data sets [32, 33]. Table 4.1 and 4.2 shows screenshots of sample test video sequences. These test videos represent a wide variety of aspects to explore. *bigbuckbunny*, *Bosphorus* and *YachtRide* are examples of slow motion videos where there are large areas of background with static natural scene and have a key role, either the bear or the boat, act as the moving object. On the other hand, *football*, *highway*, *Jockey* and *ReadySetGo* are representatives of relative fast motion videos. Also, *Mobile* has very clear sharp edges in the imag contour, *ShakeNDry* is full of textured regions in the image and *walking* has some extremely bright and dark regions that might be saturated in the camera side. All these factors are our point of interest and worth an effort to explore. Therefore, we choose these 10 videos as our test sample.

## 4.3 Conclusion

This chapter reveals the prototype implementation details. Such as screen resolution, video player and video encoding process on the transmitter side as well as camera settings and recorded video processing on the receiver side. Also, in this chapter, some of the assumptions in the experiments are stated and the reason to choose the test videos are explained. Based on all these information, we can move forward to the next chapter for experiment results evaluation.

| | |
|---|---|
| | |
| bigbuckbunny | Bosphorus |
| | |
| football | highway |
| | |
| Jockey | Mobile |

Table 4.1 Sample test videos 1.

| | |
|---|---|
| | |
| ReadySetGo | ShakeNDry |
| | |
| walking | YachtRide |

Table 4.2 Sample test videos 2.

# Chapter 5

# System Performance Evaluation

We experimentally evaluate the effectiveness of content adaptive encoding method and the accuracy of encoded pattern detection as well as decoding algorithm based on temporal signal amplitude alteration.

## 5.1 Evaluation metrics

On the transmit side, in evaluating the level of flicker, we classify it into five levels based on the viewing experience of the viewer. We use this subjective way to measure flicker is because there's no standard way of doing objective evaluation of videos on screen right now. As shown in Figure 2.2, the five levels we have is: widespread, imitating flicker; obvious flicker; noticeable flicker; minor flicker and no flicker observed.

On the receiver side, the primary metrics for evaluation are *bit error rate* and *goodput*. Bit error rate (BER) is defined as the ratio of the number of error bits decoded over total number of bits transmitted in Equation 5.1. Usually, throughput is the term in communication field to evaluate the system, but we choose goodput over throughput since the bit error rates can be highly variable for embedded screen-camera communications. Therefore, we define goodput as the number the correctly received bits per unit time as in Equation 5.2.

$$\text{Bit error rate (BER)} = \sum_{\text{all frames}} \frac{Err}{N} \tag{5.1}$$

$$\text{Goodput} = \sum_{\text{all frames}} \frac{D}{t} \tag{5.2}$$

where Err is the number of error bits decoded, N is the total number of bits being transmitted, $D$ is the number of correctly decoded bits and $t$ is the transmission time.

In addition, we define *transmit rate* in Equation 5.3 to help evaluate the effectiveness of our content adaptive encoding method. Note that the transmit rate in our system is dependent on the content of the carrier frames, because it encodes more bits in image areas that are conducive to embedding. The transmit rate therefore varies.

$$\text{Transmit Rate} = \sum_{\text{all frames}} \frac{B \times b}{V \times F} \tag{5.3}$$

where $B$ is the total number of encoded blocks per frame, $b$ is the number of bits encoded in each block, $V$ is the video frame rate, and $F$ is the number of frames needed to encode one bit.

## 5.2 System evaluation

Following the prototype implementation steps, we test the selected sample videos for our system. In terms of flicker perception, all the ten videos result in no flicker perception at all if keep a viewing distance at 50cm away or further, and only some minor flicker in small regions will be noticed if get even closer to the screen.

