CODEN:LUTEDX/(TEIE-3136)/1-94(2024)

Industrial Electrical Engineering and Automation

Temporal Artefact Mitigation Rolling Shutter Videography

Marcus Remnélius Bou
Frans Wallinius

Division of Industrial Electrical Engineering and Automation
Faculty of Engineering, Lund University

Bachelor's Thesis

Temporal Artefact Mitigation in Rolling Shutter Videography

By:

Marcus Remnélius Bou

Frans Wallinius

Supervisor at
Axis Communications AB
Philip Stjärneblad

Supervisor at
Lund University
Bernard Schmidt

Date: 10/06/2024

Division of Industrial Electrical Engineering and Automation
LTH, Faculty of Engineering, Lund University
SE-221 00 Lund, Sweden

LTH at Campus Helsingborg
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

LTH vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

Printed in Sweden
Lunds universitet
Lund 2024

Abstract

Energy efficiency is key in our modern society, and in its pursuit, unexpected problems can appear. For instance, when Axis Communications in the pursuit of image quality and energy efficiency used strobing IR-illumination to illuminate for their global shutter cameras, they observed an artefact in the form of a white band in their rolling shutter cameras. The purpose of this thesis is to find a software solution to this problem with which to update existing Axis rolling shutter cameras with.

Rolling shutter cameras capture images by exposing its sensor to light and scanning its pixels row by row. This can therefore be affected by flashing lights where only some of the rows are exposed to the light which results in a white band in the image. The PIE-team at Axis developed a solution that detected the band and found its position in the image from frame to frame. This was then used to calculate the strobe frequency which was used to synchronize the frame rate. When the band was stable in the image, a delay was applied to the exposure to capture images when the strobe is off. Unfortunately, the frame rate could never exactly match the frequency due to discretization, which resulted in the strobe eventually reappearing.

Initially, the preexisting work was analysed and alterations to the detection algorithm was implemented to store a background without the strobe of which to compare the current frame to. If there was a difference higher than a threshold, the band was identified, and its centre was calculated. The band was then synchronized, and the exposure delayed as before. To tackle the strobe's reappearance, another method was developed that checked the top and bottom row of the image to see if the strobe reappeared, and if it did, the exposure was delayed again. Testing was performed in both a laboratory and in the field with a stable and a less stable strobe trigger.

The results showed that the algorithm functioned as intended in both the laboratory and the field, with some limitations. In the controlled laboratory environment, the algorithm performed satisfactorily every run. How accurately the camera calculated the frame rate was tested. With the less stable strobe, the algorithm calculated a frame rate with a standard deviation of 0.00241 every run, and with the stable strobe, a standard deviation of 0.00032. In the field, many factors counteracted the functionality, but mostly how much light reached the camera.

The laboratory tests with the stable and less stable strobe revealed that even when a stable strobe is used, some frequency drift still occurs, confirming the need to reapply a delay. In the field, surrounding light sources and obstructing objects affected the band's appearance at different parts of the image, which made it difficult to get predictable results, but when the strobe was visible enough, the algorithm worked well. In conclusion, an algorithm was developed that detect a strobe artefact in the form of a white band, eliminates it and keeps it eliminated.

Keywords

Rolling shutter, global shutter, artefact, artifact, artefact mitigation, strobe, IR, image analysis

Sammanfattning

Energieffektivitet är viktigt i vårt moderna samhälle och i jakten på det kan oväntade problem uppstå. När Axis Communications i strävan efter ökad bildkvalitet och energieffektivitet tillämpade IR-stroboskop med hög toppeffekt som belysning åt kameror med global slutare, observerades en artefakt i form av ett vitt band i närliggande kameror med rullande slutare. Syftet med detta examensarbete är att utveckla en mjukvarulösning för detta problem där befintliga kameror med rullande slutare kan uppdateras.

Eftersom en rullande slutare läser av sensorn rad för rad sekventiellt är de känsliga för blinkande ljus eftersom de rader som exponeras för ljuset leder till ett vitt band i den slutliga bilden. PIE-gruppen på Axis utvecklade en lösning som detekterade artefakten och följde dess position i bild mellan exponeringar. Stroboskopets frekvens kunde därefter beräknas och kamerans bildfrekvens synkroniserades till denna. När artefakten var stabiliserad beräknades en fördröjning för att flytta nästkommande exponeringar till när stroboskopet inte lyste. Tyvärr kunde inte bildfrekvensen exakt matcha stroboskopets frekvens på grund av diskretisering. Detta resulterade i att artefakten dök upp igen.

Inledningsvis analyserades det befintliga arbetet och förändringar av detektionsalgoritmen bedömdes nödvändiga. Genom att identifiera en bakgrund och lagra denna kunde den aktuella bilden jämföras och en skillnad beräknas. Om denna skillnad på någon rad översteg ett gränsvärde var artefakten identifierad och kunde elimineras. För att hantera att artefakten återkommer i bild implementerades ytterligare en metod som övervakade den översta och den understa raden av bilden. När artefakten sedan återkom applicerades en ny fördröjning. Detta testades både laboratoriet och i fält.

Resultaten visar att algoritmen fungerar enligt målen i både laboratoriet och i fält med vissa begränsningar. I en kontrollerad laboratoriemiljö fungerade algoritmen tillfredställande vid varje tillfälle. Hur noga bildfrekvensen synkroniserades testades. Med ett instabilt stroboskop beräknades bildfrekvensen med en standardavvikelse på 0,00241 och med ett stabilt stroboskop uppnåddes en standardavvikelse på 0,00032. I fält påverkade en mängd faktorer resultaten men främst mängden ljus som nådde sensorn.

Laboratorietesterna avslöjade även frekvensdrift hos kamerorna vilket bekräftade behovet av att återapplicera fördröjningen. I fälttesterna påverkade ljuskällor och andra faktorer artefaktens profil vilket försvårade möjligheten att få förutsägbara resultat. Men när artefakten var tydlig nog att påverka bildkvaliteten fungerade algoritmerna som tänkt. Sammanfattningsvis utvecklades en algoritm som detekterar en artefakt från ett stroboskop, eliminerar den och håller den eliminerad.

Nyckelord

Rullande slutare, global slutare, bildartefakt, artefaktreducering, stroboskop, IR, bildanalys

Table of Contents

Table of Figures

Table of Tables

Glossary/Abbreviations

Aperture – The iris of the lens. Can be adjusted to control how much light a sensor receives.

Blanking – The time in between two image captures when the sensor is inactive.

CMOS – Complementary metal-oxide-semiconductor.

CMOS sensor – A commonly used sensor in digital cameras.

Exposure – The amount of light a sensor is exposed to.

Global shutter – A technology that exposes all pixels at once in a sensor.

IIR – Infinite impulse response.

MA – Moving average.

Readout – When the sensor has been exposed and sends the signal to be processed.

Rolling shutter – A technology that exposes the pixel rows sequentially in a sensor.

Strobe/stroboscope – A device that flashes light at certain frequencies.

Wide dynamic range (WDR) – A technology to increase the contrast range of an image.

Preface/Acknowledgements

We would like to preface this thesis by expressing our utmost gratitude to our supervisors. Philip Stjärneblad, for always being available to us, always being someone to bounce ideas with when encountering roadblocks and providing valuable feedback every step along the way. And Bernard Schmidt, for also always being available in person and online. We also greatly appreciate the time invested to understand our work in order to provide incredibly insightful feedback and ideas.

We would also like to thank the entire PIE team at fixed cameras at Axis as they are a great bunch who welcomed us as soon as we arrived and for always taking their time to help us when asked. And to Martin Nyman and Axis in general for providing us with the opportunity to write our thesis with them.

1. Introduction

Energy efficiency is one of the most important parts in our modern society. In an effort to reduce energy consumption, other unexpected problems can appear. A camera needs illumination to be able to capture video in dark environments. In surveillance applications, cameras are usually discreetly placed. Using infrared (IR) light to illuminate is therefore desired as it cannot be seen with the naked human eye. One way to reduce energy consumption is to pulse the light when the camera captures an image instead of it constantly shining.

With the rise of global shutter videography follows the adoption of pulsing or strobing IR-illumination with the goal of enhanced image quality by increasing peak power output, and in the pursuit of energy efficiency. When a master's thesis student at Axis Communications compared strobing IR-illumination to continuous illumination, the results showed that the energy consumption decreased by 95 % when strobing [1]. However, this did cause problems for existing cameras with a rolling shutter since a nearby IR-stroboscope created temporal artefacts in the form of white bands in the image. These artefacts are unacceptable as they obscure the motive and reduce forensic value in surveillance footage.

Axis Communications AB was founded in 1984 and is a manufacturer of network video surveillance solutions, access control solutions, network intercom solutions, and more. It has around 4000 employees in over 50 countries and is headquartered in Lund, Sweden.

1.1 Purpose

The purpose of this thesis is to study and implement a software solution to this problem where the software of existing systems using rolling shutter cameras can be updated. The expected result is a thoroughly tested implementation with opportunities for improvement as there are many possible situations and environments which require testing.

1.2 Goal

The goal of this thesis is the development and testing of an algorithm that identifies, eliminates and keeps the temporal artefact from a pulsing IR-stroboscope in the form of a band out of the image of Axis surveillance cameras. To verify its functionality, testing in both laboratory and real-world environments is required.

1.3 Problems in need of investigation

To set this thesis on the correct path, several questions are posed which need to be addressed. They are as follows:

1. How to develop an algorithm to reliably detect an artefact in the form of a white band in the image from an IR-strobe?
2. How to synchronize a camera's frame rate to the frequency of an IR-strobe?
3. How to eliminate the artefact by delaying the exposure window when the frame rate is synchronized?
4. How to keep the strobing IR-light out of the exposure window?
5. What is required to be able to reliably run the program many times?
6. How much does testing in a laboratory differ from real-world environments?
7. What is required for the algorithm to function in real-world environments?
8. How much does surrounding light, environment and a change in background affect the performance of the algorithm?
9. Where can the camera and strobe light be placed with respect to one another for the algorithm to function as intended?

1.4 Motivation

Developing and upgrading software is always relevant in electronics and computer science, which is why the problem described in 1.1 is a perfect introduction into this field. From a societal perspective the relevancy of reducing energy consumption is at an all-time high which is why resolving this problem with ingenuity is desired rather than reverting to using continuous illumination. Resolving this problem from Axis' perspective is imperative as addressing issues such as this one ensures compatibility between their existing and new products in the pursuit of image quality and energy efficiency.

1.5 Limitations

Testing will be performed between 25 and 30 frames per second as most cameras run at those frame rates. The exposure will be held at a specific value which means that automatic exposure will be off. Other parts of the exposure algorithm will also be unable to change. The testing and development of the algorithm will be limited to one strobe. Testing will mostly happen on Axis' premises as the development tools will be close at hand. The cameras used for testing will be limited to three Axis P1380 Series Box Cameras at different resolutions.

1.6 Report structure

This report is divided into four main chapters after the introduction. Background information about camera technology, image processing, and previous work is discussed in chapter 2: Background. A solution for the problem in the form of an algorithm that eliminates the strobe is explained in chapter 3: Method. Results of laboratory and real-world tests, observations, and measurements are shown, and an analysis of these results is held in chapter 4: Results. And at last, a discussion of the work is held, and conclusions are drawn in chapter 5: Discussion and Conclusion.

1.7 Division of labour

Task	Frans	Marcus
Algorithm development	50 %	50 %
Laboratory testing	40 %	60 %
Field testing	60 %	40 %
Graphic design	70 %	30 %
Report writing	30 %	70 %

2 Background

This chapter covers necessary background information for understanding the principles and methods used in this thesis. The background information includes digital camera technology and functionality, different data handling methods such as image data collection, dataset filters and thresholding. Previous work which this thesis is based on is also explained.

2.1 Digital sensor technology

A video camera captures many images every second by directing light with a lens towards a sensing element and outputting images at a certain frequency. In the case of analogue cameras, the image is captured on film, but in the case of digital cameras, the image is captured on a grid of sensors. One full scan of the sensor grid is captured and converted into an image or otherwise called a frame [2]. The frequency of these frames is called frame rate with unit frames per second (fps). Digital cameras are the focus of this thesis.

Most of today's digital cameras use CMOS sensors that contain a grid of light sensitive photodiodes that absorb light. The photodiodes in the sensor each produce an electric signal due to the photoelectric effect. The photodiode and its surrounding components such as transistors for buffering and amplification compose what is called a pixel [2], [3]. On these pixels are colour filters that filter the specific red, green and blue wavelengths of visible light, as well as IR-light [4]. The amount of time a sensor collects light is called integration time. When the sensor has collected enough light, it converts these analogue signals into digital signals using analogue/digital converters. The digital signals then represent pixel values in the form of intensity, colour and more depending on the complexity of the camera [2].

2.2 Image capturing

For a sensor to capture an image, it needs to be exposed to light. Exposure must be controlled to ensure that a comprehensible image is captured every time. The exposure is controlled in two ways, with the shutter and the iris. Shutters are used to regulate how much light a sensor receives in each time frame. This is either measured in shutter speed or exposure time. The iris in the lens physically adjusts how much light a sensor receives by opening and closing mechanically, which is also called aperture [5].

Different exposure times are required depending on lighting conditions. A long exposure is desired when in relatively low light conditions such as in astronomy for example. Whilst a shorter exposure is preferred in an environment with an abundance of light such as in direct sunlight [5]. Figure 2.1 illustrates how the exposure time is applied to each discrete row of pixels in a rolling shutter sensor, which is explained in 2.3.

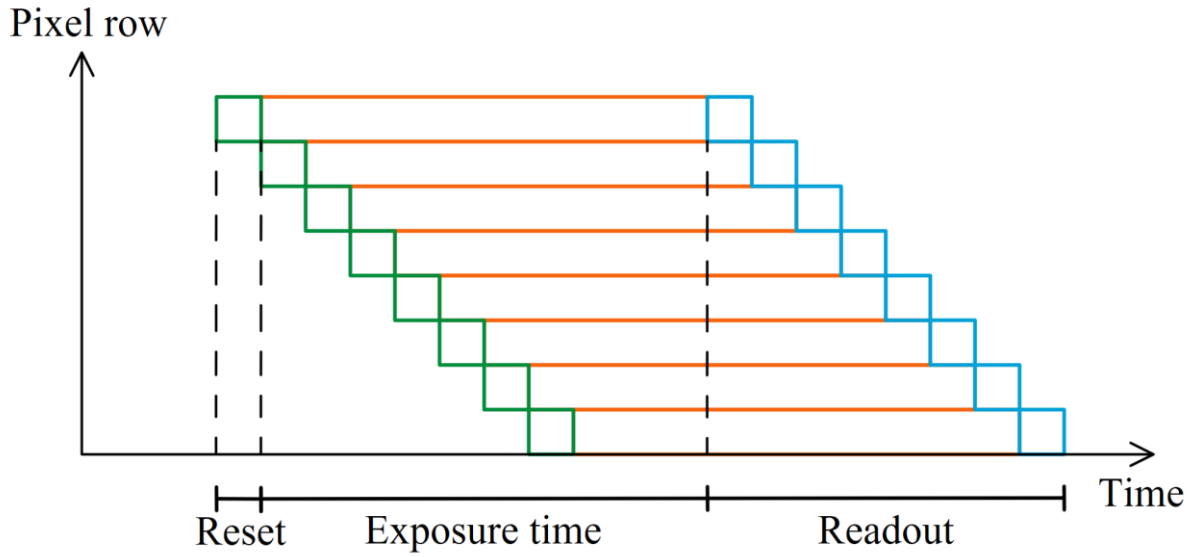

Figure 2.1: Illustrates the discrete exposure time for each pixel row for a rolling shutter where "Reset" is the reset time for a sensor row, "Exposure time" is the time a sensor row is exposed, and "Readout" is the time it takes to output the entire image.

Coupled to the photodiodes in the sensor are amplifying transistors which allows the image brightness to be adjusted by adjusting the gain. This allows the exposure time to remain the same whilst retaining a brighter image. However, this does not improve quality as no additional information is recorded, only altered [3].

Each of these techniques comes with its own set of advantages and disadvantages. Too long of an exposure time can lead to a blurry image, high amplifier gain also amplifies noise, and an adjustable aperture requires a more complex and expensive mechanical lens design. Therefore, a combination of these techniques is commonly used to achieve balanced results.

2.3 Camera shutters

There are a variety of shutter techniques that can either be controlled mechanically or electronically, but only electronically controlled shutters are used in this thesis. Unlike mechanical shutters that physically obstruct light from reaching the sensor, electronic shutters use transistors to control when the sensor is active and can receive light [4].

2.3.1 Rolling shutter

One way to capture an image is to scan the sensor grid row by row which is then converted into pixel values. Scanning row by row is called rolling shutter and each row is read out sequentially [6]. Figure 2.2 illustrates an exposure diagram representation of the principle behind a rolling shutter camera, where the cycle time is the time of each frame. Readout is the time from when the first pixel row ends its exposure window to when the last pixel row ends its exposure window. The exposure time is the time each pixel row is active, and "Blanking" is the amount of time the sensor is inactive. Figures 2.2 and 2.3 are simplifications of figure 2.1 where sensor reset is ignored, readout for every row is interpreted as instant, and the discrete steps are represented as continuous.

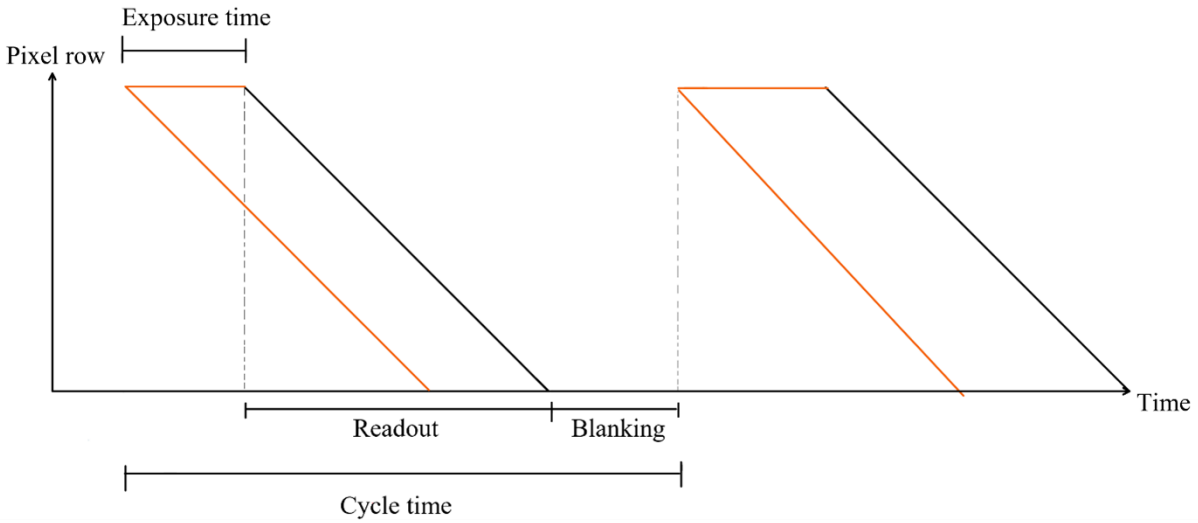

Figure 2.2: Illustrates an exposure diagram representation of the rolling shutter readout mode. The exposure time is exaggerated for illustrative purposes.

Rolling shutter videography has for a long time been the norm in digital videography because of the ability to store the pixel data in memory one row at a time. This is a cost-effective way to construct a high-resolution digital camera since the row data can quickly be stored in, accessed in, and cleared from memory. The rolling shutter does, however, come with certain disadvantages, due to the rolling shutter effect, fast-moving objects may get distorted due to sequential exposure. This issue is compounded in low light conditions when a longer exposure time is needed [6], [7].

2.3.2 Global shutter

An alternative image capturing method is the global shutter where the entire sensor grid is exposed at the same time. This puts a higher demand on the data bus and memory usage, especially with higher-resolution images compared to a rolling shutter. The advantages are however considerable in certain cases, by reading the entire grid at once, movement distortions are eliminated. Since the readout time is shorter than that of a rolling shutter, the time in between frames is longer which could allow for more exposure cycles, resulting in higher frame rates. To lessen the effect of the data requirements, the signals can be read out row by row like in a rolling shutter. Rolling shutter is still the most common technology as it fulfils most needs when unaffected by the rolling shutter effect. Although, one field where global shutter is needed is in machine vision [6], [7]. Figure 2.3 illustrates the same representation as figure 2.2 but for a global shutter sensor.

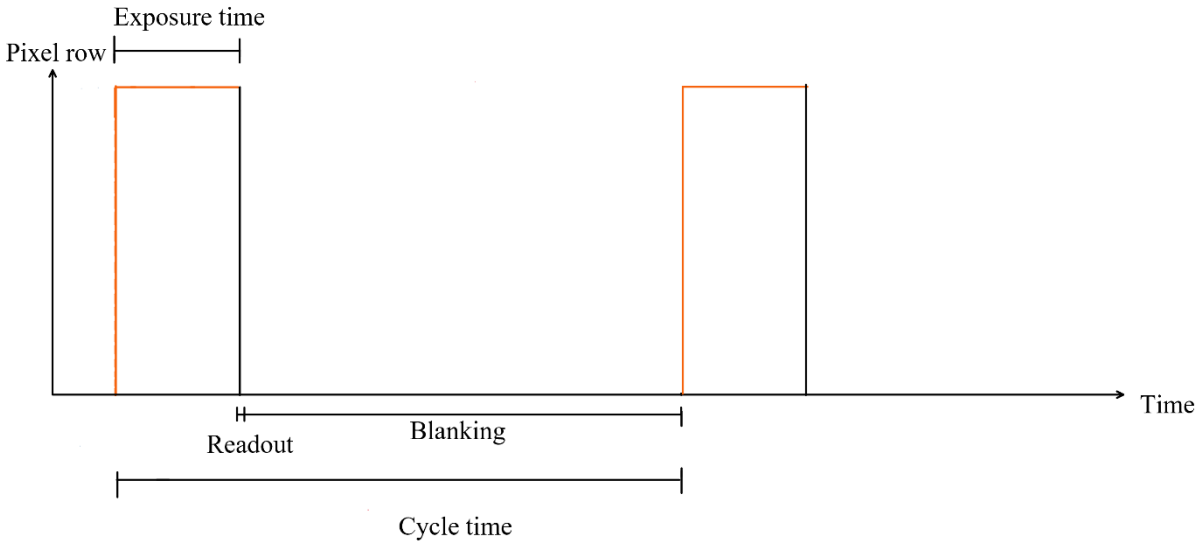

Figure 2.3: Illustrates an exposure diagram representation of the global shutter readout mode. The exposure time is exaggerated for illustrative purposes.

2.3.3 Axis cameras used in development and testing

The first and main camera used in this thesis is the Axis P1385. It has a 1/2.8” CMOS sensor with a rolling shutter at a resolution of 1920×1080 (2 mega pixels (2 MP)). It is also equipped with the ARTPEC-8 chip and 1024 MB of RAM [8].

The second camera used is the Axis P1387 which is based on the same platform as the P1385 but is equipped with a larger 1/2.7” sensor at a resolution of 2592×1944 (5 MP) with the same chip but with 2048 MB of RAM [9].

The last camera used is the Axis P1388 which is also based on the P1385 and is the last version in the series. It uses a 1/1.8” CMOS sensor at a resolution of 3840×2160 (8 MP) with the same chip and 2048 MB of RAM [10].

2.4 Videography using IR-light

Short-wave infrared light, (IR-light), can be utilized to capture images in an absence of visible light since the silicon in regular image sensors are most receptive to it and red visible light [4]. Using IR-light to capture images in low-light conditions for the purpose of surveillance comes with advantages compared to using visible light. Mainly that the naked human eye cannot see infrared light and therefore a surveillance camera using IR-illumination is more discrete than a camera with equivalent illumination in the visible spectrum [11].

As shown in figures 2.2 and 2.3 in 2.3, the global shutter sensor has a considerably longer blanking time for a given frame rate since its unnecessary to illuminate the motive while the camera sensor is inactive. Therefore, pulsing IR-light with a duty cycle matching the sensors exposure time can increase momentary peak power and drastically reduce power consumption. In this thesis an IR-stroboscope (shortened to strobe) was used with a 20 W power draw at maximum intensity and a duty cycle of 2 %, this means that the same light source at a 100 % duty cycle could theoretically draw in the vicinity of 1000 W if the same peak power is to be achieved [12].

In an effort to reduce power consumption, this is desirable. However, the usage of a strobe can interfere with nearby rolling shutter cameras, since the rolling shutter reads each row sequentially. If the strobe pulses with a shorter duty cycle than the exposure time during its exposure window, it will result in a bright band across the image as can be seen in figure 2.8 in 2.7. This is explained in further detail there [13].

2.5 Wide dynamic range

Wide dynamic range (WDR) is the name of a collection of techniques to increase the contrast range in an image by fusing two exposures into one image, either by using different gains on the internal amplifiers as shown in figure 2.4, or by performing two exposures with different exposure times as shown in figure 2.5. This stops the dark areas of an image from being underexposed and the bright areas from being overexposed and results in a useful image even under difficult lighting conditions [14].

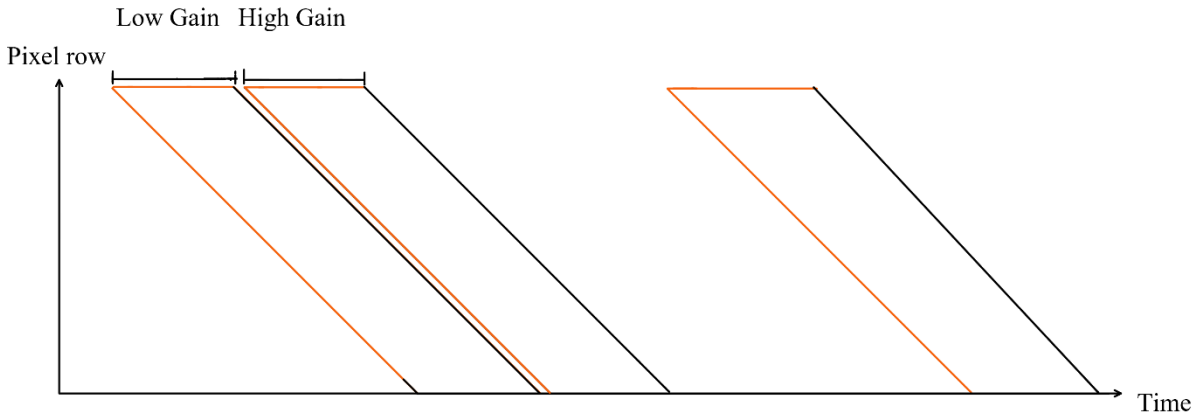

Figure 2.4: Illustrates how WDR is implemented on an Axis P1385 camera.

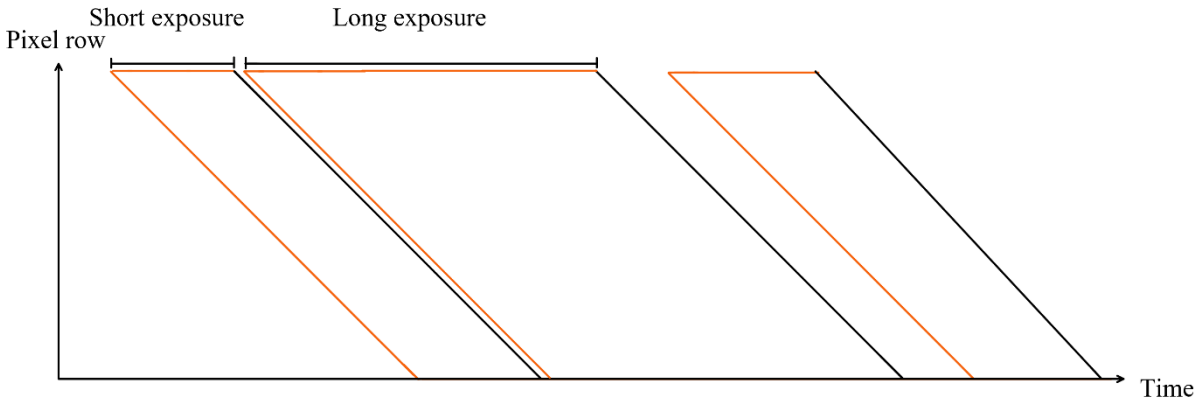

Figure 2.5: Illustrates how WDR is implemented on an Axis P1387 and P1388 camera.

2.6 Image processing techniques

2.6.1 Linear representation of an image

In digital videography, the sensor captures an image which then is represented as a two-dimensional matrix of pixels with certain values. While it is possible to make calculations on this directly, it requires matrix calculations which can be demanding on the hardware. A way to reduce the complexity of the calculations is to perform a linear representation of the image which is a one-dimensional vector with defined rows and columns [15].

A digital image can be described by a discrete function $f(x, y)$ with discrete nonnegative coordinates (x, y) as well as an $m \times n$ matrix A with m rows and n columns, this is known as coordinate indexing. These coordinates range from $x = 0, 1, 2, \dots, m - 1$ and $y = 0, 1, 2, \dots, n - 1$ where x describes the rows and y the columns of the matrix. The origin is defined as the first sampled pixel which could be any corner of the image, although the top left is a common origin [15].

A common variation, especially in programming, is linear indexing as it allows for vector operations on the pixel values. There are two choices for linear indexing, based on either scanning the rows or the columns, column scanning is the principle used in this thesis due to this being the standard implementation used by Axis. The columns scanning begins in the first column at the origin with index 0 through $m-1$, after finishing the first column, the second column scans index m through $2m-1$ and so on. The last column is indexed nm through $nm-1$. This way, every pixel has a unique index denoted by γ that can assume the values $0, 1, 2, \ldots, nm-1$, figure 2.6 illustrates this method. In programming, linear indexing is achievable with nested for-loops by for every column looping through every row [15].

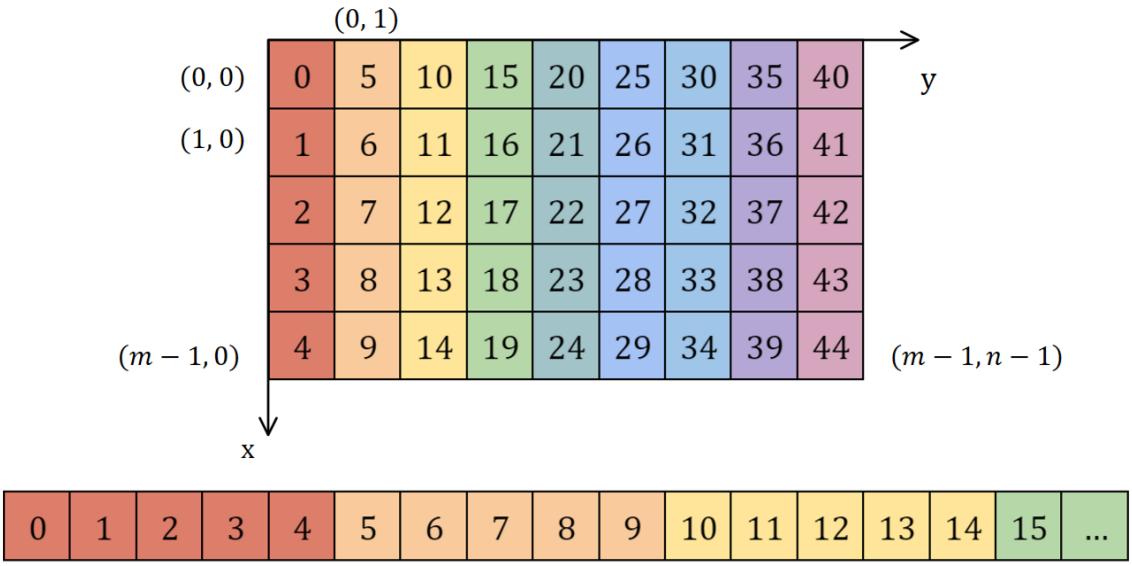

Figure 2.6: Illustrates the linear indexing of a 9×5 pixel matrix where m = 5, n = 9 and the number in each square represents the unique index γ. The numbers in the parenthesis are the coordinates of the nearby square.

2.6.2 Median and mean values of datasets

There are different possibilities to find a certain descriptive value of a dataset, in this work the principle of median and mean values are used. Median values can be found by sorting a set of elements from smallest to largest and finding the middle element of that set. Mean values can be calculated by adding all the elements in a set and dividing the sum with the number of elements [16].

How this applies to image analysis depends on the goals to be achieved. The median is an accurate measure of location in asymmetrical distributions and is useful if e.g. the goal is to ignore individual outliers in the dataset such as a noise spike or something bright in an otherwise dim image. The reason for this is that a few outliers only shift the middle value a few steps in the dataset. The mean conveys the weight of a dataset and is useful in symmetrical distributions with a peak value. However, the mean is more sensitive to outliers as they play a larger role when calculating the mean value. Although, it is more accurate compared to the median if the dataset lacks noise and is evenly distributed [16].

2.6.3 Infinite impulse response filter

An infinite impulse response filter (IIR-filter) is a recursive filter that uses current input signal and past output signals to generate an output signal. The main difference to a finite impulse response filter is that the impulse response never reaches zero but instead approaches zero indefinitely [17].

An IIR-filter is described as follows, where $y[i]$ is the current output, $y[i-1]$ is the past output, $x[i]$ represents the input signal and α denotes the filter coefficient.

$$y[i] = \alpha x[i] + (1-\alpha)y[i-1], \qquad 0 < \alpha < 1 \tag{2.1}$$

IIR-filters are commonly used in digital signal processing due to their ease of implementation in code and their relative accuracy [17].

2.6.4 Adaptive thresholding techniques

Thresholds are of great importance in digital image processing, they are necessary to determine the relevancy of data. Using statistical analysis of the image data with e.g. a basic global threshold, the threshold can be automatically adjusted to suit the current conditions without any manual input. The basic global threshold works by grouping every pixel as either an object point (G_1) or a background point (G_2) as the following equation describes, where T is an initial threshold,

$$g(x,y) = \begin{cases} 1\ (G_1) & if \quad f(x,y) > T \\ 0\ (G_2) & if \quad f(x,y) \leq T \end{cases} \tag{2.2}$$

Afterwards, the entire image is segmented in object and background, then the means m_1 and m_2 are calculated from the pixels in G_1 and G_2. These mean values are then used to calculate a new threshold as,

$$T = \frac{1}{2}(m_1 + m_2) \tag{2.3}$$

This is then repeated until the changes in T is smaller than a chosen value [18].

2.6.5 Moving average filter

Noise is a major issue in digital signal processing, no signal is perfect but in some cases the noise is severe enough to trigger false positives. To counteract noise, a filter can be implemented. It is important to choose a suitable filter for the application since each comes with its own set of advantages and disadvantages. A commonly used filter is the moving average filter (MA-filter) due to its ease of use and simplicity but also because it retains a sharp step response and can be implemented at a low computational cost [19].

An MA-filter works by utilizing the mean of a subset of values in a dataset. The number of values in this subset is the controlling factor of how much noise reduction can be achieved. In the equation below an MA-filter is described as the input signal $x[i+j]$, the output signal $y[i]$ and M which represents the number of values in each subset, also referred to as window size.

$$y[i] = \frac{1}{M}\sum_{j=0}^{M-1} x[i+j] \tag{2.4}$$

While an MA-filter is easy to implement, it is important to know that an MA-filter typically leads to loss of data when applied to a dataset. The data loss occurs due to the filter being unable to perform the same averaging calculations on the last indexes as the window shrinks when it is smaller than M [19].

2.7 Existing work and starting point

When Axis tested their global shutter cameras in a scenario where their preexisting rolling shutter cameras were present, they noticed that the global shutter's strobe caused artefacts appearing as white bands in the image of the rolling shutter cameras. As seen in figure 2.7 which is a picture of the laboratory, the image is affected by a white band which will be referred to as a "strobe band". The strobe band has a certain velocity with which it moves across the image, and a width that depends both on the camera's exposure time and the duty cycle of the strobe. "Width" is used because of the exposure diagram representations, see figure 2.8. In figures 7.1 and 7.2 in Appendix the laboratory can be seen without and with a dimmer strobe [13].

The Product Imaging Engineering (PIE) team at Axis worked on a proof-of-concept solution for the issue. Their work provided a starting point and showed that the issue could be addressed in software. It was however limited in scope, only tested in a controlled laboratory environment, limited to a linear and locked exposure and only for a strobe with a frequency of 25 Hz. Their work is described next [13].

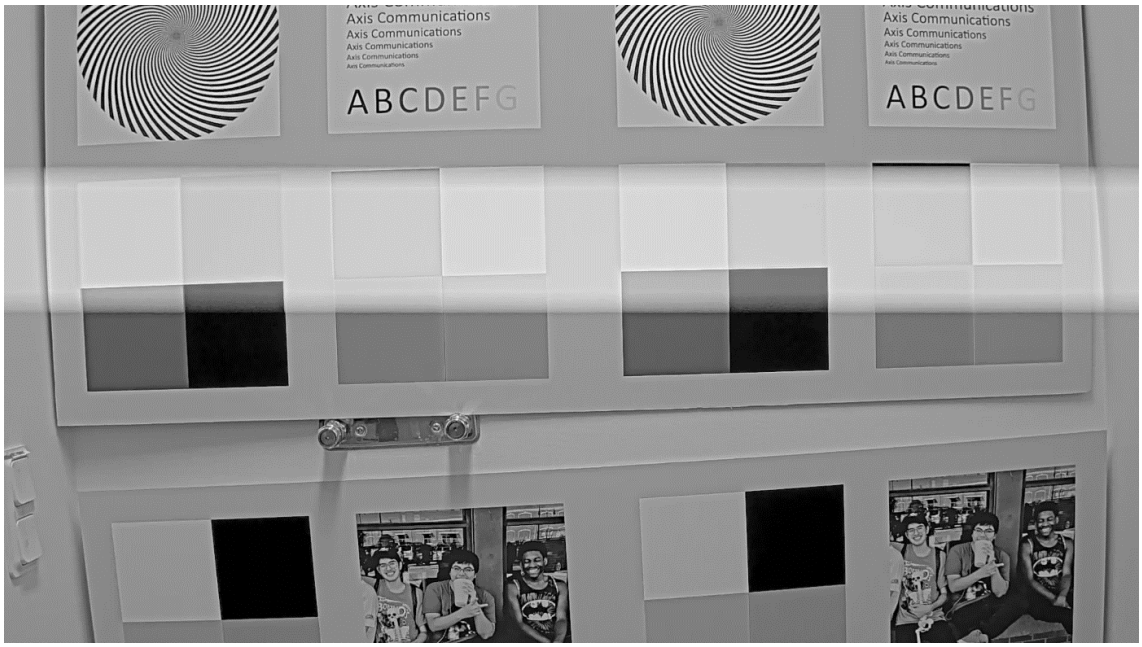

Figure 2.7: Shows a strobe artefact in the form of a bright band across the image.

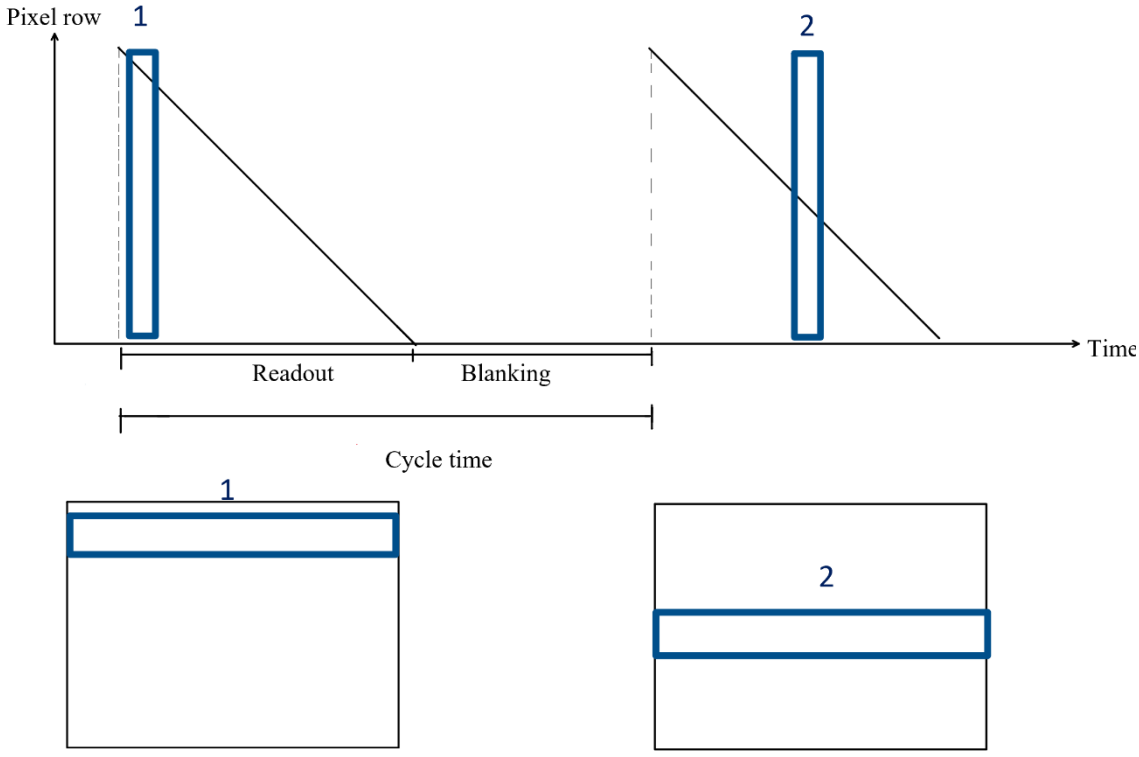

Figure 2.8: Illustrates an exposure diagram of how an unsynchronized strobe moves across the image. The movement is exaggerated to show how the strobe moves from one frame to the next.

Before explaining their work, an explanation of the data manipulation is required. As can be seen in figure 2.9, the cycle time from figure 2.8 has been replaced by N_{max} which is an integer representation of how many rows that fit into one cycle time. This means that everything in a cycle such as the readout and blanking can be interpreted with integer values and distances. N_{max} can be calculated using the following formula,

$$N_{max} = \frac{1}{fps \cdot t_{row}} \tag{2.5}$$

where fps is the camera frame rate and t_{row} is the time it takes to readout one row, also known as row time. The row time was measured at $1.33333 \cdot 10^{-5}$ s for the Axis P1385 with an accuracy of 5 decimals as that was how many decimal points that could be measured. As an example, N_{max} for 25 fps can be calculated to $\frac{1}{25 \cdot 1.33333 \cdot 10^{-5}} = 3000.0075$. As earlier mentioned, N_{max} is discretized and everything after the decimal point is removed which means that the final N_{max} is 3000. Furthermore, the readout can now be expressed as the height of the frame which is 1080 for a 1920×1080 resolution. This results in the size of the blanking window being 3000 – 1080 = 1920 [13].

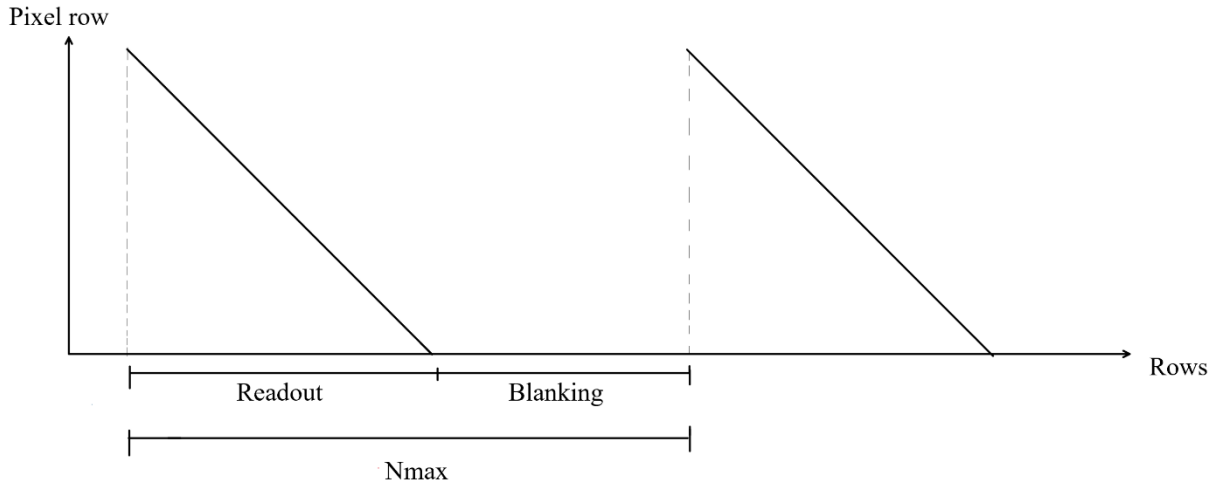

Figure 2.9: Shows an exposure diagram but with discrete values in the form of how many rows there are in a cycle and without the exposure time as it is short relative to the readout.

2.7.1 Image data collection and manipulation

To begin with, the camera collected the pixel data for every frame. The raw pixel data from the sensor-block came in the form of three separate vectors, representing the pixel x-axis, y-axis and the (x, y) pixel pairs (γ-index), this is due to how the sensor backend is designed. The data was then stored in a one-dimensional vector representation of a two-dimensional image, as described in 2.6.1. A camera with a 1920×1080 resolution has 2 073 600 pixels, and to avoid an overabundance of data, the 1920 columns were sampled by a factor of 20 down to 96 columns. As this is a row-centric matter, all the 1080 rows were kept. The data was then fed into the first method called "Find Strobe Bands" [13].

In this method the column values for every row were used to calculate row medians and appended to a vector with a resulting size of 1080 values. To detect the strobe band, the entire image median intensity value was calculated by performing a median calculation on all pixel values. If a row had a median intensity more than twice the image median, it was designated as a "strobe line candidate". If more than 20 such line candidates followed in a row, it was considered a strobe band, and its centre was calculated. The centre position of the band was then returned from the method and used to synchronize the camera's frame rate to the frequency of the strobe [13].

2.7.2 Frame rate synchronization

As could be observed in figure 2.8, the strobe moved across the image, which meant that the frame rate of the camera and the frequency of the strobe differed. To synchronize these, the centre position of the detected band in the current frame was compared to the centre position of the band in the previous frame to calculate how many rows the band moved between two frames, or the number of rows the strobe moved in one cycle n_{rows}. This could be either positive or negative depending on if the strobe was moving from the top to the bottom or vice versa. This was calculated with,

$$n_{rows} = x_n - x_{n-1} \tag{2.6}$$

where x is a band's position in the current (n) and previous ($n-1$) frame. The strobe period expressed in the number of rows, n_{strobe}, was then calculated as,

$$n_{strobe} = N_{max} + n_{rows} \tag{2.7}$$

where N_{max} is the number of rows between the start of one exposure to the start of the next one. This value was then fed through an IIR-filter, as there was a certain amount of variance in strobe frequency,

$$n^{\,IIR}_{strobe} = \alpha \cdot n_{strobe} + (1-\alpha) \cdot n^{\,IIR}_{strobe} \tag{2.8}$$

where $n^{\,IIR}_{strobe}$ is the filtered n_{strobe} value and α is the learning rate that determines how precisely the frame rate is calculated.

The IIR-filter leverages past information to enhance the signal quality and gives the ability to make precise adjustments to the target frame rate. To leverage an adaptive learning rate, α was set to $\frac{1}{N_{frame}}$ which was initially equal to 1 and decreased every iteration until it reached a value of 0.05, this value will be referred to as the α target value α_n in the rest of the report. N_{frame} denotes how many frames that has passed since the start. The cameras target frame rate was then calculated as,

$$fps = \frac{1}{n_{strobe}^{IIR} \cdot t_{row}}. \tag{2.9}$$

Now the camera frame rate converged to the frequency of the strobe. This finally resulted in the strobe being stabilized somewhere in the image now that the camera frame rate was synchronized to the strobe frequency as can be seen in figure 2.10. Furthermore, a "stability threshold" checked when the strobe moved less than 3 rows in between frames which was considered stable, and the current frame rate was stored in a vector. When 200 of these stable frame rates were collected in the vector called "minimum number of stable frames", the last recorded frame rate was set as the final and a delay to the exposure time was calculated. [13].

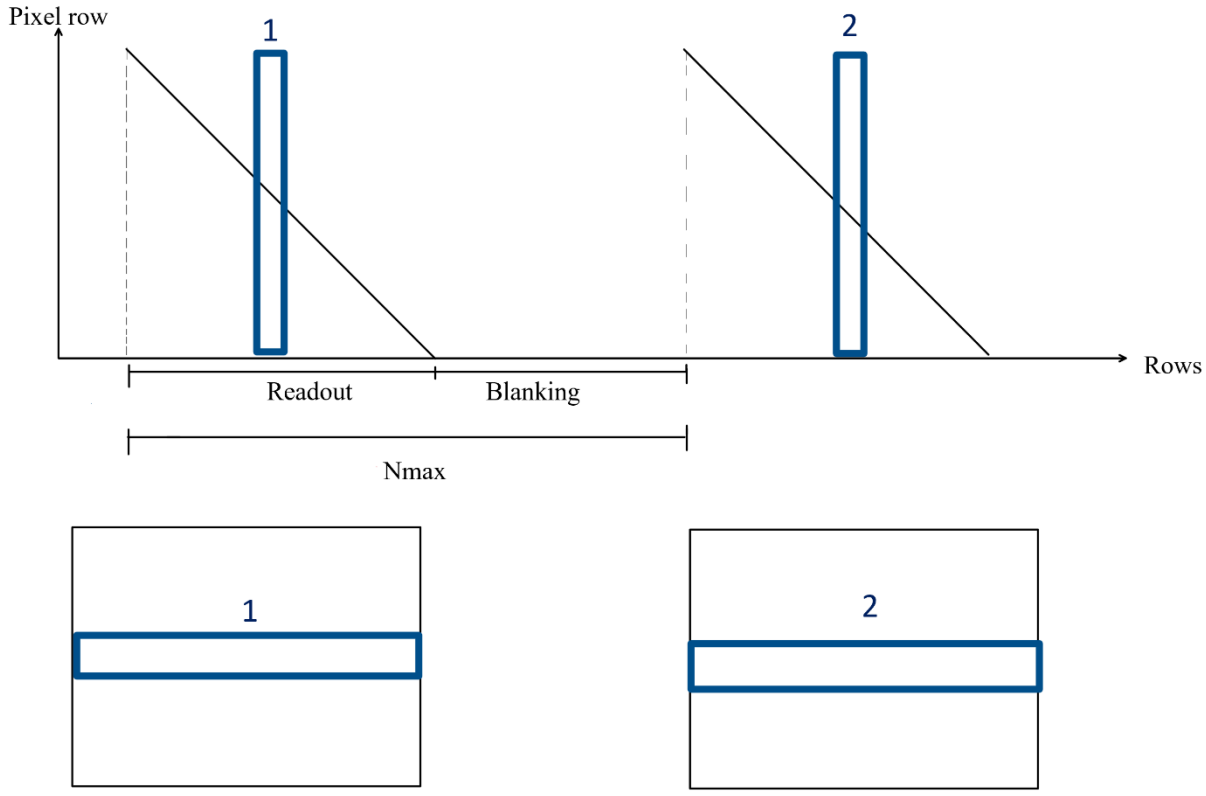

Figure 2.10: Illustrates how when the camera frame rate is synchronized to the strobe frequency, the strobe remains in the same position from frame to frame.