### 5.2.1 Encoded pattern detection evaluation

For detecting the encoded pattern on the receiver side, the error rate for 10 test videos of static scene are shown in Figure 5.1. In generally, the error rate is in the range of 2% to 6%. As can be seen, *Bosphorus, YachtRide* and *bigbuckbunny* have least error rate. In trying to understand what major factors contribute in the error, we plot the encoded pattern on the transmit side and difference image on receiver side (a sample example for *bigbuckbunny* is shown in Figure 5.1), and different error types in Figure 5.2 for comparison. In the figure, red block means not encoded block detected as encoded block, green block means bit 0 block detected as encoded block, blue block means bit 1 block detected as encoded block and gray blocks means original encoded blocks without error. We learn that higher texture regions tend to have higher error rate. Therefore, there's a trade-off between encoding method and decoding accuracy, since we need to encode into texture regions in order to hide information from being observed.

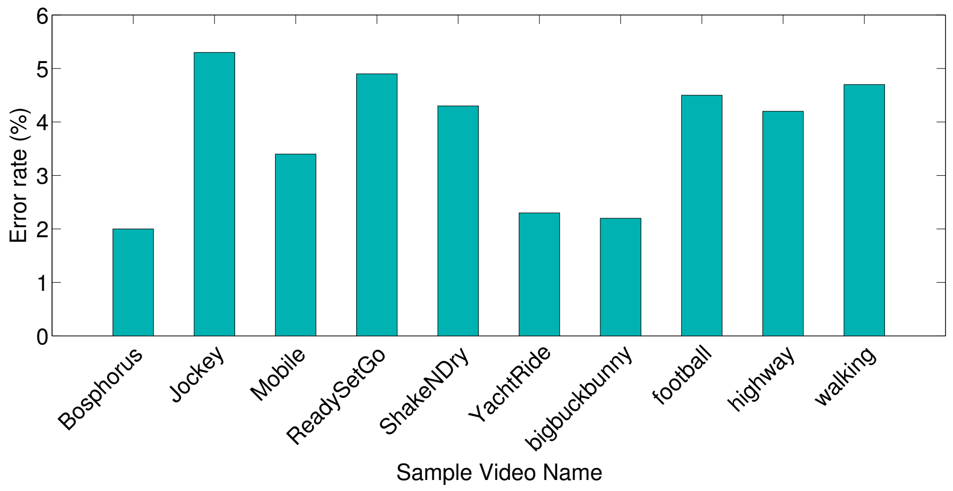


Figure 5.1 Error rate for encoded pattern detection evaluation: experiments result show the error rate of all 10 test videos are in the range of 2% to 6%.

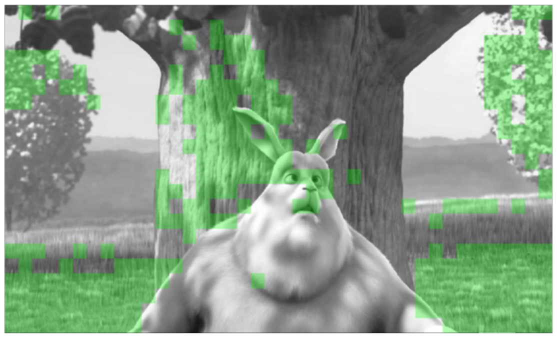
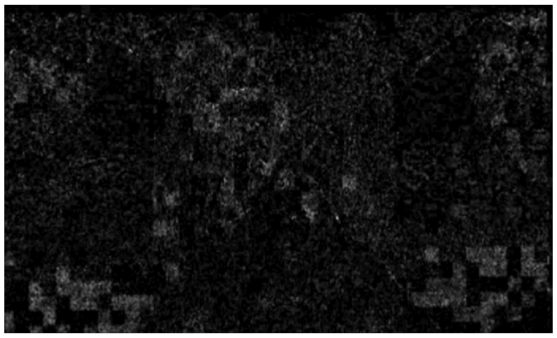
| | |
|---|---|
|  |  |
| Encoded pattern on transmit side | Difference image on receiver side |

Table 5.1 Encoded pattern vs. difference image.

As a summary, for evaluating the encoded patterns on the receiver side, we can get an error detection rate in the range of 2% to 6% for all the test videos. Consider the high reliance of decoding algorithm on the encoded patterns, we are not including this part in the decoding performance evaluation part.