2.7.3 Delaying the exposure

Once the strobe band was stable, and after a certain amount stable frames, a delay distance of where to put the strobe in units of N_{max} was calculated based on the strobe band's width and position in frame using the following,

$$Delay_{distance} = \frac{N_{max} - readout}{2} - x_n, \tag{2.10}$$

where the band position x_n was returned from the Find Strobe Bands method, and the mid-point of the blanking window was calculated as $N_{max} - readout$. This in combination with the row time was used to calculate a delay of the next exposure which moved it to the when strobe signal was low, figure 2.11 illustrates this [13],

$$Delay = Delay_{distance} \cdot t_{row}. \tag{2.11}$$

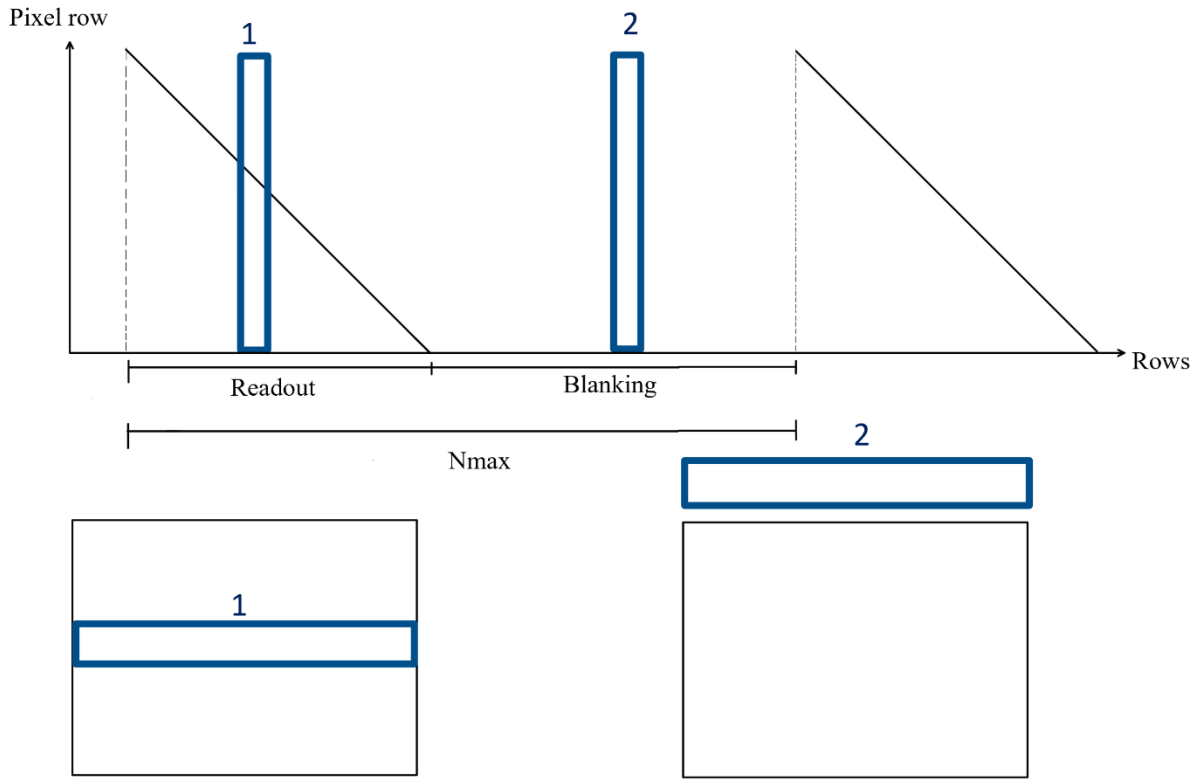

Figure 2.11: Shows how first synchronizing and then delaying the readout results in the strobe pulsing outside of the exposure window, resulting in a clear image.

Theoretically, this should have been enough to keep the strobe out of the image, but as explained earlier in this section, the frame rate was calculated using discrete values that can only be precise to a certain point. Therefore, the strobe would eventually move back into the exposure window because the camera frame rate never exactly matched the strobe frequency [20].

Once the algorithm calculates a frame rate matching the strobe frequency, a specific integer value of N_{max} is calculated as well. The time it takes the strobe to move back into the exposure window depends on how well the frame rate and N_{max} correlate. E.g. If the strobe has an exact frame rate of 25 fps, N_{max} is calculated using equation 2.1,

$$N_{max} = \frac{1}{25 \cdot 1.33333 \cdot 10^{-5}} = 3000.075 \ldots \quad (2.12)$$

However, since at least an entire row of pixels must be read out which takes one row time, N_{max} must be an integer which results in the decimals being removed from the calculated value. Now N_{max} is 3000 and is then used to calculate another frame rate with,

$$fps = \frac{1}{3000 \cdot 1.33333 \cdot 10^{-5}} = 25.000625 \ldots \quad (2.13)$$

As seen above, the calculated frame rate at the specific value of N_{max} is almost but not quite the actual frame rate of 25 fps due to the discretization of N_{max} as illustrated in figure 2.12 [20].

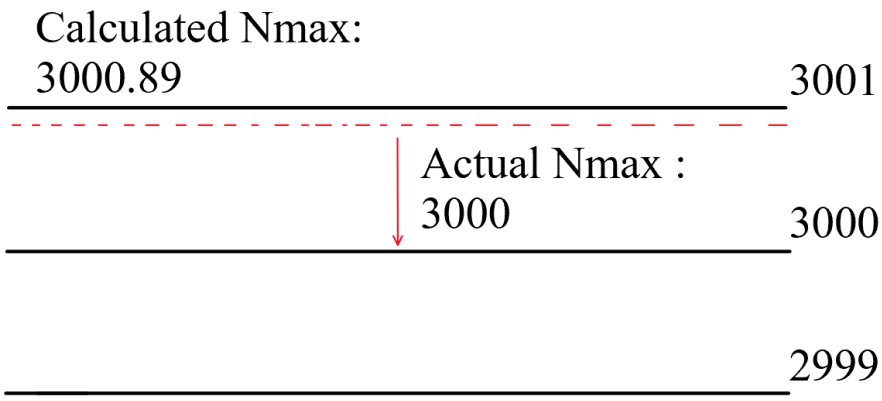

Figure 2.12: Illustrates that N_{max} only assumes rounded down integer values.

The team at PIE had discussed ways of addressing this issue where one was to use a detection algorithm and delays in order to "push" the strobe out of the exposure window when it eventually moved back into frame, which this report will focus on. They also raised three problems for future work. How are subtle bands in the form of thin, dim or irregular bands dealt with? How to differentiate between the strobe and a fixed band in the image? And can the band detection be reworked to detect edges instead of the width of the band? This will be further elaborated on in chapter 3: Method.

3 Method

As explained in 2.7.3, the strobe found its way back into the frame and with the initial code it would have to be resynchronized and eliminated every time. After discussions it was decided to implement and evaluate an algorithm that "pushes" the strobe out of frame when it reappeared in the top or bottom of the image.

However, before tackling the drifting strobe, an overhaul of the Find Strobe Bands method was preferred as the current version had to collect "strobe line candidates" where there needed to be more than an arbitrary value to be considered a strobe. It was also dependent on the width of the strobe band to return its centre. To avoid that, edge detection in the form of a large change in the median value of a row compared to the same row in the background of the image was used. This allowed for a band's start and end to be identified, the centre could then be calculated whilst ignoring the width. This method still returned the centre of a band which in turn was sent to be synchronized and eliminated.

After it was synchronized, by matching the camera frame rate to the strobe's frequency, and eliminated, by delaying the next exposure, the method to counteract the drifting strobe called "Maintain Delay" was implemented. This method continuously checked the top and bottom row of the frame to see if there was a large change in pixel intensity and if it was larger than a threshold dependent on the background, it triggered a delay, pushing the strobe back into the blanking window. This method is explored in further detail throughout chapter 3. Figure 3.1 on the next page shows a flowchart overview of the entire algorithm.

Principles and methods kept from the original code was the one for retrieving the pixel data, the principle of returning the centre of a strobe band, the algorithm to calculate how many rows the band moved between frames and to synchronize the frame rate to the strobe frequency, and lastly the method to calculate the required delay depending on the band's position in the frame. All of these are explained in 2.7.

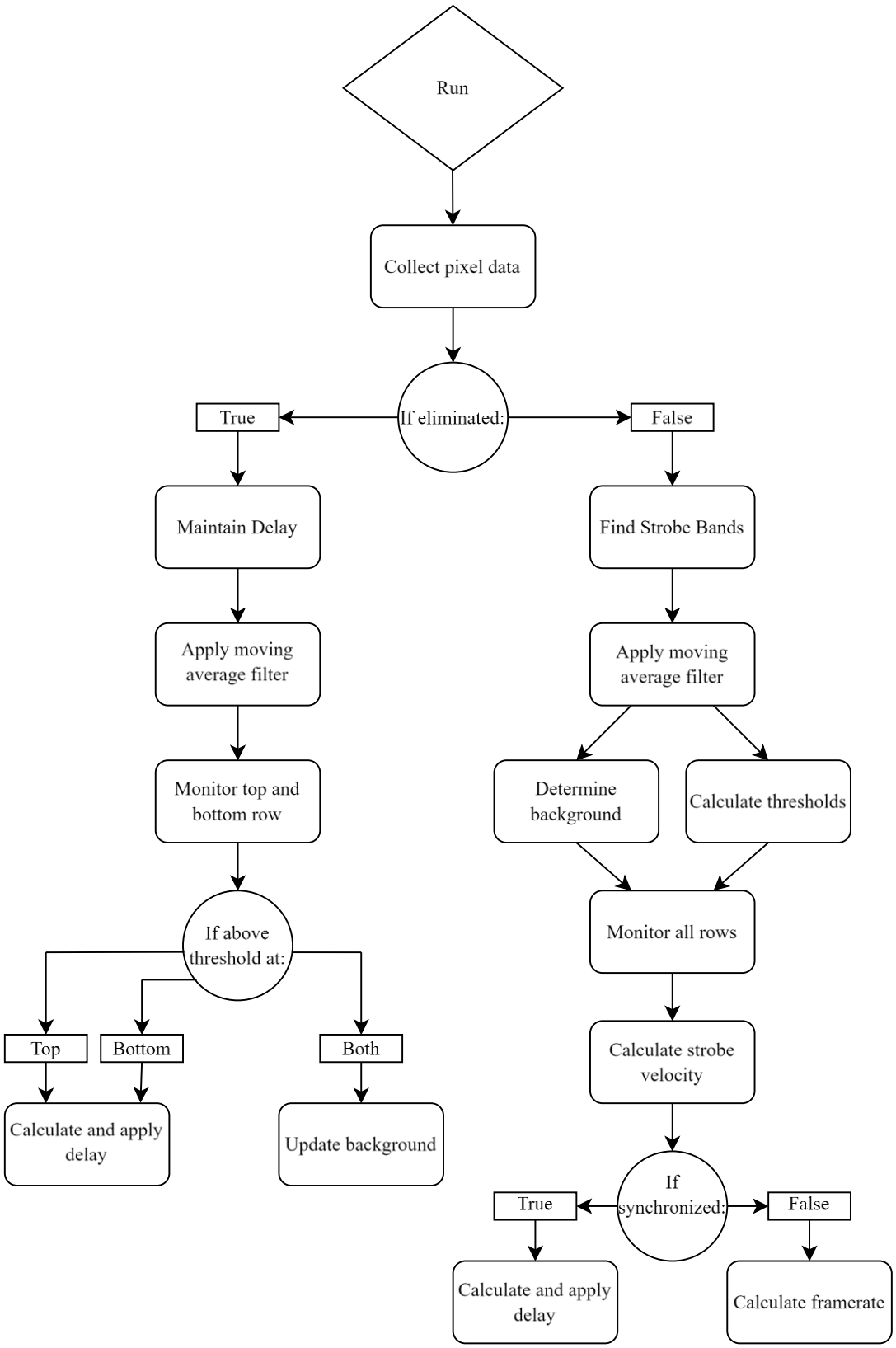

Figure 3.1: A flowchart representing the interaction between the algorithms, where Run is called every frame and where either Find Strobe Bands or Maintain Delay is called depending on the state.

3.1 Code algorithm

The entire code ran from a method called “Run” which was called upon once every capturing of a frame. Most development was in C++ with some minor changes to the LUA code which tied all the different blocks of code together. Due to the proprietary nature of Axis source code no code snippets will be presented in this report, furthermore only the parts critical to this thesis will be explained.

3.1.1 Collection of image data and calculations

The handling of the raw pixel data of every frame remained the same as explained in 2.7.2. Then in the Find Strobe Bands method, the median value of each row was calculated by collecting the 96 column values in every row and performing a median calculation. The saturated row median intensity values were recorded to 4095 at peak brightness and 215 in total darkness. By calculating a median for each row, the number of datapoints were reduced and could be placed in a vector. Since C++ contains well optimized packages for vector processing, this aided in keeping the algorithm fast enough to be ran 30 times per second.

A comparison between the median and mean values for every row was made by slightly changing the code. It was then observed that spikes in the pixel value would occur in the image from light sources such as ceiling lamps and streetlights when using mean values but were not present when using median, therefore, the median was used from this point. This was tested in 3.3.1.

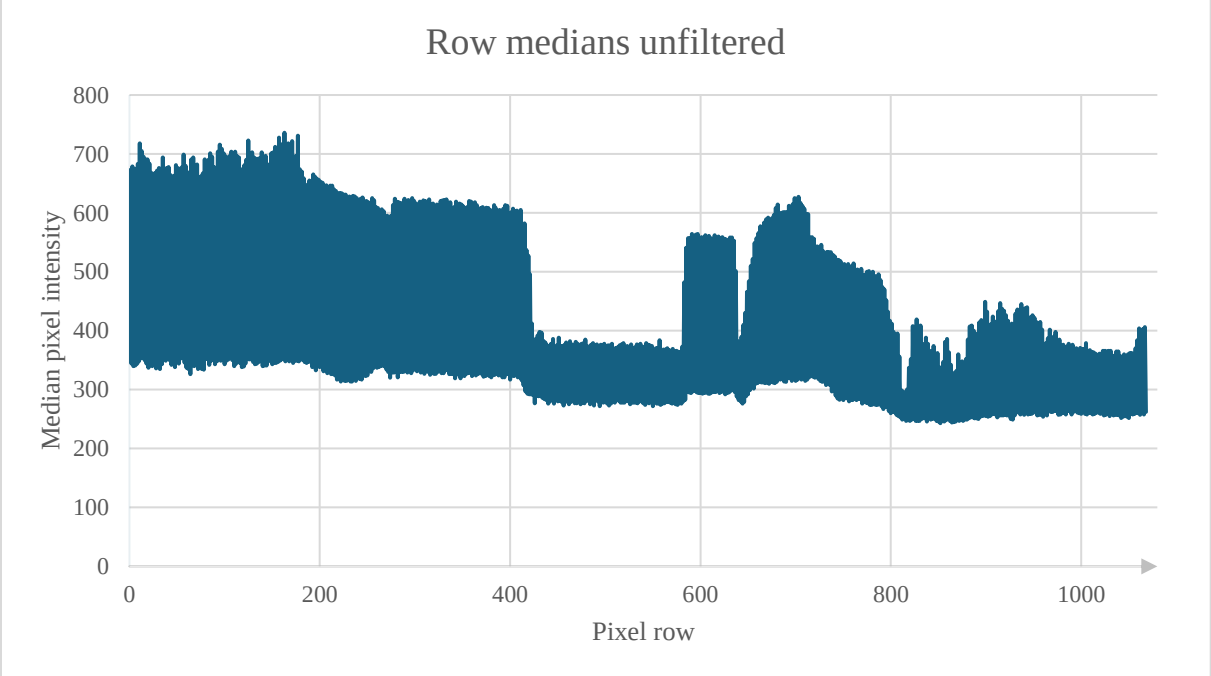

Figure 3.2: Shows the median pixel intensity for each row of a typical background in a laboratory setting with the ceiling light on.

As can be observed in figure 3.2, the pixel intensity data was noisy, and the pixel intensity value was dependent on the objects in frame. It was therefore difficult to reliably detect the strobe artefact from this data only, especially if the artefact was subtle. Another issue was the risk of a static bright line such as the horizon being identified by the algorithm as a strobe. This was potentially something the algorithms described in 2.7 would have triggered on, which could have had a detrimental effect on the algorithm's functionality, this issue was addressed with the method described in 3.1.4.

3.1.2 Application of a moving average filter

As can be seen in figure 3.3, utilizing an MA-filter significantly reduced the amount of noise in the data. This minimized the risk of false positives in the detection algorithms, thereby improving the accuracy of the calculations. However, this came at the cost of less precise data, especially towards the end of the dataset depending on the window size. This trade-off was deemed acceptable given the specific need for noise reduction. As described in 2.6.5, an MA-filter typically reduces the number of data points. However, the detection algorithms required vectors of the same size. Therefore, it was decided after initial testing to modify the filter to calculate a mean of the last indexes which did not fit in the window size and copy this mean for the last values. This can be seen towards the end of the graph in figure 3.3 and 3.4.

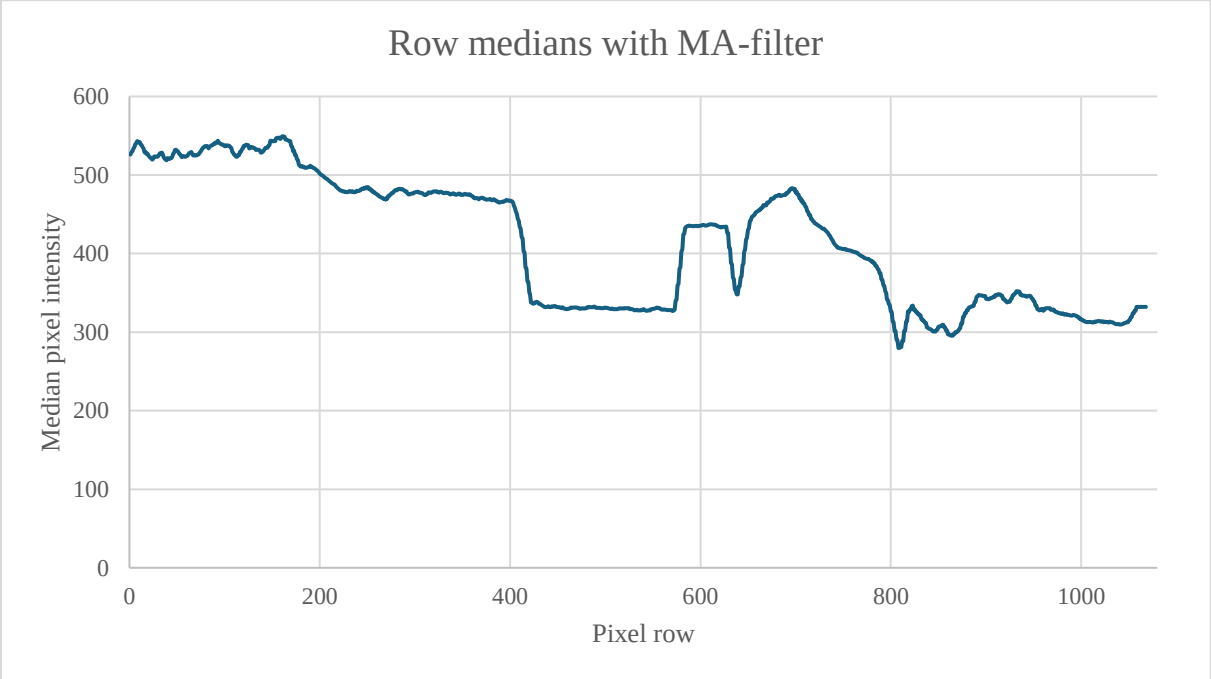

Figure 3.3: Shows the filtered median pixel intensity for each row of the same background as in figure 3.2 with a filter window size 12.

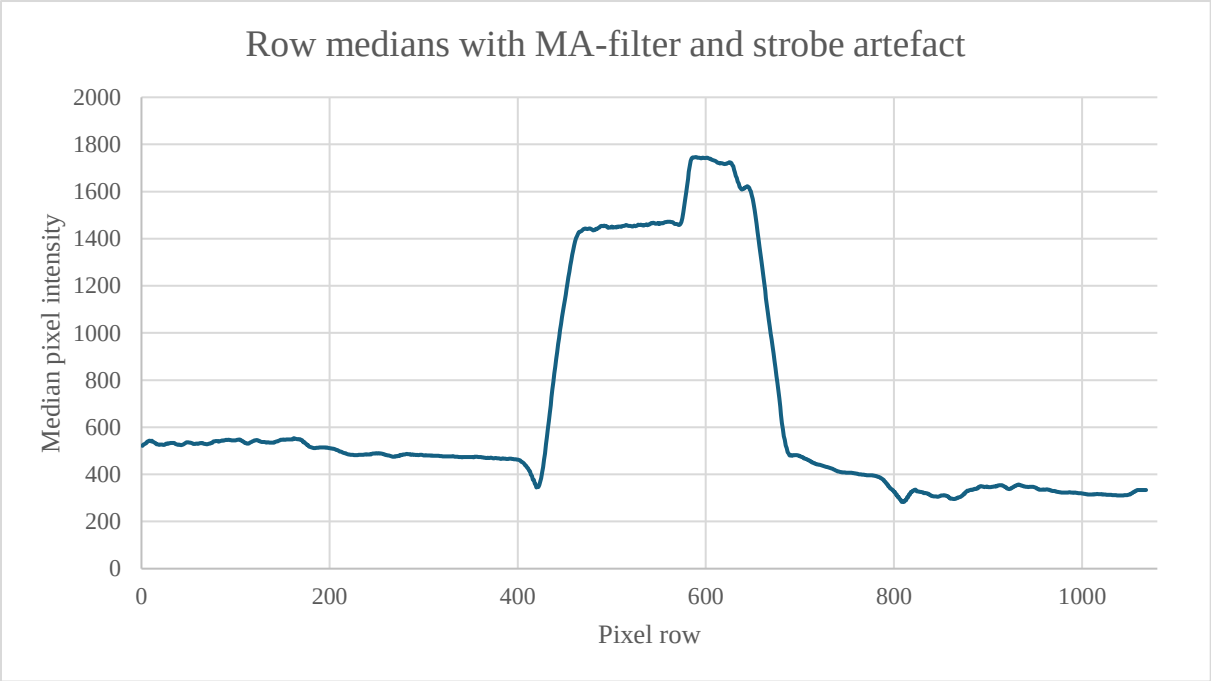

Figure 3.4: Shows the filtered median pixel intensity for each row with the strobe artefact present against the same background as in figure 3.3, and a filter window size of 12.

As seen in figure 3.4 the strobe artefact was distinguishable from the background despite the signal being filtered. A window size of 12 was chosen as it divides evenly into 1080, this helped remove noise and spikes but did not interfere with the strobe as it was held at a fixed width. Although the MA-filter causes in this case the last 12 values to be identical, the strobe artefact is wide and intense enough for this to not matter. In 3.1.3-3.1.6 methods and algorithms to take advantage of this are explained.

3.1.3 Storing the background

When using a fixed exposure, identifying a frame without the strobe artefact was possible as the camera frame rate was initially set to 27.5 fps to differ from and to be able to work at both 25 and 30 fps. There were brief windows of time to store the background when the strobe was in the blanking window as it scrolled across the frame. To leverage this, the frame rate calculations were deliberately ceased for 120 frames at the start to let the strobe scroll across the frame several times. When the strobe was out of frame, the median pixel intensity of the entire image was calculated and compared to the previous frame's image median. If the current image median was lower than the previous image median, the background was stored. Figure 3.3 in 3.1.2 shows a typical background image in a lab setting. The corresponding vectors containing row medians for the top and bottom of the frame were also stored and used later in the Maintain Delay method, which is explained in 3.1.6.

When automatic exposure was on, it was almost impossible to reliably determine a background image, as when the strobe artefact was in frame the exposure time was decreased to compensate for the brightness of the strobe. Once the strobe was out of frame the exposure time then increased. This led to the image median of a frame containing a strobe being lower than the image median of a frame without the strobe, which resulted in the stored background containing a strobe artefact. Therefore, it was decided to lock the exposure during the Find Strobe Bands method to simplify the process of matching the frame rate to the frequency of the strobe. Automatic exposure is further discussed in 3.1.7.

3.1.4 Comparison of the current frame to the background

With a stored background in hand, edge detection was implemented in the form of a comparison of the current frame to the background by subtracting the background row values from the current row values. This comparison was stored in a variable that was appended to a vector with the difference values. Figure 3.5 shows the difference values without the strobe and 3.6 with the strobe. Every element in this vector was then compared to a threshold which was calculated using an adaptive thresholding technique as explained in 2.6.4, which will be explained in 3.1.5.

As explained in 2.7.2, the centre of the identified strobe band was to be returned to the run method to calculate the strobe velocity and frequency, and to calculate a delay. But instead of storing rows as “line candidates” the algorithm now stored a vector of the rows with the first and last values larger than the threshold. This was then used to calculate the number of rows in a band and where its centre was.

A criterion for this to work reliably was that the stored background was close to or the actual background. E.g., if the camera was in an environment with sudden changes to lighting, or something large that moved, the stored background would no longer be the actual background. This was recognized as a problem and noted but because Find Strobe Bands only ran for a short while it was never a recurring issue in that method. And besides, smaller changes in the background such as an item or a person appearing did not exceed the threshold as it did not result in a large enough change in pixel intensity. This issue did however reappear in the Maintain Delay method which is explained later in 3.1.6.