### 5.2.2 Decoding algorithm evaluation

For decoding algorithm evaluation, we evaluate the communication link performance for our system for two use-cases: *static*, where the visual content of the test video does not change, and *dynamic*, where the visual content (i.e. background) of the test video is changing. The experiment results for static case is shown in Figure 5.2. From the results, we can see that the bit error rate is ensured to be within 7%. And in terms of system capacity, the goodput for these 10 videos range from 7 - 23 kbps, because the number of messages being embedded depend on the image content itself. Take *Bosphorus* for example, we can see that there are a lot of plain texture regions in the sample video sequence, like sky and sea water, thus leading to smaller regions being encoded. For *Mobile* to reach the highest goodput is because the image content in this video is highly textured and it contains a variety of difference objects, so there will be more regions available for hiding information.

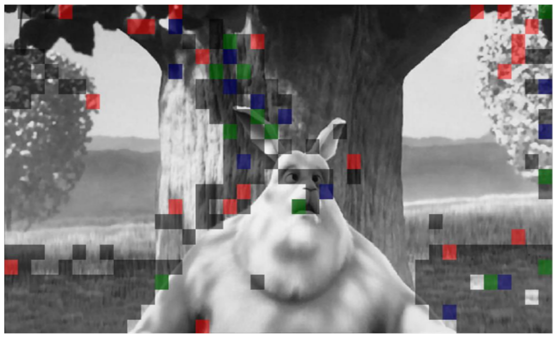
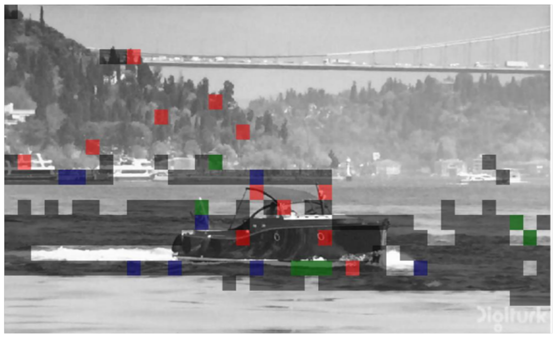
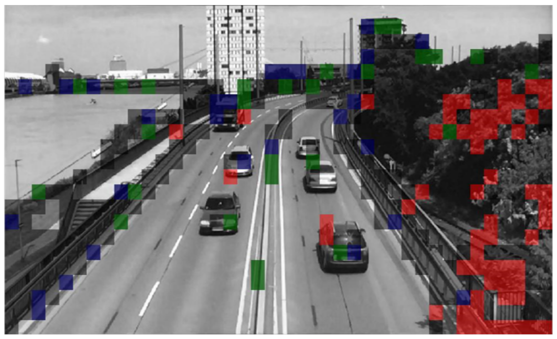
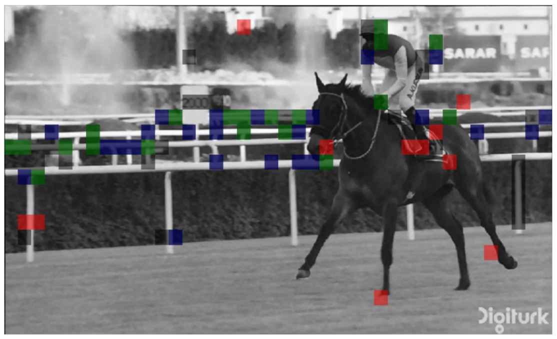
| | |
|---|---|
|  |  |
| bigbuckbunny | Bosphorus |
|  |  |
| highway | Jockey |

Table 5.2 Error type for sample videos: 1. red blocks mean not encoded blocks detected as encoded blocks, 2. green blocks mean bit 0 block detected as not encoded block, 3. blue blocks mean bit 1 block detected as not encoded block, 4. gray blocks mean original encoded blocks without error.

Moreover, the experiment results for dynamic case is shown in Figure 5.3. Dynamic scene suffer from higher bit error rate but will have better flicker perception. This observation can be explained that dynamic scene has plentiful information conveyed itself. More or less, people might focus on the moving objects much more than static background, therefore, flicker is not as noticeable. On the other side, moving objects make it harder to do decoding, because the synchronization problem, tainted frame pollution is much more severe in dynamic scene case. Fortunately, using our temporal signal amplitude decoding algorithm, we can still achieve average goodput to be 16.52kbps and maintain bit error rate within 20%. Take *Jockey* for example, it's the fastest video tested, but *Bosphorus* has slowest motion, therefore it's BER is within 5%.

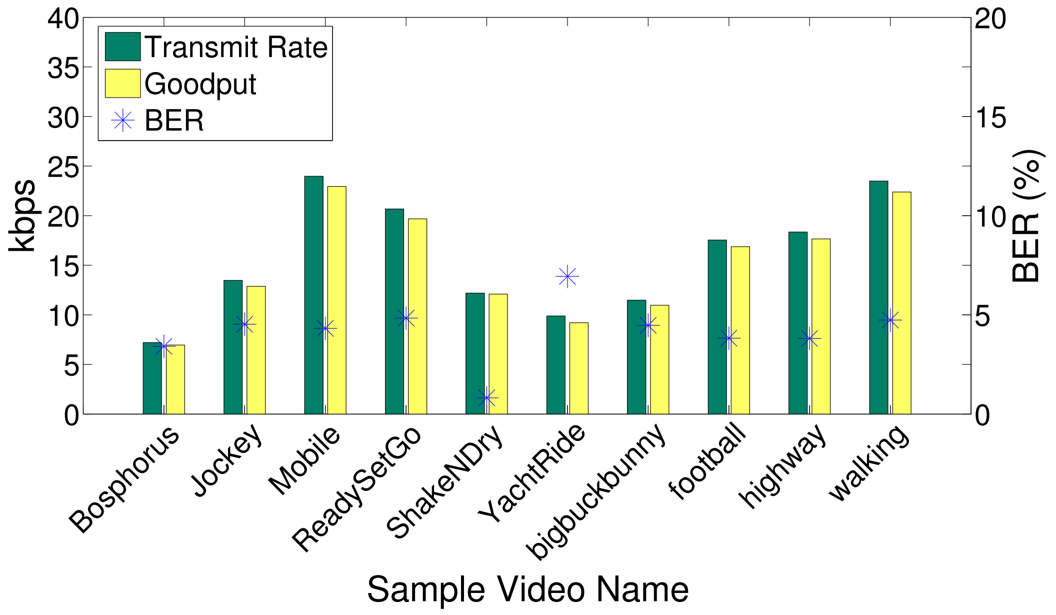


Figure 5.2 System performance for static videos: in this case, the system can achieve near zero flicker with an average goodput to be 15.16 kbps. And maintain the bit error rate to be within 7%.