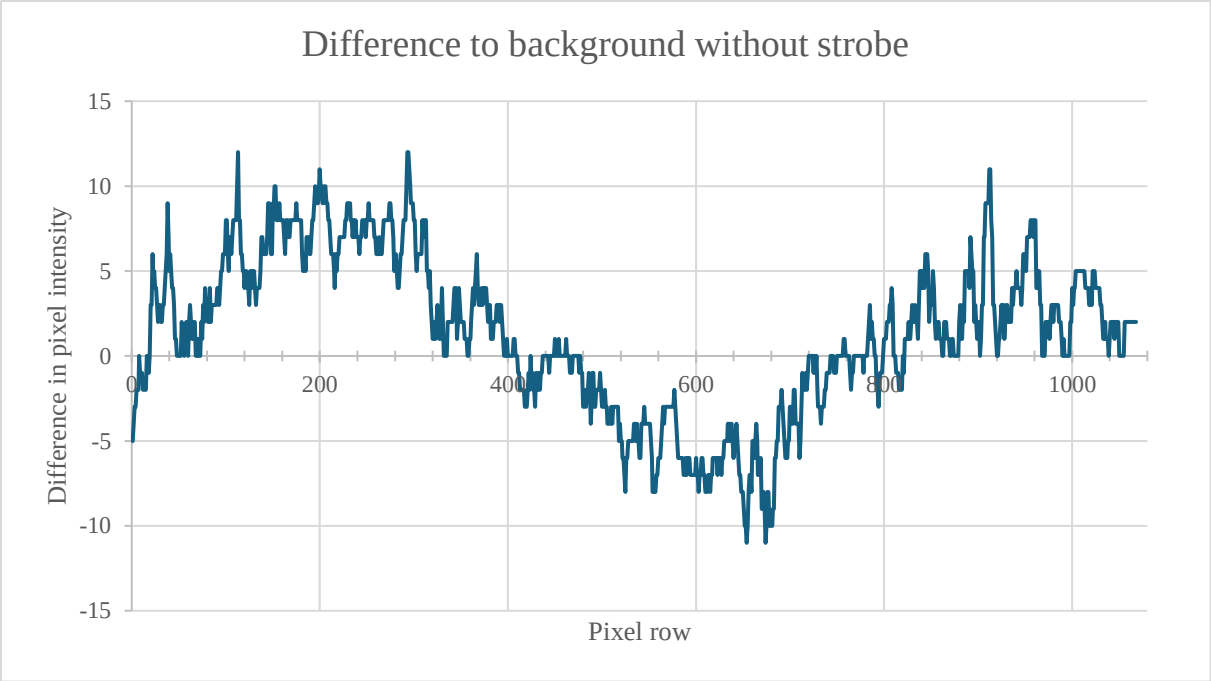

Figure 3.5: Shows the difference between the current frame and the background frame, without the strobe in frame, which is mostly noise.

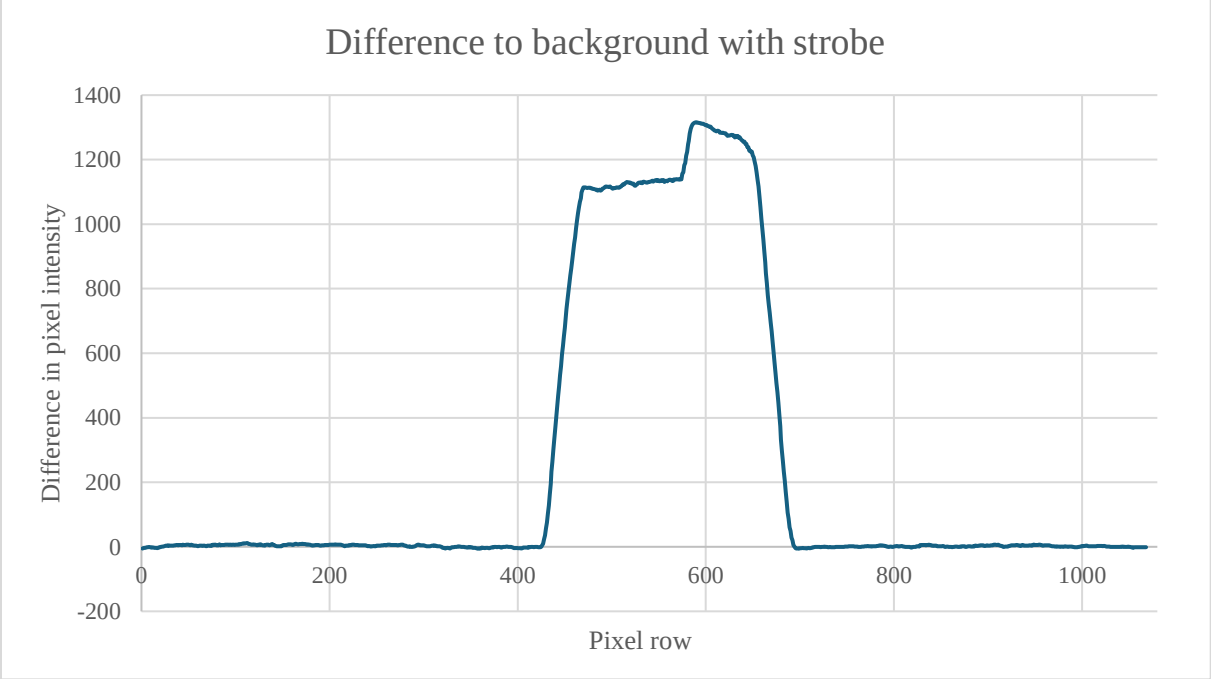

Figure 3.6: Shows the difference between the current frame and the background frame, with the strobe in frame.

3.1.5 Application of an adaptive threshold

Since the noise level in the image varied depending on what was pictured, a certain intensity level might be considered a strobe artefact in one image but be within the range of acceptable noise fluctuations in another. Therefore, an adaptive algorithm like the basic global threshold described in 2.6.4 was implemented with some important differences. Firstly, the algorithm in 3.1.1 calculated the row median and the relevant metric was whether the row median was above or below the threshold which means this threshold calculation was performed on one-dimensional data. Secondly, since the median value for each row was calculated, it was redundant to calculate the mean value of median values. Therefore, the choice was made to instead iteratively fraction the highest row median and compare it to the same value from the previous frame until the initial detection was complete. This helped to limit the computation time of the threshold algorithm since it had to calculate three different thresholds every frame. A minimum threshold was set at 40 due to the noise in the lab reaching intensities of around 20. This was performed when storing the background in 3.1.3

Three thresholds were calculated, and the first was used in the Find Strobe Bands method to differentiate the strobe from the background. To capture values where the strobe was both in frame and not, the algorithm was allowed to calculate a correct frame rate. However, at the same time as storing the background for 120 frames, the frame rate was set to the correct value minus one. This allowed the strobe to scroll across all the pixels of the image and guaranteed calculating similar thresholds every run. Then the largest pixel value in the difference vector was recursively stored which was then halved and designated as the threshold. This threshold was then used to synchronize and eliminate the strobe.

The second and third threshold was used in the Maintain Delay method to also differentiate the strobe from the background. Although this time only the threshold for the top and bottom of the image was stored. All the thresholds were calculated in the Find Strobe Bands method using the same method except that the top and bottom thresholds were divided by five for increased sensitivity. It was noticed during field testing that increased sensitivity was necessary for it to apply delay when wanted. A consequence of this was the increased risk of triggering on something other than the strobe, which was deemed a necessary trade-off.

3.1.6 Maintain Delay: Keeping the strobe out of the exposure window

Once a strobe artefact was detected, the frame rate synchronized and strobe moved out of frame, a second stage was initiated. In this stage, only the top and bottom row of the image was monitored. As explained in 2.7.3, due to the backend discretization of the readout time of each pixel row and possible small variances (jittering) in the strobe frequency, it would inevitably drift back into the image. Depending on the calculated frame rate, the strobe reappeared in either the top or the bottom of the image, an appropriate delay to the exposure window was calculated to “push” the strobe to the midpoint of the blanking time between frames. Checks such as only allowing delays if there was a change in exclusively the bottom or the top were implemented to verify that it was the strobe and not the background changing.

The reason for only checking the first row in the top and last row in the bottom was because they were the first to encounter the strobe and be obstructed by it as it moved further. A concern about noise and unintentional triggers arose. Although these concerns faded as only checking the first and last row was justified because of the rows already being filtered from noise by the median, and that this never occurred during observations.

Depending on if the calculated frame rate was slightly above or below the strobe frequency, the strobe either appeared in the top or bottom of the frame as shown in figure 3.7. The strobe reappearing and being delayed will from now on be referred to as “delay update” or just “update”. An appropriate delay was calculated in the following way: If the strobe appeared in the top of the frame, the delay was calculated as,

$$Delay = \frac{N_{max} + readout}{2} \cdot t_{row} \tag{3.1}$$

And if the strobe appeared in the bottom of the frame, the delay was calculated as,

$$Delay = \frac{N_{max} - readout}{2} \cdot t_{row} \tag{3.2}$$

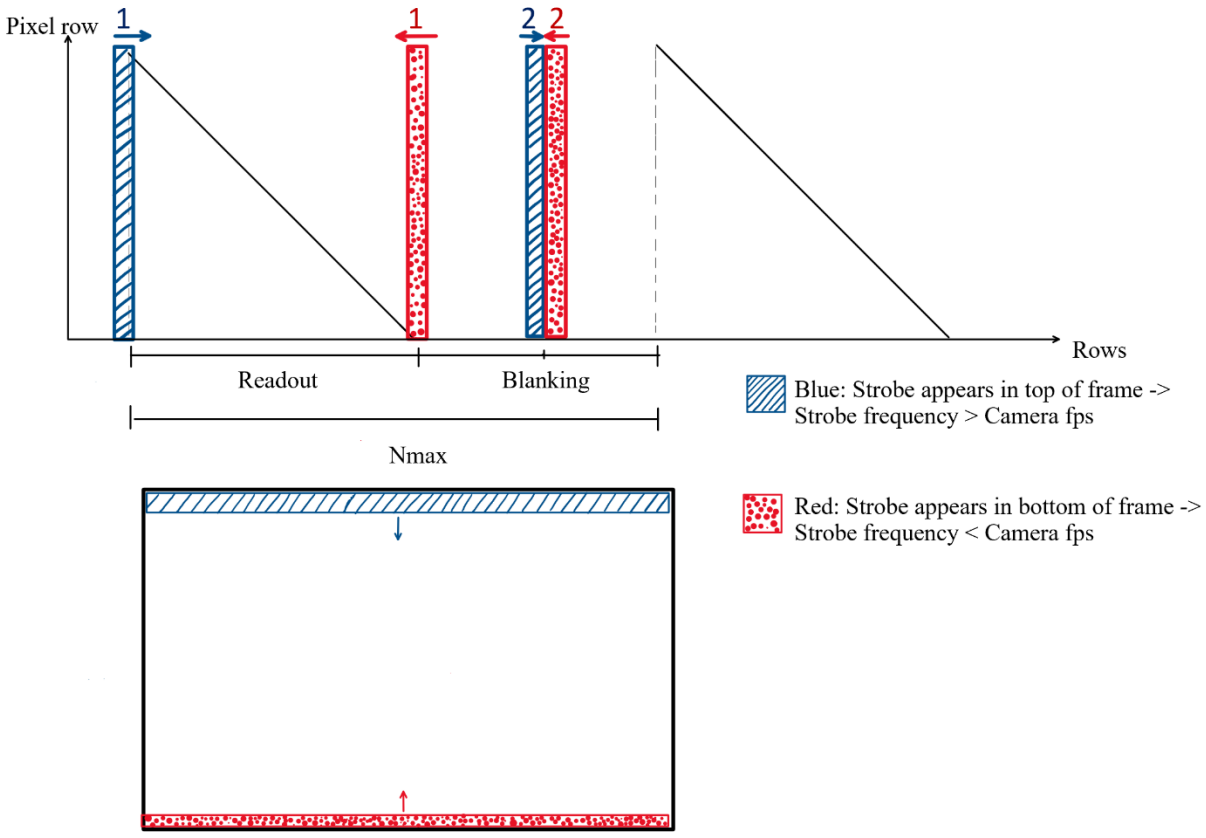

Figure 3.7: Illustrates how the Maintain Delay method applies a delay based on the strobe’s direction of approach.

A requirement for this to work properly was no large changes in the strobe frequency after the frame rate was set. If that happened, the algorithm would have to be restarted and the camera frame rate recalculated. A check for this was implemented in the following way: If the number of frames between two delays were less than the mean number of frames between two delays found when testing, the frame rate was deemed poorly synchronized, and the algorithm was restarted. How this number was decided upon is explained in 3.2.1. This was deemed necessary as the delay command sometimes failed to apply the correct delay, which resulted in the strobe being delayed into the image and would only disappear once it touched either edge. This could take a long time if the frame rate was well synchronized. This problem was exacerbated when too many delay updates occurred due to a poorly set frame rate. Therefore, as few delay commands as possible was preferred, and for this to be true, the frame rate had to match as close as it possibly could to the strobe frequency. This is further explored in 3.2 where tests of the α_n value (the target value of the IIR-filter), frame rate and delay updates were performed.

Another feature of the Maintain Delay method is that it monitors large changes in the background by checking large changes in both the top and bottom of the image. As the strobe only affects either the top or bottom, a change in both was not the strobe and therefore no delay was applied, but the background and threshold was updated.

The requirement for updating the background was that the maximum and minimum value out of 1/36 of the rows in both the top and bottom had to exceed a specified value which was set to be half of the current threshold at that edge. If the current image was brighter than the stored background, the difference values were positive, but were negative in the reverse scenario. The absolute value of the maximum and minimum difference value was used to avoid negative values. After waiting 80 frames to guarantee that nothing else was happening in the code, the background vectors for the top and bottom rows were then updated by performing the same calculations as mentioned before in 3.1.1.

To avoid updating when the strobe was about to reappear, the background could only update if a counter that counted the number of frames since the last update was 2000 smaller than the recorded number of frames between updates as this number was almost the same for the same frame rate. E.g. if the mean number of frames until an update was recorded at 12 000, the background could only update if the counter since the last update was smaller than 10 000 to allow for some variance which was tested and is shown in 4.1.5.

After finishing the background update, the top and bottom thresholds were updated by storing the previous background, this was then used to calculate a factor of how much the previous and current background differed. This value was then multiplied with the threshold values to set the new thresholds. During the update, the algorithm's ability to apply delays was interrupted. To balance the background in the long run, a counter was implemented to count a certain number of frames for when to update the background. The lower limit was set to half of the number of frames between updates to avoid updating the background when the strobe was about to return. The counter was reset every time a delay was applied.

3.1.7 Functionality with automatic exposure

As described earlier in 3.1.3, the algorithm was initially developed using a fixed exposure time which made finding a background and comparing the current frame feasible. However, when the algorithm was started whilst the automatic exposure mode was active, it was unable to compare to any background because of the automatic exposure constantly changing the pixel values when the strobe scrolled across the frame at a fast pace.

There were two possible implementations to counteract this. The first one discussed was to use a fixed exposure time when finding and eliminating the strobe and to then reenable automatic exposure when entering Maintain Delay. This way the background could be found and used to compare with. This implementation worked well as the correct background was stored for Maintain Delay.

A problem with this implementation was that the fixed exposure time was not necessarily the same as the automatic exposure time. Because of this, when the automatic exposure was reenabled, the exposure time stored with the background and the exposure time of the current frame (which was the background as the strobe was eliminated) could differ. A solution to this was to implement an “exposure time factor” which was calculated using the current exposure time divided by the stored background’s exposure time. This way, if the exposure changed during the comparison in Maintain Delay, the values would scale accordingly.

This worked in some conditions but failed in others where the background was too dark which increased the exposure time to a degree that the strobe band was too large, sometimes covering the entire image. Another effect this had was rendering the strobe dim if the exposure time increased as the background intensity also increased. The algorithm was therefore unable to detect the strobe as its effect diminished. This also caused the image brightness to change which sometimes triggered the delay even though the exposure time factor changed.

Therefore, it was finally decided that automatic exposure added too many factors of which to consider, and a special exposure sequence was implemented. First, before detecting the strobe during Find Strobe Bands, a special 3 ms exposure time in automatic mode was used to obtain a visible image in both dark and bright environments. As sometimes when moving from a darker scene to a brighter or vice versa, the image was completely white or black. Therefore, allowing the automatic mode to adjust sensor gain and lens aperture resulted in a discernible image. Then, after waiting 120 frames for the exposure algorithm to settle, it was locked in “hold” mode where gain and aperture were also locked. Locking these three functions guaranteed a fixed strobe width at 275 ± 10 rows for the 1080p camera without camera induced brightness changes, which allowed Find Strobe Bands function in all tested environments.

3.2 Laboratory testing

Laboratory tests were performed continuously throughout development by adhering to an agile workflow. Having the ability to stream video from the camera and immediately upload new code with Axis' software building tool, eased the process of quickly testing new ideas and concepts. By using the building tool, the time-consuming process of recompiling the entire code base was avoided. This saved several minutes of compilation time each upload when something new needed testing.

Data was conveniently collected using Axis' data collection software "Imagetool". This software had an interface that displayed data such as variables and vectors as graphs that were easily interpreted. This data could then be saved and used to plot graphs, which is what most of the graphs in this report are created from.

The physical laboratory consisted of an enclosed, windowless room with approximately 150 cm between the camera and the motive. The camera used for testing during all of development was the P1385 with a resolution of 1920×1080. The settings used were the following: Wide-Dynamic-Range (WDR) turned off, the base frame rate set to values between 25 and 30 fps and exposure both locked and automatic during different stages of the algorithm. Later, the P1387 and P1388 were introduced, and further testing was performed, although most tests were limited to the P1385.

A Raytec Pulsestar VTS-20 stroboscope was used to emulate a nearby global shutter camera's strobe light. The strobe has an internal pulse trigger and a software which was used to set specific pulsing frequencies. It did however have a high variance which led to an unstable strobe frequency that jittered, meaning that it bounced up and down in spot when synchronized. Considering that a global shutter camera's IR-strobe would be synchronized to its clock, it is assumed that one of those strobes would have considerably less jitter. For testing purposes, the jittering represents an absolute worst-case scenario which is desirable, as if it works then, it has a high chance of working better with a more stable strobe pulse trigger. To force the strobe into the frame, the strobe's software was used to control its duty cycle and frequency. This was used to slightly increase or decrease the strobe frequency to simulate the strobe moving into frame. Furthermore, the strobe was pointed away from the cameras to diffuse the light.

3.2.1 Testing the accuracy of the frame rate calculations

As mentioned earlier in 2.7.2, the learning rate α controlled how well the frame rate matched the strobe frequency. As explained in 3.1.6, as few delay updates as possible were preferred as this improved algorithm stability. Therefore, it was imperative that the frame rate converged and matched as closely as possible to the target in as many runs as possible. This was done by altering the learning rate target value α_n, the tests will be referred to as the α_n tests. During the tests, the camera never moved, and the background illumination was constant from another camera's built in IR-backlight (due to the ceiling light breaking during these tests).

Initially, α_n was kept at 0.05 until it was time for tuning. It was then decided to test different values ranging from 0.01 to 0.5 by changing α_n, letting the frame rate stabilize and applying a delay. Smaller and larger values than 0.01 and 0.5 respectively worked poorly and was excluded from testing. A test was then concluded when the first delay update occurred as soon as the strobe appeared in either the top or the bottom of the frame. The number of tests per α_n value was also increased in order to narrow in on the better values. The goal was to see as small of a difference as possible in the calculated frame rate between runs whilst maximizing the number of frames between delay updates every run. Therefore, the mean of the frame rates, the standard deviation of the frame rates, the number of frames between updates, and the standard deviation of the number of frames between updates were primarily calculated.

To evaluate which α_n values where suitable candidates for further testing, an evaluation score was calculated as follows in equation 3.3. This was used as a basis to determine what α_n values to test further.

$$Evaluation\ score = \frac{Mean\ frames\ until\ update}{Mean\ frames\ until\ stable \cdot Frame\ rate\ standard\ deviation} \tag{3.3}$$

Once a candidate for the optimum α_n was found, the middle value of the two best result was tested. This was done to increase precision and narrow in on the best value. All the following tests were performed with a sample size of ten. This was chosen out of necessity due to the long runtime of the tests, as well as the many different α_n tested. When satisfied with a smaller interval of α_n, more in-depth tests were performed by increasing the number of decimals gathered and the number of tests to 50 at each α_n. In addition to the values previously mentioned, the number of frames it ran for during Find Strobe Bands, the top and bottom thresholds values, N_{max}, the number of times it failed to apply a delay, and if it appeared in the top or the bottom was recorded.

When satisfied with a range of values, these were tested further to narrow in on the one to use. Values in the range of 0.015 to 0.04 resulted in the highest mean numbers of frames between updates and a narrower range of calculated frame rates between tests. This range was tested at intervals of 0.005. Further precision was deemed unnecessary due to there being few changes in behaviour in between tests and a lack of time.

The test was automated by altering the code to reset the first time the strobe reappeared in the Maintain Delay method. A reset state function was created to reset all essential variables whilst retaining vectors with the desired calculated values. When the decided upon number of values in each vector were collected, they were exported and examined using Excel. The tests were performed on both the P1385 and the P1387 but mostly on the P1385 as the P1387 handled the algorithm a bit worse. No tests were performed on the P1388 as another issue in the camera prevented further testing. Even though the second round of testing was automated, testing each α_n value took on average eight hours to run. Therefore, only six values were tested on each camera. The results of these tests are presented in 4.1.1.

After an α_n of 0.03 was chosen, another way of setting the frame rate was implemented. As explained earlier in 2.7.2, when the frame rate was considered stable, it was appended to a vector to determine a minimum number of stable frames. As this vector contained values which all represented the actual frame rate, the mean of the last half of these values was calculated and used as the final frame rate. There was a stabilization period before the target value was reached as seen in figure 4.6 in 4.1.2, which was why only the last half was used. This resulted in a smaller range of calculated frame rates when collecting values in further tests. As these frame rates were closer matched and more predictable, this behaviour was preferrable and used in the final implementation. This is presented in 4.1.2.

Additionally, a test with a function generator was performed to solidify the prediction that a tuning with a less stable strobe would aid in more stable cases. With the function generator producing the trigger signal, the strobe was very smooth and never jittered compared to the strobe's internal trigger. The function generator was set to pulse-mode at 25 Hz with a duty cycle of 5%. Since an α_n value of 0.03 had previously been the most reliable it was used in these tests as well. Additional tests with the function generator were performed, but this time by using what was described in the previous paragraph. See the results in 4.1.3.

During the α_n test, the stability threshold was kept at 3 and the minimum number of stable frames was kept at 200 to reduce any changing factors. After an α_n of 0.03 was chosen, the stability threshold and the minimum number of stable frames was altered to test which stability threshold gave the best results and how many frames was needed to let the frame rate stabilize.

The decision was made to sacrifice some speed and let the algorithm run for a little longer in order for everything to settle, this was justified as the Find Strobe Bands method was supposed to only run once.

3.2.2 Delay applications during Maintain Delay

While in the Maintain Delay method, it was noticed that the algorithm sometimes failed to apply the delay correctly and pushed it into frame, despite the implementation of a frame counter to avoid it applying more than one delay. Therefore, a test was performed where the number of times a delay was applied, referred to as “updates”, and the times it failed to correctly apply a delay, referred to as “delay failures” was recorded. By then dividing the failures with the updates, the failure rate could be calculated. As mentioned earlier, the delay updates incremented when the strobe reappeared, and the delay failures incremented if the strobe reappeared in first the top and then the bottom or vice versa as this sequence was supposed to be impossible.

Another test was performed to see how much the strobe varied when forcing a fixed frame rate of 24.967 fps in the code. This test was performed to show the effect of the strobe’s internal variance on the frames between updates. 24.967 was chosen as it was close to the one the algorithm tended to calculate but far off enough to not take too long for an update to occur. The results of these tests for both cameras are shown in 4.1.4.

3.2.3 Testing thresholds and counters

There were a few counters in the program that halted operations to wait for certain things to settle. One of them, which ran for 120 frames, was needed to let the automatic exposure settle at the start of the algorithm. Another one which ran for 120 frames was to find a background and to let the three thresholds settle which needed a few seconds. Yet another was needed when updating the background during Maintain Delay as it could trigger too fast and catch something moving which would shortly after, not be in the background. This one ran for 80 frames. The final one ran for 24 frames to only allow one delay command when the strobe reappeared. Although the delay application counter could be at minimum 8 frames to avoid more than one delay triggering, it was tripled to completely avoid the risk. These counters were tested by rerunning the program and over and over and lowering them until it stopped working, they were then increased again.

The sensitivity of the top and bottom thresholds was changed by altering the divisor. When dividing by two and three, the algorithm struggled with noticing the strobe as the thresholds were too large. It worked better with four and five where four was used when wanting less triggers and five when in need of more sensitivity. These varied depending on test scenario.

The criterion for updating the background was also tested by starting with a change of 100. Then 50 and 200 were tested where it triggered too often at 50 and not enough at 200. It was then set as half the threshold at that edge, which worked in most tested cases.

3.3 Field testing

Field testing was primarily performed on Axis' premises, although when the algorithm was in an almost finished state, more tests in different environments were performed. The tests on Axis' premises consisted of tests in their underground parking garage as the lighting conditions were consistent regardless of the time of day. The same P1385 camera and strobe light were used as in the laboratory tests, with an additional P1387 to verify that the program worked on another sensor size. Furthermore, two battery packs with accompanying inverters were used to power the cameras and the strobe. As well as two separate ethernet switches, one used to control and monitor the cameras and one to control the strobe.

By using two separate power sources, flexibility was achieved in what various setups could be tested. E.g. in one test, the strobe was placed 20 meters away and pointed straight at the cameras. This was to emulate how a global shutter camera could be placed in relation to an existing rolling shutter camera. Moreover, the strobe was both pointed away from and towards the cameras to diffuse and concentrate the light respectively.

Many unexpected issues never observed in the laboratory tests appeared when performing field tests. And due to the inconvenience involved when testing in the field, an iterative process was implemented. This process consisted of performing a round of tests, identifying issues, addressing the issues, and testing them in the laboratory. Once the new or modified algorithms worked in the laboratory a new round of field tests commenced.

3.3.1 Parking garage

The parking garage was long and narrow which helped when testing at different distances. There were cars, pillars, pipes and especially lights in the ceiling that all affected the strobe's appearance in the data and the image. The tests consisted of altering the distance between the camera and the strobe, the angle the strobe is in relation to the camera, and diffusing, concentrating and obstructing the strobe's light. The distances tested were 10 and 20 m where the strobe was turned away from the camera at 10 m also. The strobe was not turned away from the camera at 20 m as it was almost invisible and was below the minimum set thresholds.