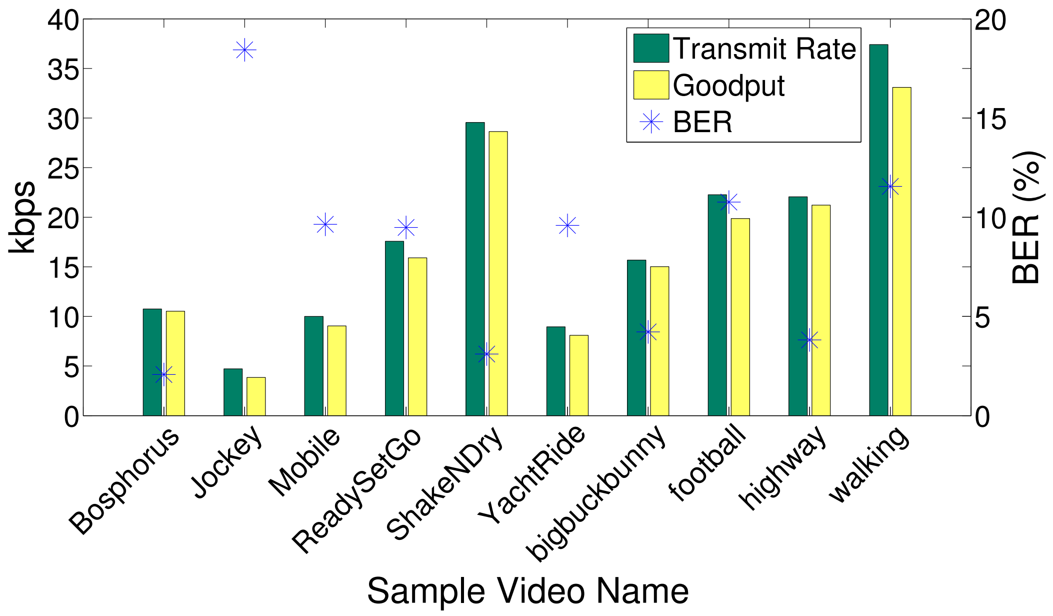


Figure 5.3 System performance for dynamic videos: in this case, the system can achieve no flicker observed for all test videos and an average goodput to be 16.52 kbps. The bit error rate is within 20%. Dynamic scene video contains plentiful information, so there's a trade-off between flicker perception and bit error rate.

## 5.3 Conclusion

In this chapter, we evaluate the system performance for the test videos. In general, all the encoded videos produce near zero flicker perception. And dynamic scene video performs slightly better than static scene video in the flicker perspective. Considering the system capacity, both cases can give an average goodput to be around 15 kbps. However, for the static case, the bit error rate is within 7% and the dynamic case is below 20%. This is because tainted frame issues and synchronization issues are severe in this case. In addition, the encoded pattern detection method has an error rate to be from 2% to 6%. We exclude this part from the decoding algorithm because the decoding process is highly dependent on the accuracy of this method.

# Chapter 6

# Conclusions

This thesis proposes a novel content adaptive flicker-free screen-camera communication system. Following the insight we obtain from aspects leading to flicker perception, we apply the Manchester encoding scheme with content adaptation pattern in an edge avoiding checkerboard pattern. Also, we analyze factors causing error in the communication channel and propose a temporal signal amplitude alteration detection algorithm to determine the original encoded patterns and decode real embedded messages. Results show that our system can achieve near zero flicker perception and ensure a high communication goodput and low bit error rate at the same time.

The key contributions of this thesis are listed as follows:

1. Conducting experiments and providing insights on factors that contributes to flicker perception of an embedded screen-camera communication system, including frame rate, modulation amplitude, image content, edge effect and the viewer's field of view.

2. Proposing a content adaptive encoding method to maximize system capacity in both temporal and spatial domain and maintaining the system to be in the near zero flicker perception level.

3. Identifying factors causing communication error and providing directions to improve system accuracy. Investigating detection of the encoded patterns and decoding real embedded messages on the receiver side and proposing a temporal signal amplitude alteration detection algorithm to achieve high accuracy for the screen-camera communication system.

4. Providing comprehensive system performance evaluation and showing that this content adaptive system has the performance to outperform existing methods in the sense that it can achieve flicker-free visual light communication and maintain high system capacity and embedded information accuracy at the same time.

# Bibliography

[1] Samuel David Perli, Nabeel Ahmed, and Dina Katabi. Pixnet: interference-free wireless links using lcd-camera pairs. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, pages 137–148. ACM, 2010.

[2] 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.

[3] Kensei Jo, Mohit Gupta, and Shree K Nayar. Disco: Displays that communicate. 2014.

[4] Tan Jin Soon. Qr code. *Synthesis Journal*, 2008:59–78, 2008.

[5] Wenjia Yuan, Kristin Dana, Ashwin Ashok, Marco Gruteser, and Narayan Mandayam. Dynamic and invisible messaging for visual mimo. In *Applications of Computer Vision (WACV), 2012 IEEE Workshop on*, pages 345–352. IEEE, 2012.

[6] 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.

[7] Anran Wang, Zhuoran Li, Chunyi Peng, Guobin Shen, Gan Fang, and Bing

Zeng. Inframe++: Achieve simultaneous screen-human viewing and hidden screen-camera communication. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 181–195. ACM, 2015.

[8] Tianxing Li, Chuankai An, Xinran Xiao, Andrew T Campbell, and Xia Zhou. Real-time screen-camera communication behind any scene. In *Proc. of MobiSys*, 2015.