The lights in the parking garage used motion detection to change the luminance which neatly tested the background updating function. The further away the strobe was the lower its intensity was, and when far enough away, the light from the ceiling lamps dominated over the strobe.

During the first round of field testing in the parking garage, several major issues were discovered. Calculating the row mean values caused light sources to sometimes trigger the detection algorithm and the first implementation of the thresholds was dependent on the difference between the background and the current image, with the effect that when the strobe left the frame, the threshold sank and caused false positives.

The tests were ended, and the issues were addressed by switching mean calculations to median and completely rebuilding the threshold algorithm as explained in 3.1.5. Further development

was also required to reliably store the background whenever large changes in the image background were detected as explained in 3.1.6. Furthermore, a comparison between median and mean was done to determine which method would work best in this scenario. This was done by performing both calculations at the same time in the code and storing the data for the same image. The results of this are shown in 4.2.1.

3.3.2 Horizon at dusk

A test which was conducted outside of Axis' premises was in a setting where the sky was in the image during dusk. The camera was aimed so that both the ground and sky was in view, and the strobe was placed in the vicinity pointing in the same direction. This resulted in a split image where the lower half was dark, and the upper half bright as seen in figure 3.8. Furthermore, the strobe was visible in half the image but faded in the upper half because of the sunlight at dusk being bright enough to hide it, as well as the lack of something for the strobe to reflect off.

One of the main goals of this thesis was to produce a detection algorithm robust enough to not trigger on bright static lines, such as the horizon. This was tested 15 minutes after sunset in a rural setting with the P1385 camera. As can be seen in figure 3.8, there is still sufficient light to create a distinct difference between the horizon and the ground below. While the detection algorithm reliably found the strobe and eliminated it in the bottom half of the image, the Maintain Delay method was unable to compensate properly due to the sky, as this test was performed before implementing a minimum threshold. Results are shown in 4.2.2.

Figure 3.8: Shows the background when testing against the horizon at dusk.

3.3.3 Nighttime test

Another test outside Axis' premises was a test at night in a cluttered environment with objects such as a barn and houses at different distances from the camera to see how the strobe band would appear in the image. The strobe was pointing in the same direction as the camera which diffused the light. The scene can be seen in figure 3.9. Results are explained in 4.2.3.

Figure 3.9: Shows the background when testing in a more cluttered environment at night.

3.4 Evaluation of sources

3.4.1 Thesis papers

[1] This thesis by C. L. Tormo was published by KTH, The Royal Institute of Technology in association with Axis Communications AB in 2018. As a master's thesis it has been peer reviewed and published, and downloaded 217 times. This thesis contains Axis specific work which is relevant to this thesis.

3.4.2 Scientific literature

[2], [3], [5] "CMOS Image Sensors" by K. D. Stefanov was first published 2022 by IOP Publishing which is a scientific research publisher branch of the Institute of Physics in the UK. The book references other articles and scientific literature in every chapter. It is cited in four different scientific papers relating to CMOS technology.

[4], [5] “Smart Camera Design” by M. Wolf was first published 2018 by Springer International Publishing AG which is a scientific research publisher and references other articles and scientific literature in every chapter. It is cited in seven scientific papers.

[15], [16] “Digital Image Processing” by R. C. Gonzalez and R. E. Woods was published by Pearson Education in 2018 and has over 90 000 citations on Google Scholar. This indicates the book’s substantial influence in the field of image processing.

[16] “Matematisk Statistik” by Kerstin Vännman and Adam Jonsson is part of the course literature for “Probability Theory and Discrete Mathematics” (FMSF40) at Lund University. It is published by Studentlitteratur. This book provides information on statistical methods and probability theory used in this thesis.

[17], [19] “Digital Signal Processing: A Practical Guide for Engineers and Scientists” by S. W. Smith was originally published by Elsevier Science publications in 1997 and is cited over 5800 times according to Google Scholar. The principles used from this book were functional in this work.

3.4.3 Manufacturer popular science article

[7] As a popular science article from a manufacturer with the intent to generate interest in their products, the “RED-101 – Global & Rolling shutters” article needs to be read with a certain amount of scepticism. However, the information presented in the article correlates with internal Axis documentation and aligns with descriptions provided by supervisors at Axis. This consistency across multiple sources lends credibility to the technical aspects of the article, allowing for confident use of this information in the context of this thesis.

3.4.4 Datasheets

[8], [9], [10] Axis datasheets with technical specifications about their cameras were used in this thesis. These datasheets contain detailed information about their products and provide essential information about the camera models used for testing.

[12] The datasheet regarding the Raytec Pulsestar VTS provided vital information concerning the operation of the strobe as well as a manual detailing correct use of the device. This information has been required to understand the capabilities and limitations of the strobe.

3.4.5 White papers

[11], [14] Two Axis whitepapers have been used in this thesis, as these papers can be considered marketing material, they should be viewed critically. However, since this thesis has been focused on technical information presented in these papers, and the specifications has been cross verified with supervisors at Axis to ensure reliability. This approach ensures that conclusions drawn are based on the technical foundations, despite the potential bias inherent in marketing materials.

4 Results

In this chapter, results from measurements and observations are presented. The program was tested in both laboratory and real-world environments. Results are presented from the various laboratory tests in the form of the α_n-test, mean frame rate tests, function generator tests, delay updates and failures, and final implementation behaviour. Results from the real-world testing in the form of issues discovered, performance in different situations are also presented.

The final implementation features a modified version of the original implementation. The algorithm still utilizes the principle of detecting the centre of a strobe band in the Find Strobe Bands method which to return to the Run method. Calculations are then performed to determine its position in frame compared to the previous frame which determines the number of rows the band has moved. An IIR-filter is then applied to this to counter variance and to allow the frame rate to converge. Eventually, the frame rate is synchronized to the strobe frequency. However, as mentioned in 3.2, the frame rate varies slightly, which caused different results every run. Therefore, the IIR-filter was tuned, and when the strobe moves less than 2 rows in between frames, the calculated frame rates are appended to a vector. Then the mean of the latter half of the collected frame rates is used as the final frame rate to increase predictability and reduce variance between runs. Finally, a delay to the exposure is applied to move the strobe out of the exposure window.

During the first 120 frames of Find Strobe Bands, a special exposure sequence runs. The exposure is initially in automatic mode but locked to a 3 ms exposure time to let gain and aperture adjust themselves. Then the mode is changed to hold in order to lock the three parameters to keep the strobe a fixed width and to prohibit in camera image alterations.

Find Strobe Bands was modified to now use the difference of the current frame to the background by storing a background when the strobe is out of the frame as it scrolls by when unsynchronized. At the same time, a threshold is calculated for another 120 frames of which to detect the strobe by also utilizing the scrolling behaviour. An MA-filter was also introduced to reduce noise as it affected the algorithm at low strobe brightness and to remove spikes in the data.

Due to the earlier described discretization issue where the frame rate never exactly matched the strobe frequency, a new method called Maintain Delay was introduced. This method only checks the top and bottom row of the image by comparing each to a threshold which is calculated alongside the threshold for the Find Strobe Bands method. If the algorithm detects the strobe, a new delay to the exposure window is calculated. Furthermore, in this method, the background and thresholds can update if a change is detected in both the top and bottom of the image at the same time. Here, 1/36 of the rows are checked in both the top and bottom to increase the chances of updating.

4.1 Results from laboratory tests

The algorithm never calculated the same frame rate every run of the program which resulted in variance in its performance in the form of how fast the strobe would move back into frame. This affected how well synchronized the frame rate was and how long it took for the strobe to reappear.

As explained in 2.7.3, the discretization of N_{max} affected how far off the real strobe frequency the frame rate was set. N_{max} could in one run equal 3004 at a calculated frame rate of 24.959 but also equal 3004 at a calculated frame rate of 24.966 in another run. Observe table 4.1 where a frame rate of 24.959 and 24.966 equal the same N_{max}. Both frame rates equal 3004 when cutting the decimals but 24.966 is much closer which results in smaller errors when performing the iterative calculations.

Table 4.1: Shows the frame rates from one of the first tests with their corresponding N_{max}.

Calculated frame rate (fps)	N_{max} (rows)	Calculated N_{max} (rows)
24.959	3004	3004.936
24.967	3003	3003.973
24.961	3004	3004.695
24.961	3004	3004.695
24.966	3004	3004.093
24.968	3003	3003.852
24.957	3005	3005.176
24.972	3003	3003.371
24.963	3004	3004.454
24.961	3004	3004.695

4.1.1 Frame rate accuracy test results with the strobe's internal trigger

The α_n tests were first performed with a sample size of 10 at a larger interval of α_n values ranging from 0.01 to 0.5. Testing was then narrowed to an interval of 0.015 to 0.04 with a sample size of 50. Table 4.2 demonstrates the test of the larger interval, which was only performed on the P1385. The standard deviation of the mean frame rate reveals how closely the calculated frame rates matched between runs. The strobe frequency was 25 Hz for all tests, frequencies up to 30 Hz were quickly tested and worked but were not recorded.

Table 4.2: Results from the first round of α_n tests with the P1385 with a sample size of 10, meaning 10 runs at each value of α_n.

α_n	Mean frame rate (fps)	Frame rate standard deviation	Mean frames until stable	Mean frames until update	Median frames until update	Eval. score
0.01	24.9635	0.004625	1206	2091	1784	375
0.015	24.9647	0.001636	1181	7477	7041	3870
0.02	24.9643	0.002214	809	19709	3976	11008
0.025	24.9643	0.001703	889	9333	8960	6168
0.03	24.9645	0.001080	723	18310	13022	23449
0.035	24.9633	0.002312	709	12900	5447	7871
0.04	24.9622	0.010475	643	7562	4083	1123
0.05	24.9618	0.006088	653	11959	5665	3010
0.075	24.9640	0.003742	616	4690	3609	2036
0.1	24.9645	0.002759	604	4264	3207	2560
0.15	24.9624	0.006433	586	1314	1157	348
0.2	24.9676	0.004881	551	3001	1821	1116
0.25	24.9663	0.004616	581	4032	2221	1505
0.3	24.9623	0.004040	505	9883	3237	4842
0.35	24.9644	0.006067	615	4967	1255	1332
0.4	24.9666	0.005275	649	3516	3794	1027
0.45	24.9618	0.008343	617	18275	1145	3552
0.5	24.8616	0.314083	608	1539	1136	8

As can be seen in table 4.2, the value 0.03 scored the highest, therefore, it and the surrounding values were tested further. Tables 4.3 and 4.4 are the same as table 4.2 but with the smaller interval of α_n values. Here, the P1387 was introduced in testing as well although the P1385 was prioritized over the P1387. Table 4.3 shows that 0.03 scored the highest for the P1385 but 0.02 scored the highest for the P1387. Even though 0.02 seemed better, it was slow and inconsistent in how many frames it took to synchronize and eliminate the strobe which is why the figures following figure 4.1 use 0.03 for the P1387 also. The other graphs for the other α_n values in tables 4.3 and 4.4 is found under 7.2 in Appendix.

Table 4.3: Results from the second round of α_n tests on the P1385 with an increased sample size of 50.

α_n	Mean frame rate	Frame rate standard deviation	Mean frames until stable	Mean frames until update	Median frames until update	Eval. score
0.015	24.96421	0.00215	16467	6530	1250	6141
0.02	24.96439	0.00204	971	14045	5546	7090
0.025	24.96473	0.00231	784	13173	4193	7287
0.03	24.96430	0.00241	795	17800	4791	9274
0.035	24.96416	0.00268	793	8593	4021	4048
0.04	24.96498	0.00268	11266	4094	708	5951

Table 4.4: Results from the second round of α_n tests on the P1387 with an increased sample size of 50.

α_n	Mean frame rate	Frame rate standard deviation	Mean frames until stable	Mean frames until update	Median frames until update	Eval. score
0.015	24.96444	0.00442	1416	2243	2110	359
0.02	24.96438	0.00196	1001	2621	2118	1334
0.025	24.96544	0.00218	1833	2262	2063	565
0.03	24.96601	0.00812	1172	1619	1250	170
0.035	24.96601	0.00935	880	2131	2049	259
0.04	24.96379	0.00663	971	2306	1960	358

As previously stated about table 4.2 above, the larger sample size drastically reduced some evaluation scores confirming that a sample size of 10 is insufficient to draw any conclusions from. It could be argued that an even larger sample size is preferred. Time is however a limiting factor and even though the tests were automated, testing on the P1385 took a total of 48 hours of continuous runs.

While testing different α_n values, it was noted that a relatively small range of calculated frame rates resulted in the most stable synchronization. As can be seen in figure 4.1, a tightly grouped number of frame rates close to the mean frame rate led to the artefact being outside the exposure window for more than 10 minutes without any additional delays required. Surprisingly, some frame rates were very well synchronized and resulted in almost two hours until an update.

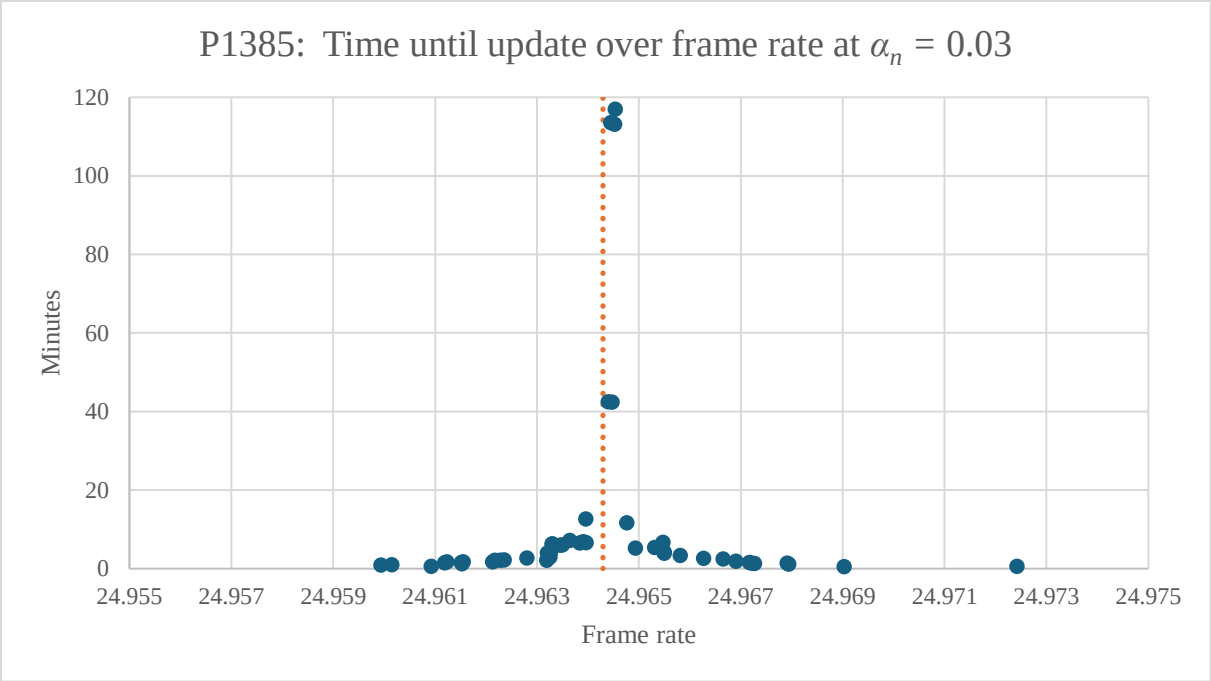

Figure 4.1: Shows the amount of time in minutes until update for each calculated frame rate at $\alpha_n = 0.03$ for the P1385. The dotted line represents the mean calculated frame rate.

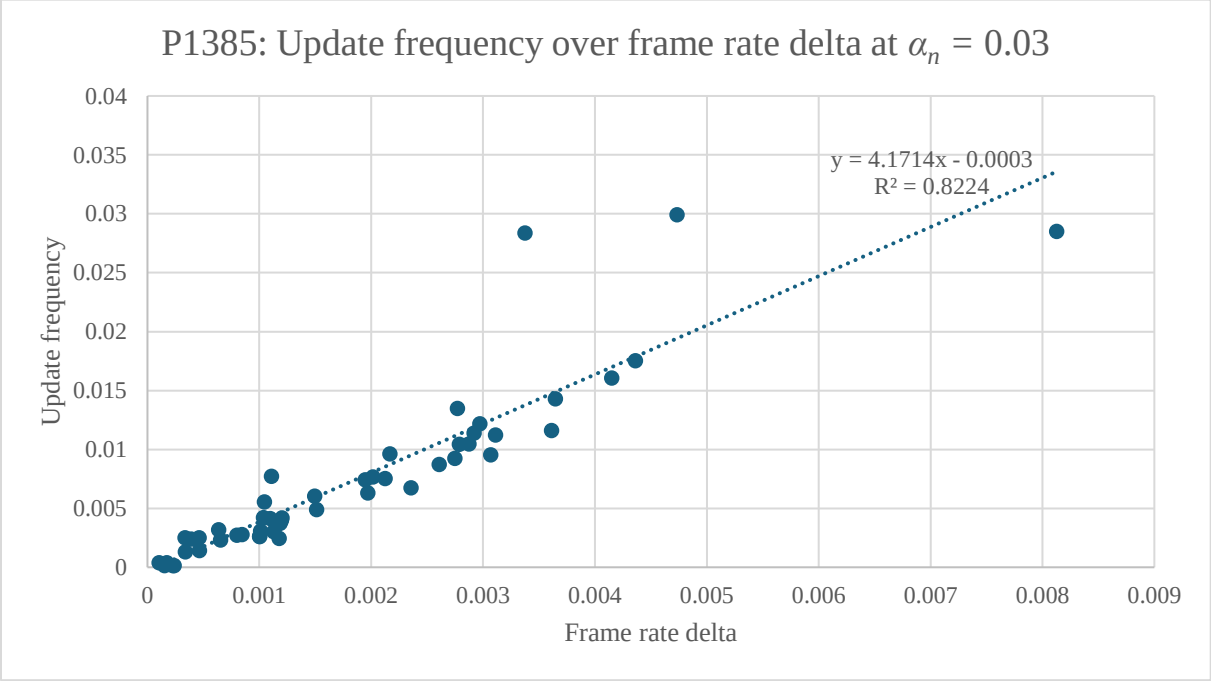

Figure 4.2: Shows the linear relationship between the frequency of updates and the frame rate delta to the mean from the data in figure 4.1.

Figure 4.2 contains the same data as in figure 4.1 except that the y-axis (Update frequency) is the frequency of updates as opposed to time between updates and the x-axis (Frame rate delta) is the difference from the mean. Values close to the origin are close to the mean and the update frequency is low which is the best outcome. The ideal trendline intersects the origin and the ideal $R^2 = 1$. The three values furthest from the trendline are caused by the delay command failing to apply the correct delay. These were excluded from the calculations as they obscured the results and dealing with this behaviour was out of the scope of this work. They are however included in the figure to show that they do occur. The R^2 without the three outliers is 0.9346 and the line equation is $y = 3.76x - 9 \cdot 10^{-5}$.

Unexpectedly, the P1387 preferred to calculate frame rates that did not yield the longest time until update. As can be seen in figure 4.3, most of the frame rates are clumped at frame rates that did not result in the longest times until an update. Moreover, this camera failed more often to apply the delay as shown by the dots around the dotted mean line in figure 4.3. The linearity is also poor compared to the P1385 as shown in figure 4.4.

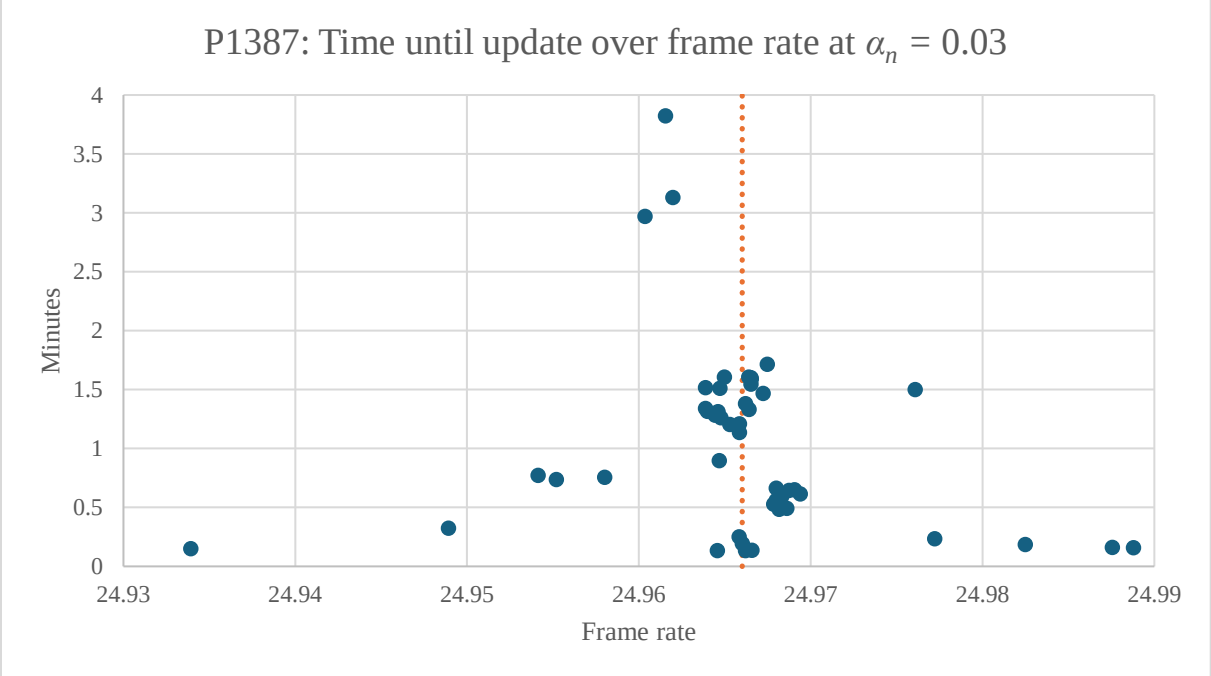

Figure 4.3: Shows the amount of time in minutes until an update for each calculated frame rate at $\alpha_n = 0.03$ for the P1387. The dotted line represents the mean calculated frame rate. Note the wider spread and the smaller magnitude of time compared to figure 4.1.

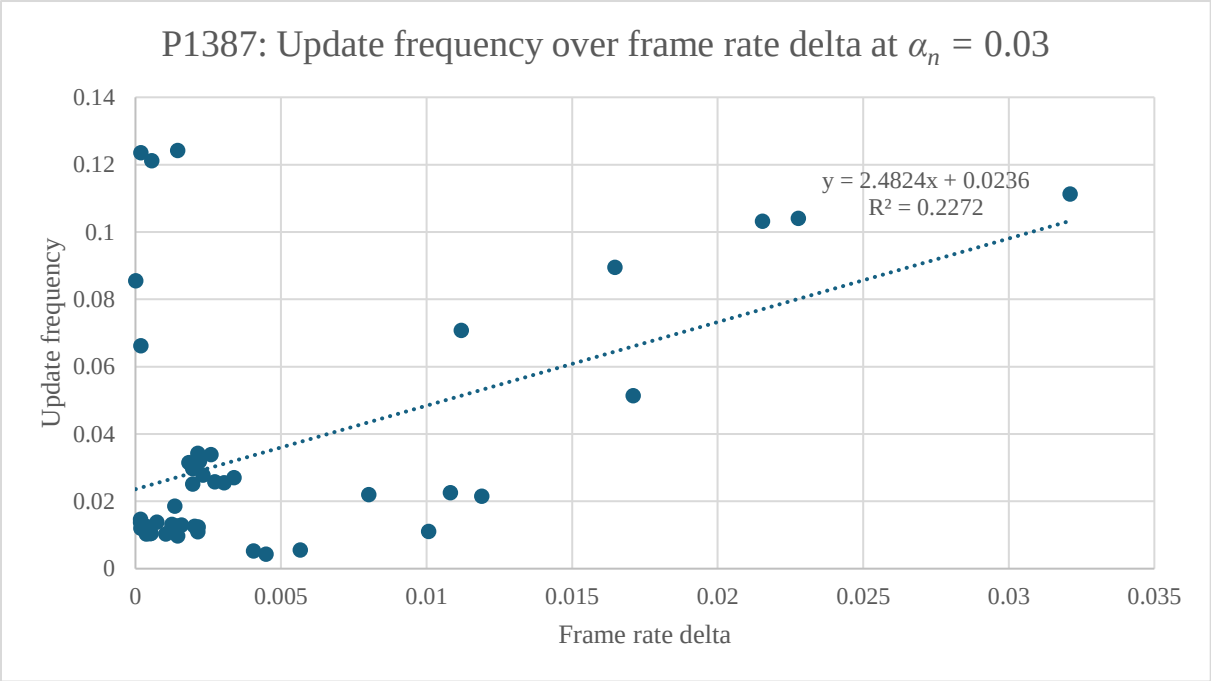

Figure 4.4: Shows the linear relationship between the frequency of updates and the frame rate delta to the mean from the data in figure 4.3.

As seen in figure 4.5, the initial prediction that a low α_n value would increase the number of frames needed to stabilize the artefact was correct and is verified by the data from figure 4.1. The magnitude of the values is uninteresting in this case as there are frame counters which are used to let exposure and thresholds settle which is why the lowest value at 0.3 is 0. There is also the possibility that the strobe drifts out of the image as the frame rate is calculated, which artificially extends the time as no calculations can be performed.

When testing α_n at and below 0.02 the algorithm failed more often to apply the delay. It was sometimes very slow as it calculated an accurate frame rate, but the strobe sometimes moved out of the frame before it was finished which resulted in the strobe having to slowly move back to set the final frame rate.

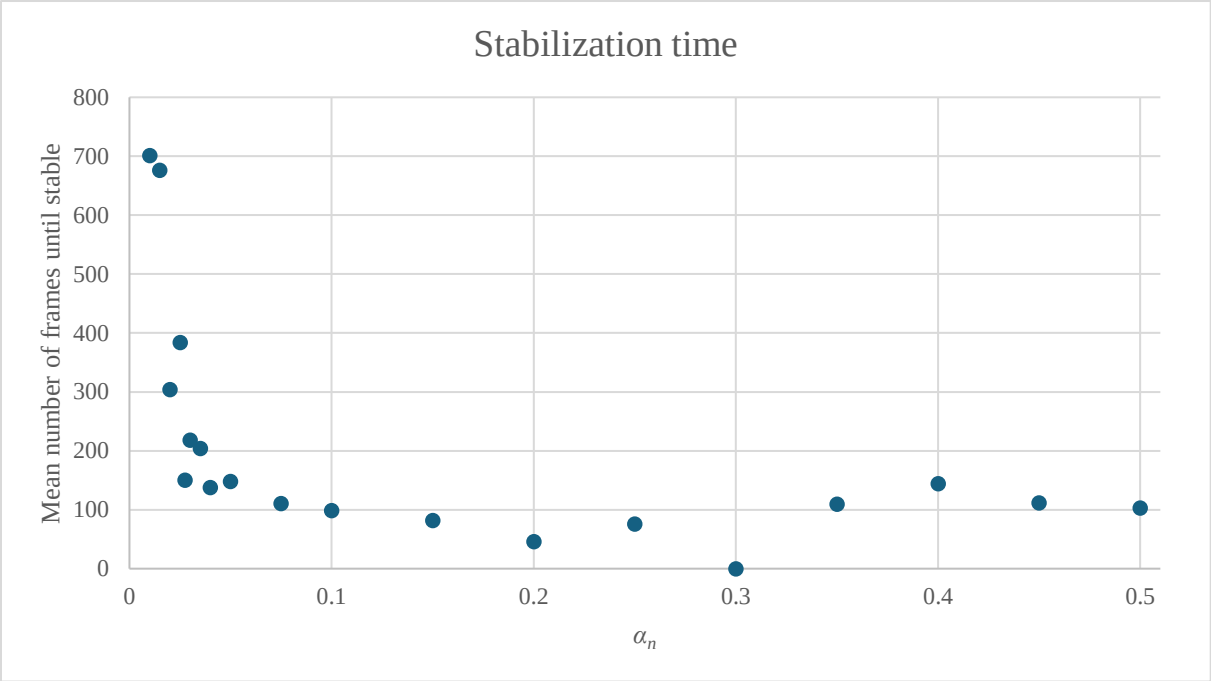

Figure 4.5: Shows the mean number of frames from table 4.1 at each α_n required to calculate a stable frame rate on the P1385. More tests were performed in the range of 0 to 0.1 as those values had better evaluation scores.

4.1.2 Utilizing mean frame rate with chosen α_n value

Instead of using the last calculated value as the frame rate before applying the delay, the mean of half of the last 200 frame rates was used to set the final frame rate. Figure 4.6 is an example of this. Table 4.5 is the result of the calculated frame rate when using the mean. As can be observed, the spread is narrower, and the standard deviation is smaller compared to using only the last calculated frame rate. This test collected 200 out of the calculated mean frame rates and ran over the weekend as it took 40 hours to complete due to the increased accuracy of the frame rate calculations.

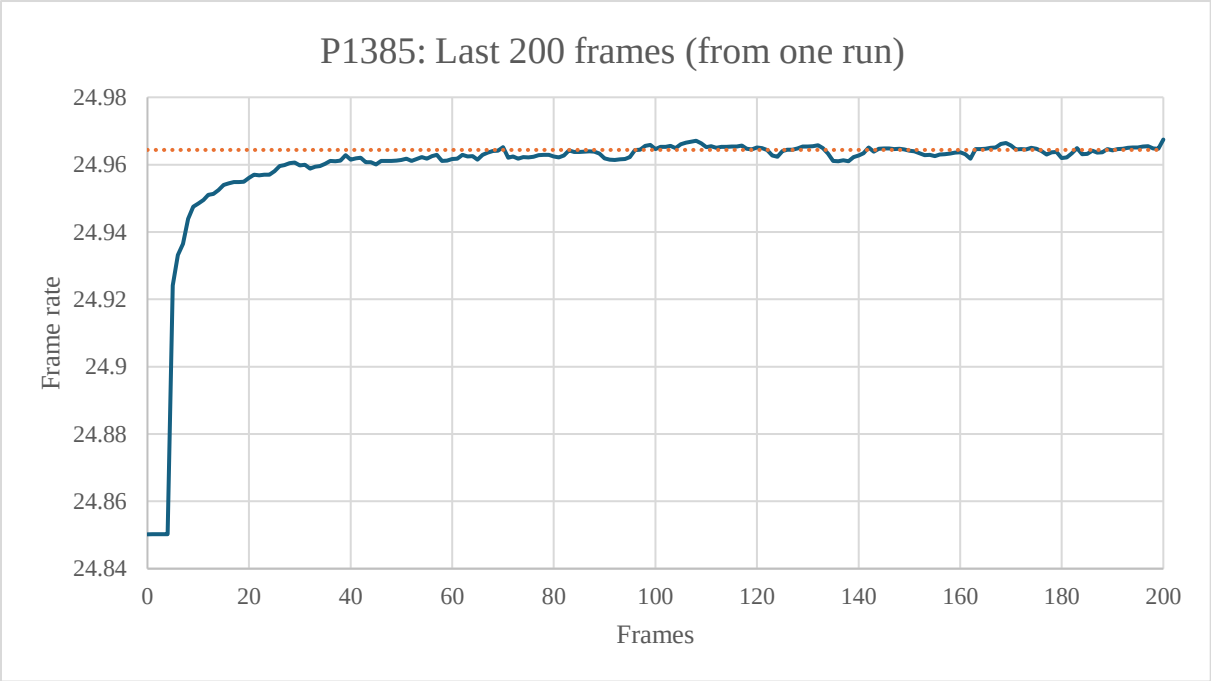

Figure 4.6: Shows the last 200 calculated frame rates before the camera is synchronized. The dotted line represents the mean of the last half (100) of the frame rates.

As presented in table 4.5, using the mean frame rate of the last half of the frame rates to set the final frame rate before applying the delay increases the evaluation score by 34 %. This is mainly due to the frame rate standard deviation decreasing. Furthermore, while the mean frames until update decreased by 6 %, the median frames until update increased by 52 %, implying that there are fewer outliers.

Table 4.5: Shows the results of tests using the mean frame rate compared to using the last calculated frame rate with $\alpha_n = 0.03$ on the P1385.

Half of last frame rates?	Mean frame rate	Frame rate standard deviation	Mean frames until stable	Mean frames until update	Median frames until update	Eval. score
Yes	24.96435	0.00136	994	16774	7257	12435
No	24.96430	0.00241	795	17800	4791	9274

In figure 4.7, by using the mean of the last half of the calculated frame rates, the spread of the frame rates is smaller compared to figure 4.1. Here, 200 values were collected instead of 50 to acquire a clearer outline. Figure 4.8 shows that more of the frame rates are grouped near the origin which is the desired outcome. There were outliers as in figure 4.2 but they are removed from this one to increase clarity.

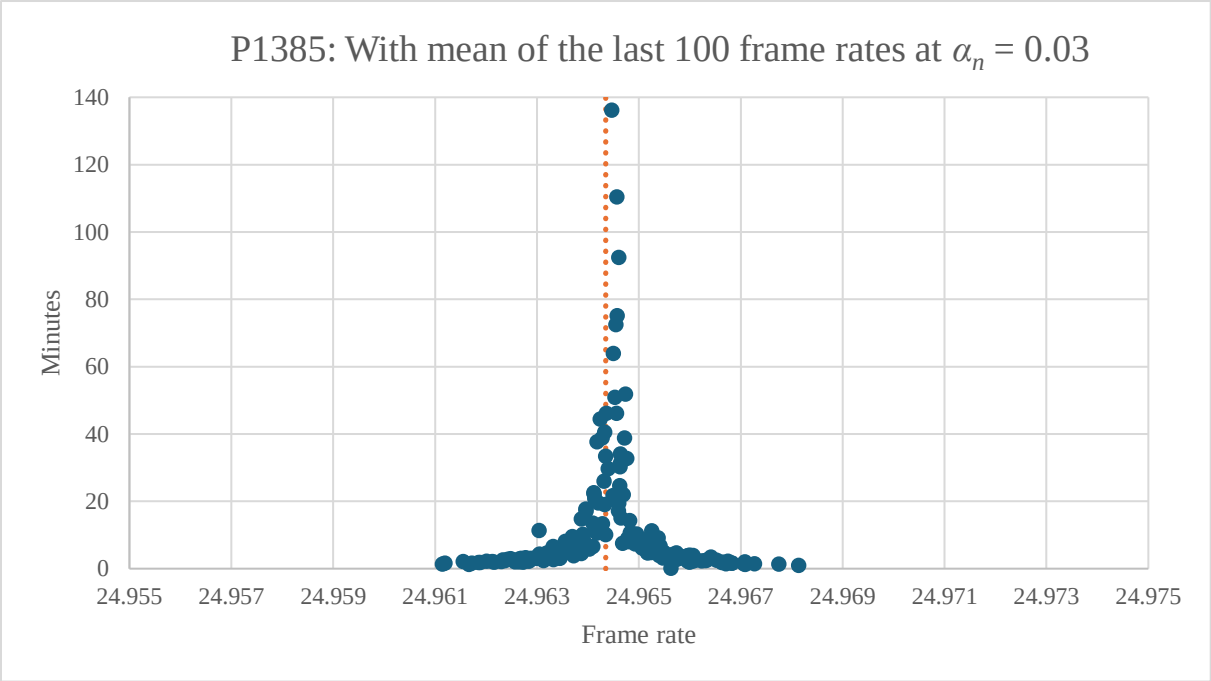

Figure 4.7: Shows the amount of time in minutes until artefact reappearance for each calculated mean frame rate at $\alpha_n = 0.03$ and when using the mean of the last half of the frame rates for the P1385. The dotted line represents the mean frame rate.

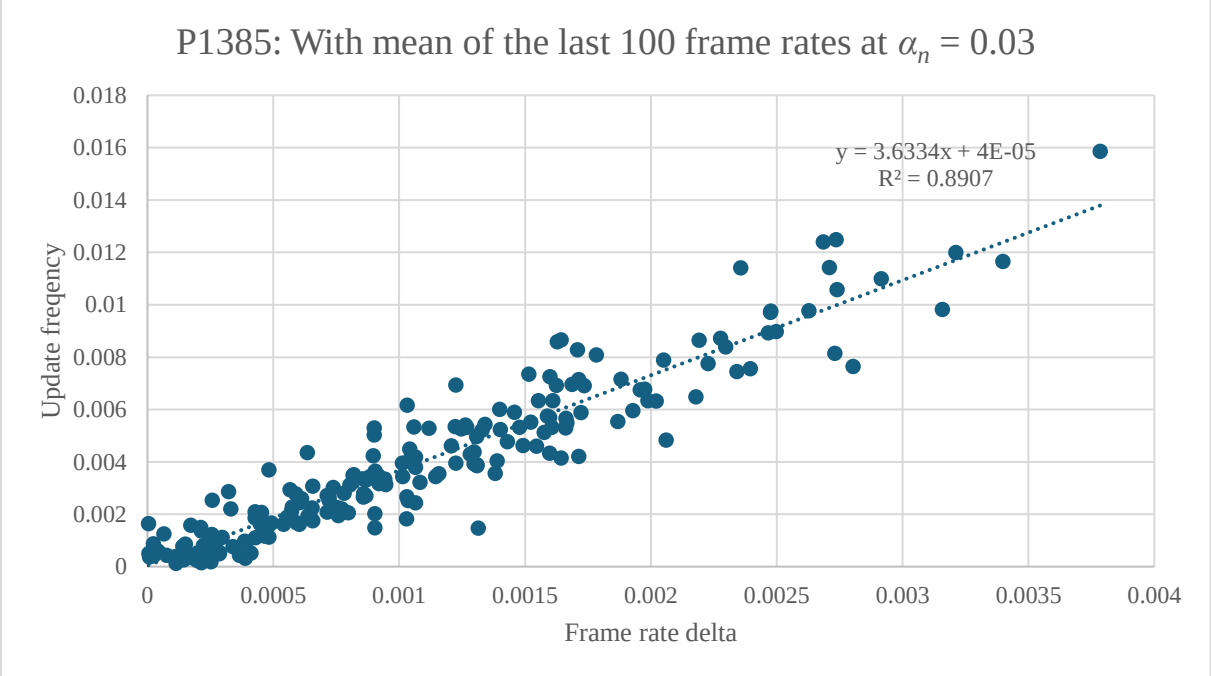

Figure 4.8: Shows the linear relationship between the frequency of updates and the frame rate delta to the mean, when using the mean of the last 100 frame rates on the P1385.

Figures 4.9 and 4.10 show that even though there were other inconsistencies in the P1387 which resulted in many more delay command failures, using the mean of the last half of the last 200 frame rates resulted in a smaller standard deviation compared to figures 4.3 and 4.4 in 4.1.1. The data is unfortunately difficult to interpret due to the delay problem which is why only the figures are shown for this camera.

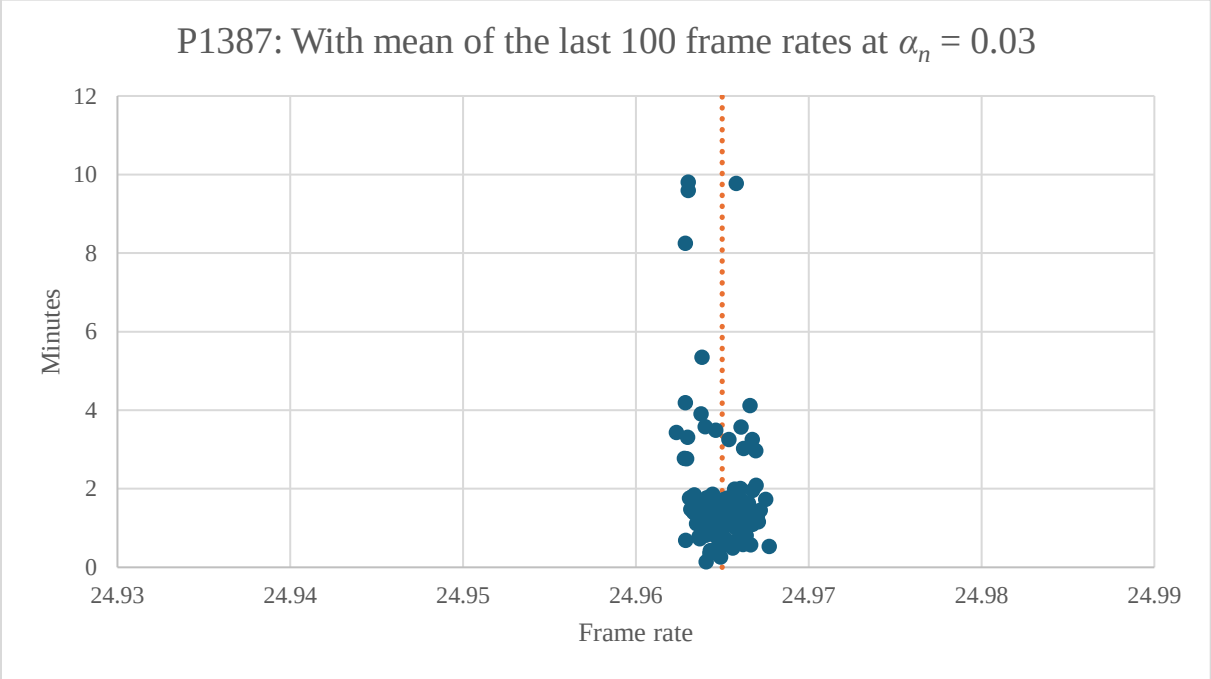

Figure 4.9: Shows the amount of time in minutes until artefact reappearance for each calculated frame rate at $\alpha_n = 0.03$ for the P1387. The dotted line represents the mean calculated frame rate.

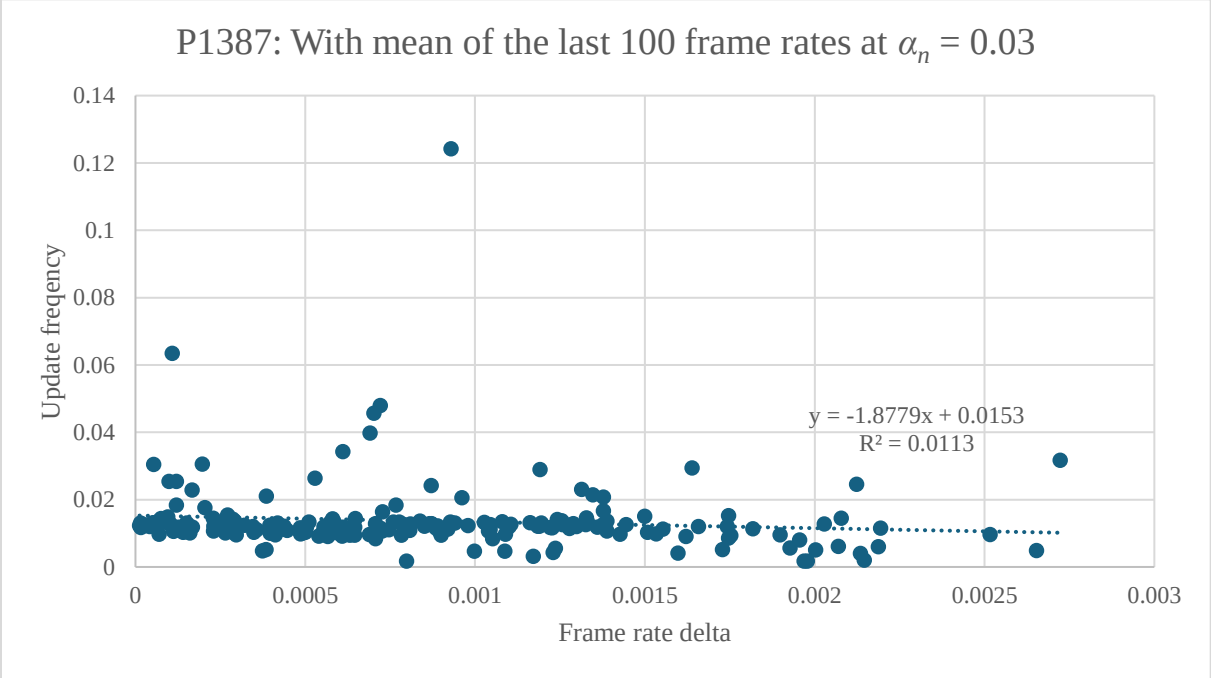

Figure 4.10: Shows the linear relationship between the frequency of updates and the frame rate delta to the mean, when using the mean of the last 100 frame rates on a P1387.

An issue discovered when using the mean of the last half of 200 frame rates was a bouncing behaviour when the strobe was stabilized in the bottom edge of the image as seen in figure 4.11. This did not occur in the top of the image. As explained in 3.1.2, the MA-filter performs a mean calculation of the last elements (the bottom edge) which could be the cause. Another cause of the bouncing behaviour was when there were large intensity changes in the strobe itself as shown in figure 4.12. Where the band start was set at 465 and band end was set at 511 if the threshold was too large, which meant that the detected band centre was not the actual centre as when the band end was set, then Find Strobe Bands returned that value. This bouncing could occur in any part of the image. This is further discussed in 5.1.2.

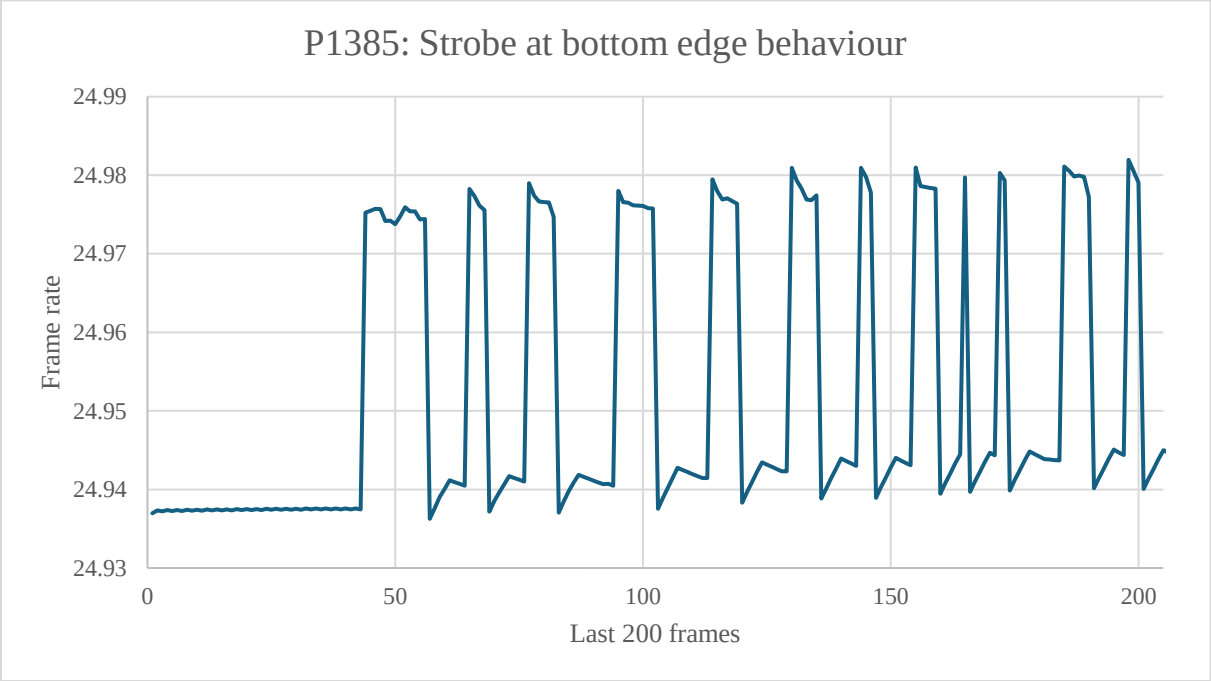

Figure 4.11: Shows a problem when the strobe is at the bottom edge of the image where the frame rate jumps between two levels.

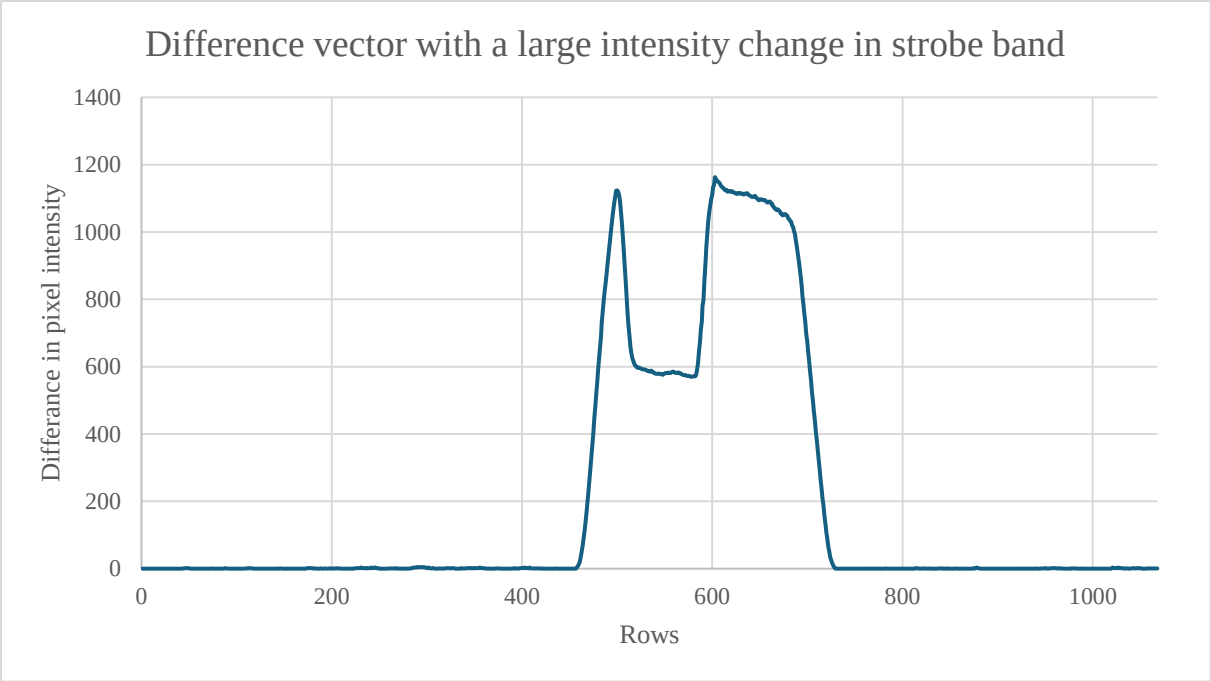

Figure 4.12: Shows the difference to the background where there is a large intensity drop in the middle of the strobe.

The stability threshold and the minimum number of stable frames were also tested where 100 values were collected for each test, the results are shown in tables 4.6 and 4.7. When testing the stability threshold, the difference was small and down to variance and a real difference was only noticed when increasing the threshold at larger steps of 5. The stability threshold was set at 2 as this only allowed the strobe to move one row in between frames, this was justified as if it worked with the jittering strobe, it would work with more stable strobes. The minimum stable frames showed more promise and the value of 200 showed the smallest standard deviation and was kept. These two tests were deemed of low priority and more testing is required for better accuracy.

Table 4.6: Shows how the stability threshold affects the frame rate standard deviation.

Stability threshold	Mean frame rate	Frame rate standard deviation
2	24.96432	0.001482
3	24.96420	0.001689
4	24.96429	0.001510

Table 4.7: Shows how the number of stable frames affect the frame rate standard deviation.

Minimum stable frames	Mean frame rate	Frame rate standard deviation
50	24.96857	0.003574
100	24.96483	0.001954
200	24.96457	0.001285
300	24.96322	0.002705

4.1.3 Tests with function generator

As described in 3.2, the strobe's internal trigger was unstable as shown in figure 4.6 which was also proved during this test, As can be seen in figures 4.13 and 4.14, using a stable trigger result in a more precise calculation of the camera's frame rate.