[9] Abbas Cheddad, Joan Condell, Kevin Curran, and Paul Mc Kevitt. Digital image steganography: Survey and analysis of current methods. *Signal processing*, 90(3): 727–752, 2010.

[10] Neil F Johnson and Sushil Jajodia. Exploring steganography: Seeing the unseen. *Computer*, 31(2):26–34, 1998.

[11] Shuyu Shi, Lin Chen, Wenjun Hu, and Marco Gruteser. Reading between lines: high-rate, non-intrusive visual codes within regular videos via implicitcode. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 157–168. ACM, 2015.

[12] Wenjun Hu, Hao Gu, and Qifan Pu. Lightsync: Unsynchronized visual communication over screen-camera links. In *Proceedings of the 19th annual international conference on Mobile computing & networking*, pages 15–26. ACM, 2013.

[13] 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.

[14] Anran Wang, Shuai Ma, Chunming Hu, Jinpeng Huai, Chunyi Peng, and Guobin Shen. Enhancing reliability to boost the throughput over screen-camera links. In

*Proceedings of the 20th annual international conference on Mobile computing and networking*, pages 41–52. ACM, 2014.

[15] DH Kelly. Sine waves and flicker fusion. *Documenta Ophthalmologica*, 18(1):16–35, 1964.

[16] H DeLange. Experiments of flicker and some calculations of an electrical analogue of the foveal systems. *Physica*, 1952.

[17] JD Bullough, K Sweater Hickcox, TR Klein, and N Narendran. Effects of flicker characteristics from solid-state lighting on detection, acceptability and comfort. *Lighting Research and Technology*, 43(3):337–348, 2011.

[18] Hany Elgala, Raed Mesleh, and Harald Haas. Predistortion in optical wireless transmission using ofdm. In *Hybrid Intelligent Systems, 2009. HIS'09. Ninth International Conference on*, volume 2, pages 184–189. IEEE, 2009.

[19] Börje Andrén, Kun Wang, and Kjell Brunnström. Characterizations of 3d tv: active vs passive. In *SID symposium digest of technical papers*, volume 43, pages 137–140, 2012.

[20] Edward H Adelson. Lightness perception and lightness illusions. *New Cogn. Neurosci*, 339, 2000.

[21] Tom Cornsweet. *Visual perception.* Academic press, 2012.

[22] Charles Chubb, George Sperling, and Joshua A Solomon. Texture interactions determine perceived contrast. *Proceedings of the National Academy of Sciences*, 86(23):9631–9635, 1989.

[23] Albert Abraham Michelson. *Studies in optics.* Courier Corporation, 1995.

[24] James Davis, Yi-Hsuan Hsieh, and Hung-Chi Lee. Humans perceive flicker artifacts at 500 [emsp14] hz. *Scientific reports*, 5, 2015.

[25] John C Russ and Roger P Woods. The image processing handbook. *Journal of Computer Assisted Tomography*, 19(6):979–981, 1995.

[26] Manish H Bharati, J Jay Liu, and John F MacGregor. Image texture analysis: methods and comparisons. *Chemometrics and intelligent laboratory systems*, 72 (1):57–71, 2004.

[27] Tan Pang-Ning, Michael Steinbach, Vipin Kumar, et al. Introduction to data mining. In *Library of Congress*, page 74, 2006.

[28] William S Noble. What is a support vector machine? *Nature biotechnology*, 24 (12):1565–1567, 2006.

[29] How-Lung Eng and Kai-Kuang Ma. Noise adaptive soft-switching median filter. *Image Processing, IEEE Transactions on*, 10(2):242–251, 2001.

[30] Chirs A Glasbey and Kantilal Vardichand Mardia. A review of image-warping methods. *Journal of applied statistics*, 25(2):155–171, 1998.

[31] glvideoplayer. `https://code.google.com/p/glvideoplayer/`.

[32] `http://ls.wim.uni-mannheim.de/de/pi4/research/projects/retargeting/test-sequences/`, .

[33] Elemental technologies 4k test sequences. `http://www.elementaltechnologies.com/resources/4k-test-sequences`, .