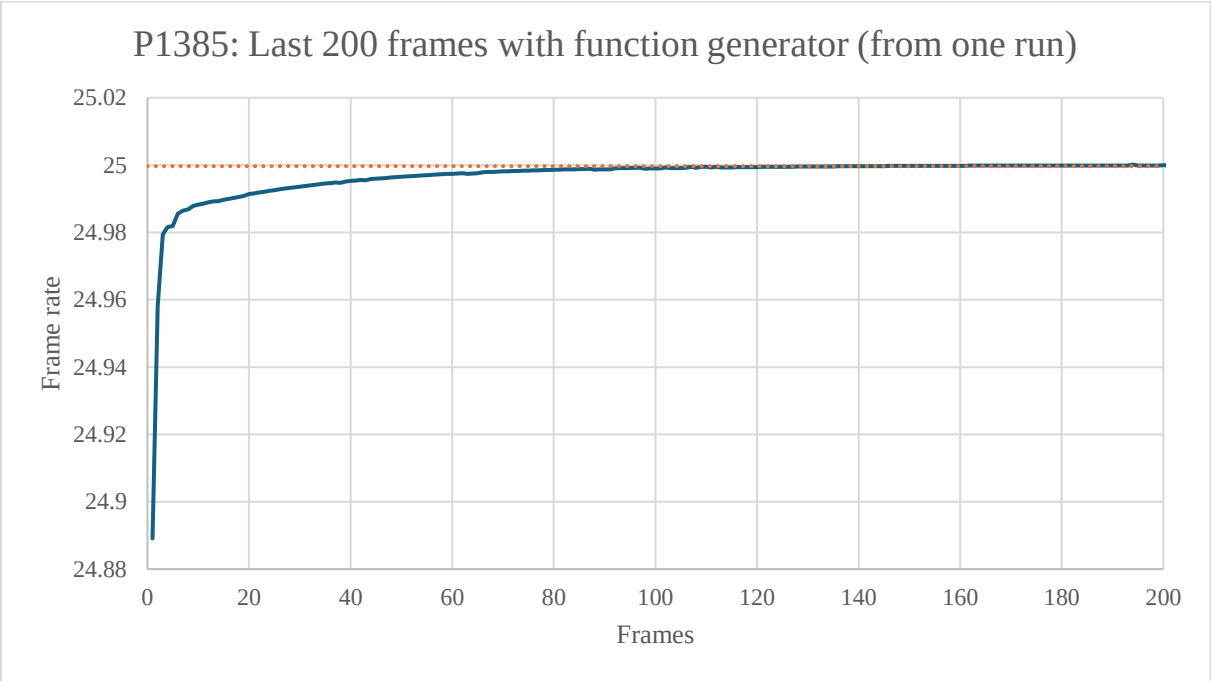

Figure 4.13: Shows the last 200 calculated frame rates when synchronized with a function generator trigger. The dotted line represents the mean of the last half (100) of the frame rates.

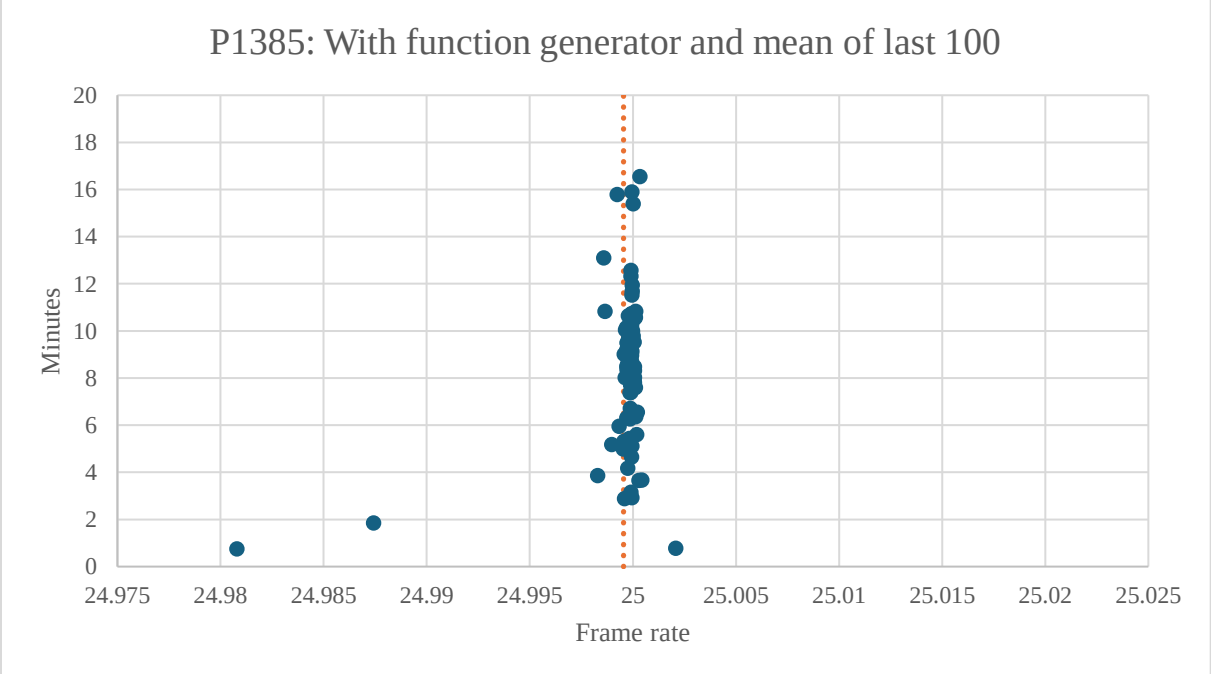

Figure 4.14: Shows the time until artefact reappearance for each frame rate at $\alpha_n = 0.03$ for the P1385, with a function generator trigger. The dotted line is the mean calculated frame rate.

Figure 4.15 shows the linear relationship between how precise the frame rate calculations are and the update frequency. This graph might be misleading as most frame rates are close to the mean value, but the time until update still varies. This will be further discussed in 5.2.

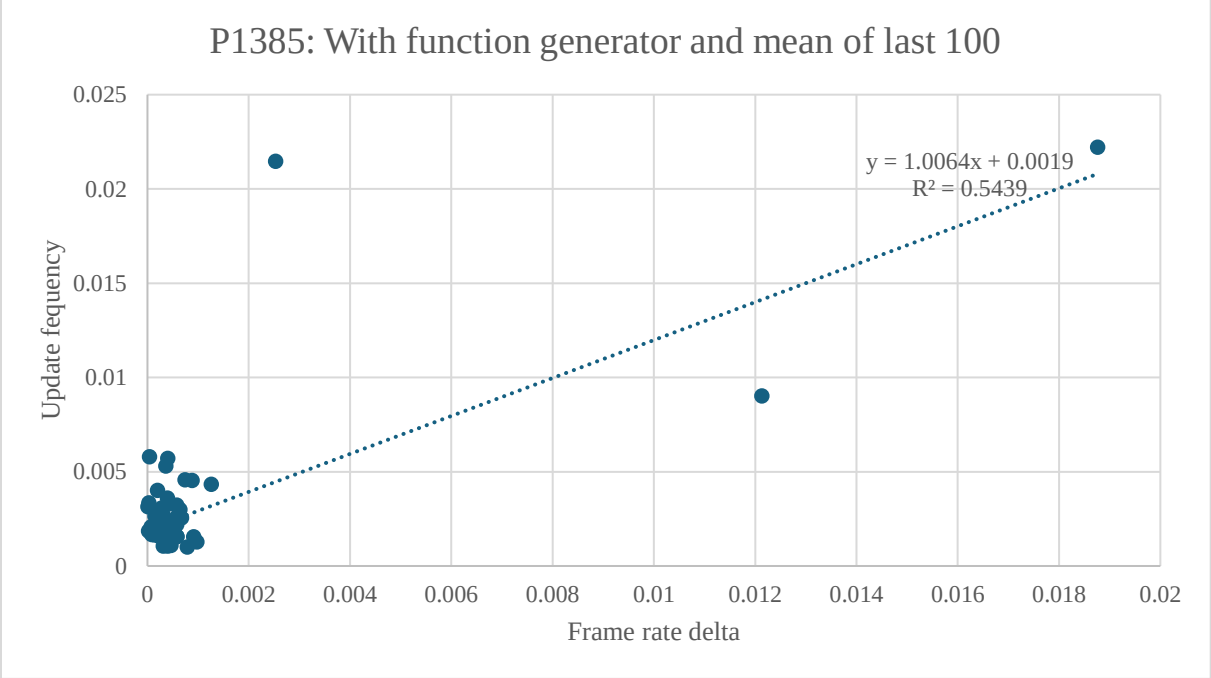

Figure 4.15: Shows the linear relationship between the frequency of updates and the frame rate delta to the mean, while the strobe trigger is controlled by a function generator.

Table 4.8: Shows the results of 100 tests with a function generator controlling the strobe on a P1385 with $\alpha_n = 0.03$. As well as the corresponding test using the strobes internal trigger.

With outliers?	Mean frame rate	Frame rate standard deviation	Mean frames until stable	Mean frames until update	Median frames until update	Eval. score
Yes	24.99955	0.00229	614	12696	12828	9037
No	24.99984	0.00032	614	13036	12985	67377
$\alpha_n = 0.03$ from table 4.5	24.96435	0.00136	994	16774	7257	12435

As previously showed in figure 4.13, using a function generator to control the strobe reduces the variance in the interval between two pulses. This in turn leads to more precise frame rate calculations as can be seen table 4.8. If the outliers are ignored, the result is a synchronization phase with lower standard deviation compared to when the strobe's internal trigger is used. As the evaluation score used in this thesis favours a low standard deviation, the evaluation score of 67377 is misleading. The evaluation score for the row from table 4.5 is also skewed by the large outliers.

4.1.4 Delay updates and failures in Maintain Delay

After letting the program run overnight six times it managed an average failure rate of 0.945 % as shown in table 4.9. The failure rate is calculated by taking the number of times the delay command failed to function as it was supposed to, divided by the number of delay updates. This behaviour appeared random, and no correlations were found, only that the delay command sometimes failed. These tests were performed at which frame rate the algorithm calculated for that specific run, which was unfortunately not recorded. But the data tends towards no correlation between the frame rate and failures, although more is required to say for certain.

Table 4.9: Shows the results when testing the delay application command on the P1385.

Test	Runtime (h)	Updates	Failures	Failure rate
1	19	588	5	0.85 %
2	20	620	6	1.00 %
3	18	480	4	0.80 %
4	16	437	2	0.48 %
5	17	1608	24	1.49 %
6	19	2847	30	1.05 %

4.1.5 Differences between camera models

For unknown reasons, the P1387 tended to calculate a frame rate which did not yield the highest frames between updates as can be seen in figure 4.3 where most of the frame rates are clumped together around the mean frame rate. But the frame rates with the highest frames between updates are achieved at a frame rate slightly below the mean frame rate.

Figures 4.15 and 4.16 show the time between updates for the P1385 and P1387 respectively at a fixed frame rate of 24.967 fps. The dips in the figures are when the delay command failures, resulting in an almost immediate update. The larger outlier in figure 4.16 is also a failure but in the opposite direction where the strobe is delayed further into the blanking window than specified. Table 4.9 shows the difference between camera models at the same frame rate where the P1387 has a 12.7 % shorter mean time until the update with a higher standard deviation. The failure rate is also shown where the P1385 failed to apply the delay once and the P1387 failed nine times during the short test. The highest recorded frames between updates was 3282 and the lowest, excluding the outlier, was 1833 (excluding outliers) which was used to decide when to allow the background to update in Maintain Delay as described in 3.1.6.

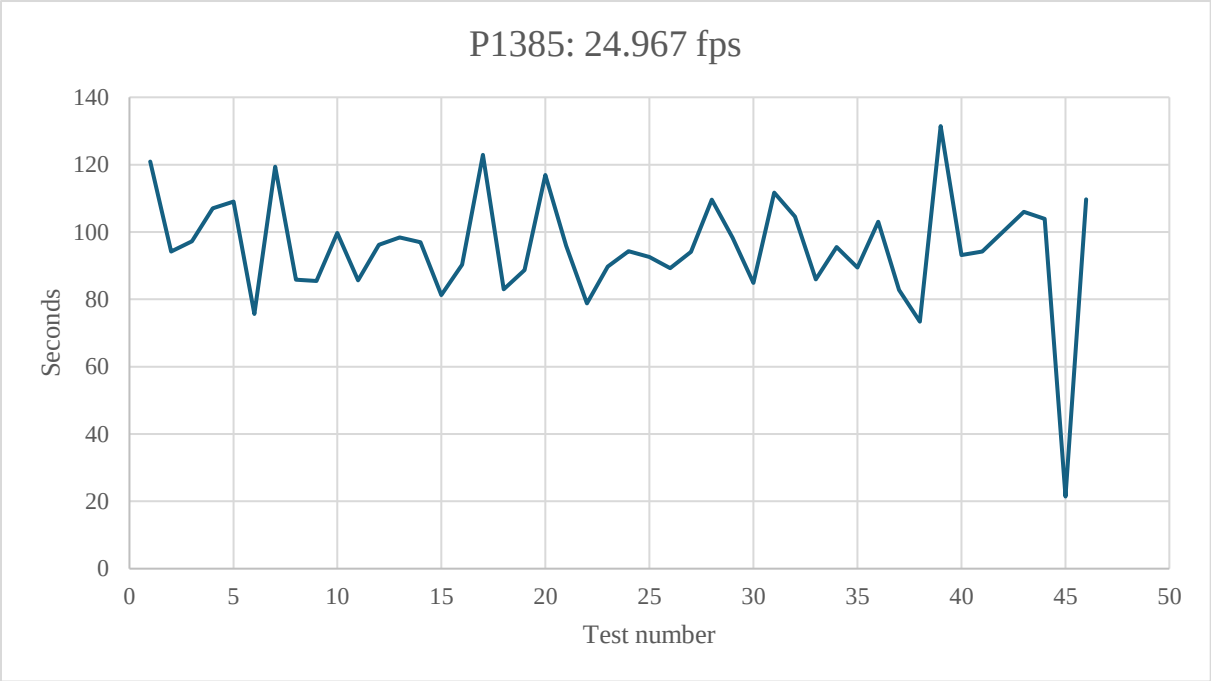

Figure 4.16: Shows the time until update on the P1385 with the frame rate set to 24.967 fps.

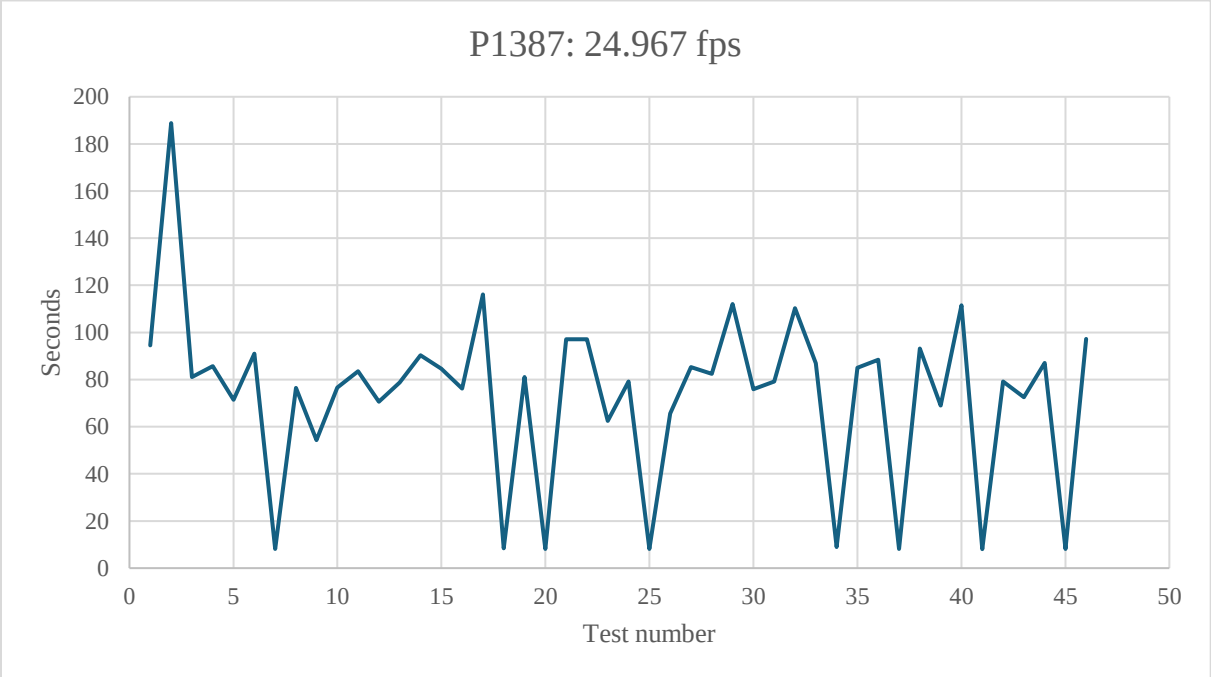

Figure 4.17: Shows the time until update on the P1387 with the frame rate set to 24.967 fps.

Table 4.10: Shows the difference between the P1385 and the P1387 when coding a fixed frame rate of 24.967.

Camera	Mean time to update (s)	Highest frames to update	Lowest frames to update	Time to update standard dev. (s)	Failure rate
P1385	97	3282	1833	12.8	2 %
P1387	84.6	2898	1356	13.6	20 %

4.2 Results from field tests

4.2.1 Parking garage

In figure 4.18, changes in the row intensity can be observed depending on the placement of the strobe. When the strobe was placed 10 m from and pointing at the camera as shown in figure 4.19, all algorithms worked as intended. However, when the strobe was placed 10 m and angled away or placed 20 m from the camera as seen in figures 4.20 and 4.21 respectively, issues in Maintain Delay arose. The initial detection still worked, and the strobe was synchronized but slower. But once the artefact reappeared, the algorithms were unable to detect it due to the low intensity at the top and bottom compared to where in the image it was stabilized. The intensity at 20 m and 10 m angled away are similar and has decreased by a factor of around 6 compared to the test at 10 m. The strobe is in the figures but barely visible. This will be further discussed in 5.3.

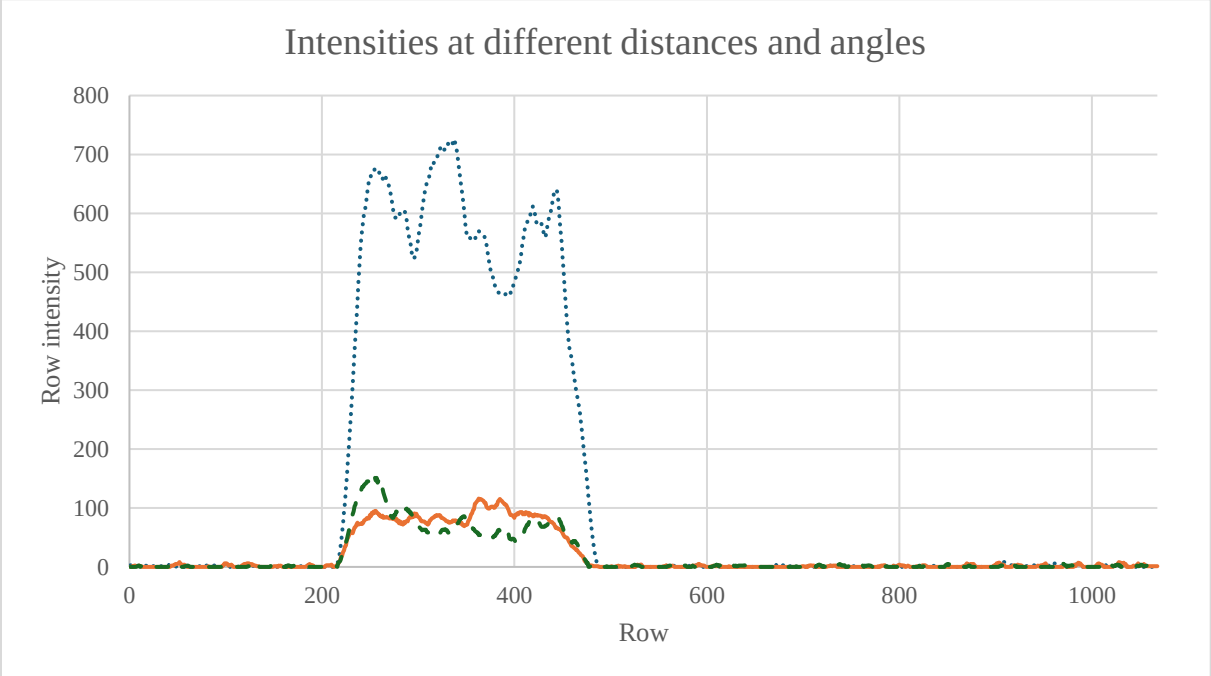

Figure 4.18: Shows how the distance from the camera and angle of the strobe affects the intensity. The blue dotted line is the strobe light 10 m away, the green dashed line is the strobe at 10 m angled away from the camera and the orange full line is the strobe 20 m away.

Figure 4.19: Shows the strobe 10 m away, pointed at the camera. Note the concentration of the light.

Figure 4.20: Shows the barely visible strobe 10 m and angled away from the camera, reflecting off the cars in the background.

Figure 4.21: Shows the barely visible strobe 20 m away, pointed at the camera.

The median and mean of the row values were tested as explained earlier in 3.3.1. Figures 4.22 to 4.27 show the camera images and median and mean data when capturing several light sources in the image. As can be seen in figure 4.23 and 4.25, the row mean value is sensitive to bright spots in the image, whereas the row median acts as a filter.

Figure 4.22: Shows the parking garage with several bright spots. The data matching the image is shown in figure 4.23.

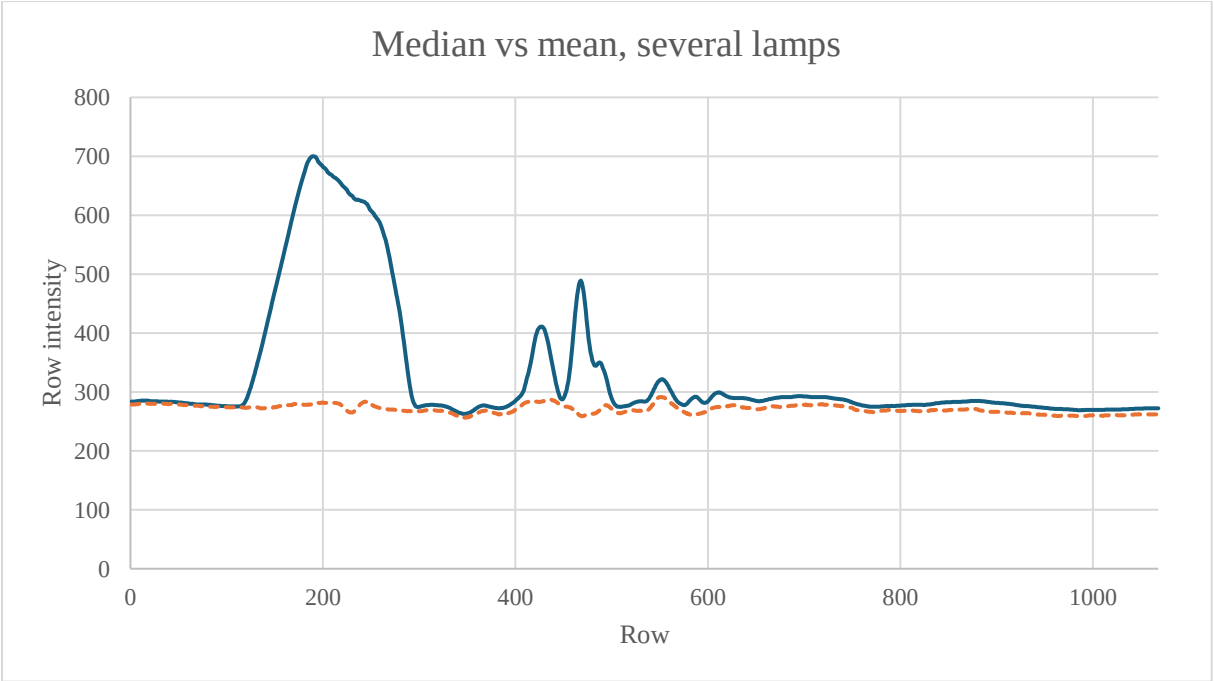

Figure 4.23: Shows the difference between calculating the row mean and row median when light sources are present in parts of the row. The dotted line is the median and the solid line is the mean.

As can be observed in figures 4.24 and 4.25, even when the lamp is covering a large part of several rows, this barely affects the median but drastically affects the mean.

Figure 4.24: Shows the parking garage with a lamp covering a larger part of the image. The data matching the image is shown in figure 4.25.

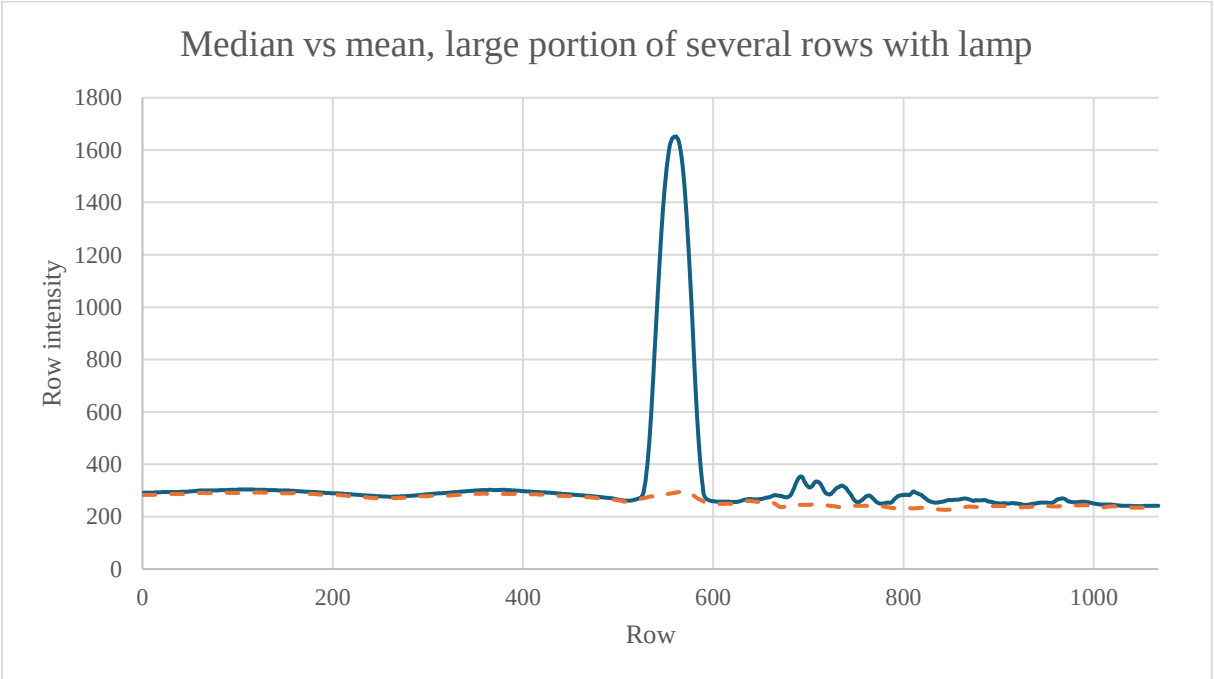

Figure 4.25: Shows how the row mean is heavily affected by a bright light source. The dotted line is the median and the solid line is the mean.

As seen in figure 4.26 and 4.27, once the light source covers several entire rows, the mean and median row values are similar. As the strobe artefact always appears across the image, this is preferable as this is the expected behaviour. This is deemed an acceptable limitation as if this were to happen, the background would contain the lamp, and the strobe could still be found in the other parts of the image.

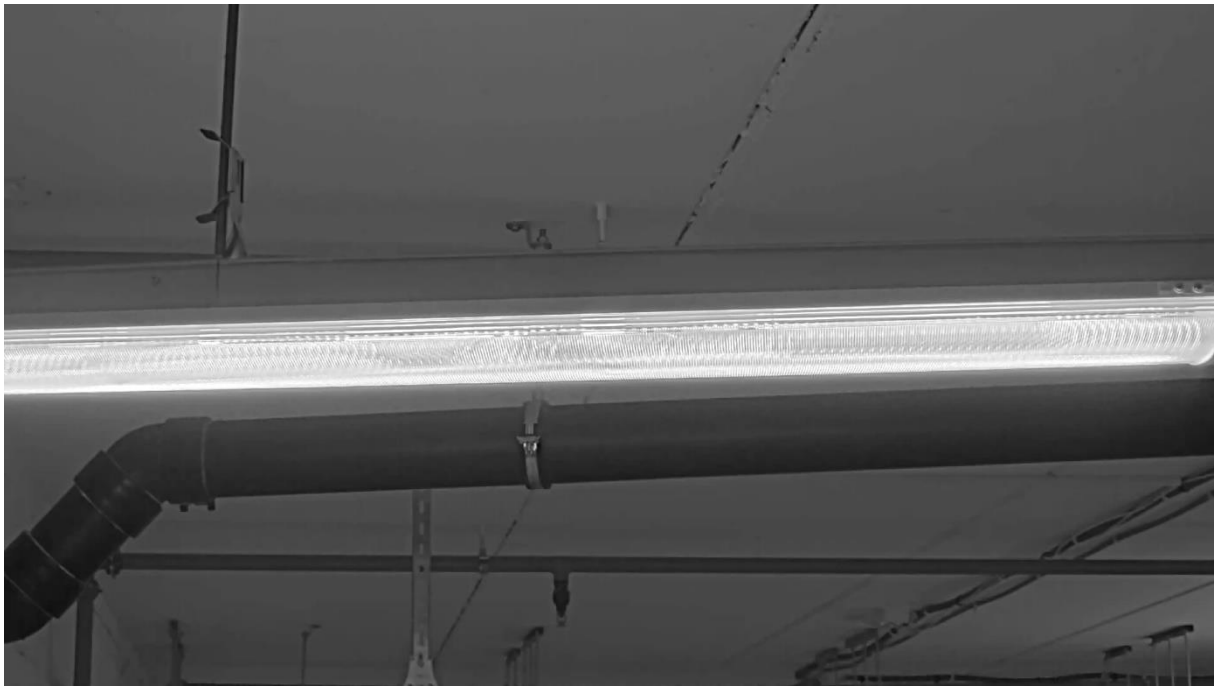

Figure 4.26: Shows when a lamp covers several entire rows of the image. The data matching the image is shown in figure 4.27.

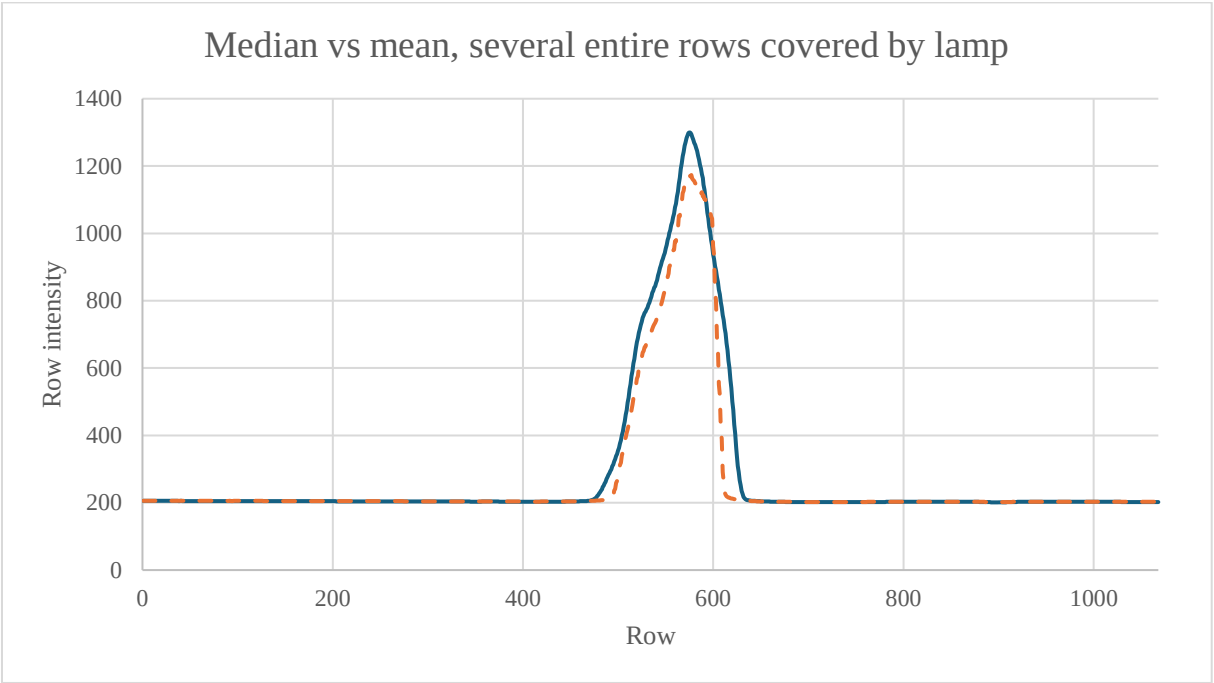

Figure 4.27: Show how the mean and median are similar once the light source covers a band across the image. The dotted line is the median and the solid line is the mean.

4.2.2 Horizon at dusk

Although the functionality of the algorithm during the horizon test described in 3.3.2 were only partially successful, they did verify that the initial detection algorithm works, even with a bright horizon is present. The Maintain Delay method, however, failed at the edge with the horizon. While these tests were performed the algorithm described in 3.1.6 utilized the top and bottom 1/36 of the rows to detect the strobe, this was later changed to just the first and last row. As can be seen in figure 4.28 the top half of the rows are covered by the horizon which is too bright to distinguish the strobe. Furthermore, due to how the threshold was calculated the top threshold ends up being lower than the noise level and causing false positives, this was later addressed with a minimum threshold. With the result being an erratic strobe which jumps in and out of the frame due to the noise triggering the delay.

In figure 4.28, the strobe was still visible, and the detection algorithm could identify it and successfully synchronize the camera's frame rate, although the calculated stable frame rate is not as precise as when the strobe was visible throughout the entire frame. Finally, as can be seen in figure 4.29, once the strobe was in the part of the image with mostly the horizon, it was invisible and undetectable.

Figure 4.28: Shows: the same motive as figure 3.8 but with the IR-strobe turned on and visible.

Figure 4.29: Shows the IR-strobe mostly obscured by the bright horizon.

4.2.3 Nighttime test

The test described in 3.3.3 was performed under more favourable conditions than the test in 3.3.2, and everything worked nearly identically to laboratory testing. Since the motive provided plenty of nearby reflective surfaces E.g. walls of corrugated metal, there was no issue detecting the strobe, see figure 4.30. Since the strobe was visible throughout the image there was no issue with thresholds being set too low. This test should be considered a best-case scenario since it provides a setting similar to the laboratory. This test was only run briefly period to verify that all algorithms described in 3.1 worked as intended.

Figure 4.30: Shows the same motive as in figure 3.11 but with the strobe.

5 Discussion and conclusion

This chapter features a discussion of all the implementation decisions, the laboratory and field tests, functionality with automatic exposure and WDR, possible future work, and finally, a conclusion of thesis is drawn.

5.1 Implementation decisions

5.1.1 Row median vs mean

Initially, this thesis opted to use the mean of the row values as it was assumed that a strobe band would be difficult to detect in many situations and that the mean would allow for an easier detection. This approach worked well in the laboratory with consistent lighting. However, field testing revealed that the mean value was too sensitive to intensity changes in parts of a row. Light sources directly illuminating the camera sensor also skewed the row mean calculation enough to trigger the detection algorithms as validated in 4.2.1. Although this could have been ok if the background never changed, but as explained in 3.1.4, that possibility is always there, and the mean could be more affected than the median in the middle of detecting the strobe.

As discovered and shown in 4.2.1, the median sufficed and the mean was overly sensitive. The PIE-team's original usage of the median proved to the better choice due to the median being less sensitive to large individual changes. The median of the rows does help in dealing with subtle bands which was one of the problems mentioned at the end of 2.7.3. Although the median was chosen, it is fair to note that the mean values were larger overall but by using an adaptive threshold with the median, this was not an issue.

5.1.2 Difference vector

Using the difference between the current frame and the background proved to work well. The original implementation performed the median calculation on every row and checked if it was above a threshold comprised of twice the median of all rows in the image. It then collected line candidates where 20 was required to be considered a strobe band as explained in 2.7.1. This requires the image and strobe band to be rather uniform as something bright as a static line or light source could trigger a false positive. The difference of the current frame to the background rectifies two of the three questions posed by the PIE team explained at the end of 2.7.3.

It rectifies "How to differentiate between the strobe and a fixed band in the image?" by storing the background which contains the fixed band where the difference to the current frame will be 0. The only issue is if the fixed band is brighter than the strobe. The strobe would then have to be synchronized in the parts of the image where it could be detected, which would still work. It also rectifies "Can the band detection be reworked to detect edges instead of the width of the band?" by doing just that. When the difference vector encounters the strobe, it is considered an edge where the intensity exceeds the set threshold.

The difference to the background does however suffer from the problem of a changing background as explained in 3.1.4, where a change will incur a difference in the vector where there is no strobe. Hopefully, the strobe is bright enough to overshadow this issue, but it is there, and as shown in the field tests, the strobe is barely visible in some scenarios. However, Find Strobe Bands is supposed to run only a short while, which does justify this implementation.

A test comparing the two methods of finding the strobe could have been interesting but as priorities lied elsewhere, it was not performed. Anyhow, a test could be comprised of creating a threshold algorithm that would work for both. Then the two methods could be compared in fair conditions. The original with the image median could work as a threshold for both.

An issue noticed late in development was at the edges of the image when some of the strobe was outside the frame. What happened was a miscalculation of the frame rate which resulted in the strobe bouncing in place. E.g. if the strobe has a width of 10 rows and it moves from row 1 to 0, the band centre with the band end at 1 would equal $\frac{1+11}{2} = 6$ and at 0 it would equal $\frac{0+10}{2} = 5$. The problem arises when the strobe then moves to out of frame as the algorithm thinks the strobe width is smaller now that it only sees 9 rows. The band centre would equal $\frac{0+9}{2} = 4.5 = 5$ and only at the next row would it change, $\frac{0+8}{2} = 4$. This is a problem as it returns the same band centre for two rows even though the strobe is still moving. The effect is that no frame rate will be calculated until it reaches every other row.

The bouncing behaviour occurs especially in the bottom of the image as shown in figure 4.11 in 4.1.2. This is likely due to a compounding issue with what was explained in the last paragraph and the data loss at the end of the MA-filter. This was not a major issue as the strobe in most of the test runs had a visible start and end in frame. Moreover, the stability threshold allowed the strobe to move one row in between frames to be considered stable, therefore, it would apply a delay. A well synchronized frame rate would perhaps not be set, but still close if using the mean of the last half of the calculated frame rates.

A discussed solution to this is to return to using the width by calculating how many rows the strobe occupies, which was done in the previous work explained in 2.7. In this implementation, it could be done by taking the absolute value of the band start minus the band end to determine the width when the full strobe was in frame. This could be done iteratively by using a growing value which grows until it reaches the strobe's width if it is outside of frame. The strobe's width is constant due to the fixed exposure. A further problem is that the strobe's centre could be outside the frame where an artificial extension to the frame could be implemented so that the centre could in theory be outside the frame.

A less thorough solution would be to disallow any frame rate calculations if some of the strobe is outside the image. This could be done by checking if the band start index is the first row or the band end index is the last row. Or by calculating the width of the band and only allowing synchronization if the entire strobe band is in the image.

5.1.3 Varying intensity in the strobe band

As described in 4.1.2, the strobe sometimes displayed a bouncing behaviour. One cause of this, due to how the thresholds were implemented, was that the threshold could sometimes be too large for some parts of the strobe to be detected. This was due to the threshold being calculated as the maximum intensity recorded over many frames. E.g. in figure 5.1 where the dotted threshold is calculated to 765 which results in the band start being at row 465. The band end is then detected as row 511 and the centre is calculated to 498 even though there is more of the band. Now, if the band then moves down a few rows in consecutive frames, the centre may be drastically shifted as that first peak is no longer in the band, resulting in the bouncing behaviour. Alternatively, as represented by the dashed line, where all parts of the strobe are above the threshold, the algorithm has no problem detecting correctly and no bouncing behaviour is observed.

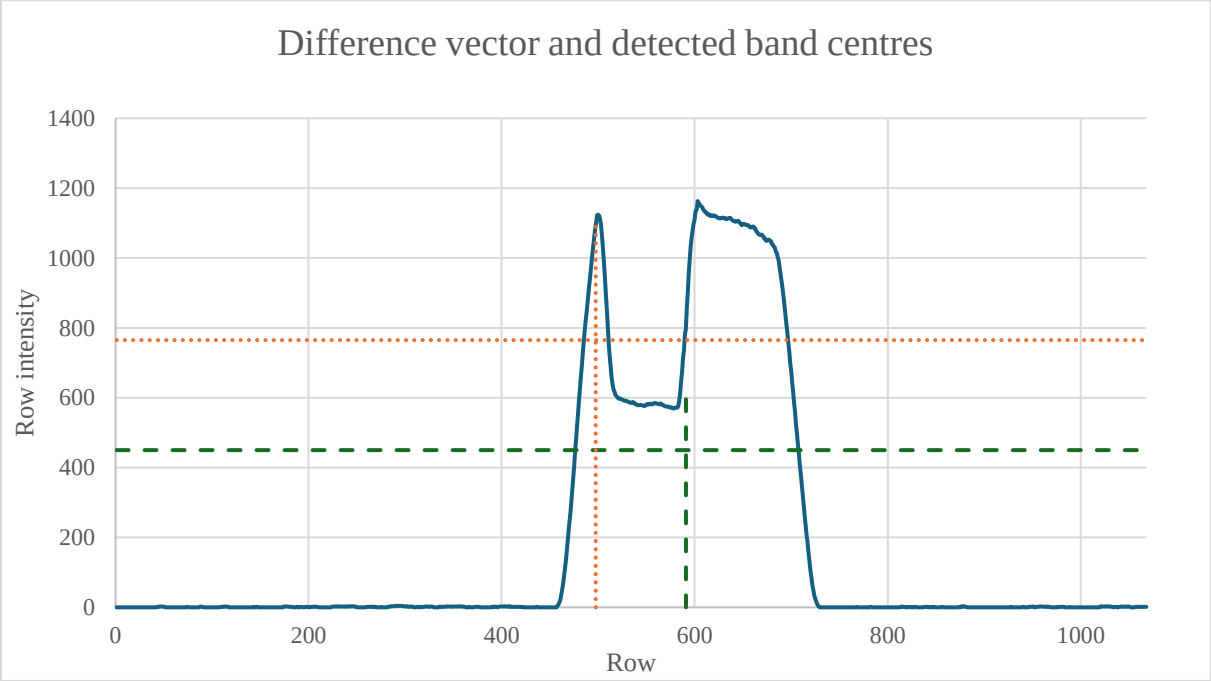

Figure 5.1: Shows the difference vector with two hypothetical thresholds where the dotted line would result in the incorrect band centre and the dashed line would result in the correct.

This happened more often in brighter environments as there were less headroom for the difference to the background. A compromise could be to lower the threshold by dividing something larger than 2 as shown in figure 5.2, which runs the risk of false positives. It all depends on the situation a camera is in.

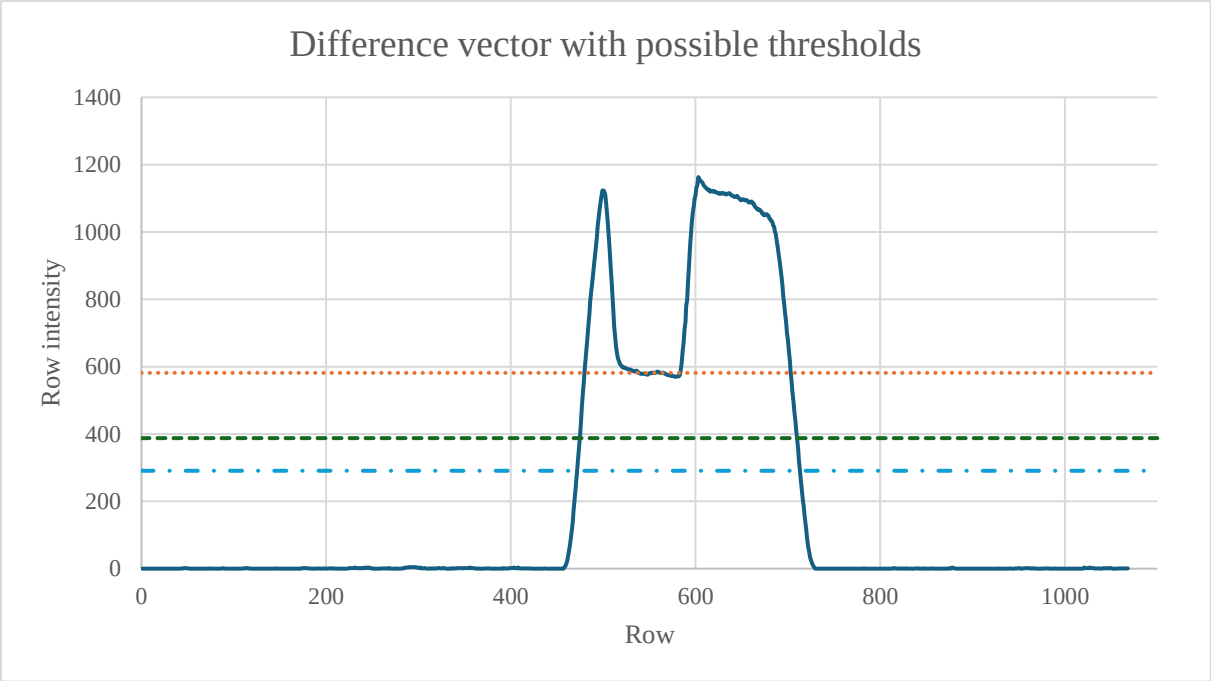

Figure 5.2: Shows the same data as in figure 5.1 but with the thresholds calculated as the maximum recorded intensity in this frame divided by either 2, 3 or 4.

5.1.4 The IIR-filter and frame rate convergence

Throughout development the IIR-filter remained unchanged as it fulfilled the requirements and did not cause issues when developing other parts of the algorithm. However, when testing began, the IIR-filter was more closely inspected, and it was brought to attention that initially for the N frames depending on the α_n, the filter averaged the collected values and functioned as a finite impulse response filter until it reached α_n where it then functioned as the IIR-filter described in 2.6.3.

When using the average for the first N frames, the system converges to the target faster which was something the PIE-team had intentionally implemented. Initially from equation 2.8 in 2.7.2, when $N_{frames} = 1$, n_{strobe}^{IIR} is just equal to n_{strobe}. In the next frame,

$$n_{strobe}^{IIR2} = \frac{1}{2} n_{strobe}^{2} + \frac{1}{2} n_{strobe}^{IIR1} = \frac{1}{2} n_{strobe}^{2} + \frac{1}{2} n_{strobe}^{1} = \frac{1}{2}(n_{strobe}^{2} + n_{strobe}^{1}),$$

then,

$$n_{strobe}^{IIR3} = \frac{1}{3} n_{strobe}^{3} + \frac{2}{3} n_{strobe}^{IIR2} = \frac{1}{3} n_{strobe}^{3} + \frac{2}{3}\frac{1}{2}(n_{strobe}^{2} + n_{strobe}^{1}) =$$

$$\frac{1}{3}(n_{strobe}^{3} + n_{strobe}^{2} + n_{strobe}^{1}),$$

and so on, until it reaches α_n where the actual infinite impulse response begins and repeats the calculation at that value. A test with the current implementation and with just an IIR-filter could be interesting to perform on the different strobe stabilities.

The frame rate setting command is of the same character as the delay setting command. If these are similar, perhaps there is a small delay before the frame rate update occurs as there is with the delay command. Therefore, a different approach to calculating the strobe frequency could be implemented. Instead of updating the frame rate by sending the command every frame, the strobe frequency could be calculated in the same way by comparing its position in the current frame to the previous frame. Then the frequency could be calculated and only one frame rate command is used.

The frame rate would have to be fairly synchronized to the strobe frequency already to be able to do this reliably. If the frame rate is way off the strobe frequency, a sequence of test frame rates could be set ranging from perhaps 20 to 30 fps with steps of 1 fps every test. Then when the slowest strobe velocity is measured, the frame rate calculations could begin properly. This was in fairness never an issue but perhaps something to consider if the pipeline is occupied.

5.1.5 The moving average filter

Due to its simplicity to implement and its sharp step response, the MA-filter was chosen to filter the different parts of the algorithm. Since the filter aided in the detection and functioned as desired, it was never replaced. Throughout this thesis there has been issues with the artefact “bouncing” while in the bottom of the frame, as can be seen in figure 4.11. In the final stages of testing, one cause of this behaviour was determined to be the MA-filter. This was discovered when plotting the band position in the code where only 1068 out of the 1080 rows were recorded. Due to this implementation, each value in the final window would be treated as the same row, as this made it possible to process all available data at the cost of inaccuracies in the output.

This was thought to be an acceptable trade-off, as this filter provided good noise reduction and retained the step response when the artefact appeared. But treating the final rows as the same after filtering caused the band centre to move several additional rows, equal to half the size of the filter window. After this was discovered the window size was reduced to minimize this effect at the cost of some noise reduction, alleviating some of the bouncing behaviour. Moving forward this should be addressed, by either redesigning the current implementation or replacing the MA-filter. A redesign of the filter could be to reflect the calculations when reaching the final window by starting at the other end of the vector and moving the opposite direction until reaching the next window.

In the final stages of this thesis, the fluorescent light in the laboratory broke and was replaced by a continuous IR-light from another camera. This showed that the noise level of any background illumination had a dramatic effect on the row median variations. One possible improvement for future work would be to analyse the variations in each setting and then evaluate whether the MA-filter is needed. Then the filter could be automatically applied when needed.

5.1.6 Deciding thresholds

Since the algorithm should work in a variety of environments and on different cameras, static thresholds would work poorly outside of the laboratory. Therefore, an adaptive threshold was necessary to ensure functionality under various lighting conditions without having to adjust the algorithm.

One edge case is when the image is noisy and the strobe artefact is of low intensity, which could lead to the Find Strobe Bands threshold being set within the range of noise. To counteract this, a minimum threshold of 40 was set. An artefact below that threshold was deemed to be dim enough to not have any effect on the forensic value of the image. This depends on the application; the sensitivity could be modified depending on its environment perhaps.

Although the threshold for detecting a strobe was adaptively set, the algorithm is dependent on several different thresholds. One example of this is the threshold to determine when to update the background. Since designing statistical analysis for all these thresholds would be cumbersome, the background threshold for example was simply set as half of the strobe threshold at the specific edge. This could become an issue in extremely noisy environments, this has however not been the case after the initial adjustment so far.

As described in 3.1.5, the thresholds were determined by the maximum intensity found in the entire image. This could sometimes lead to the algorithm being unable to detect the strobe in certain parts of the image if the strobe intensity varied enough as the example in 5.1.2 explained. Therefore, the threshold could have been improved by calculating a threshold every frame as that would consider the varying strobe intensity throughout the image. This would however increase the computational requirements. Although, the effect could be measured and is perhaps a worthwhile trade off.

5.1.7 Number of rows to monitor in Maintain Delay

Regarding how many rows to check every frame during Maintain Delay, as explained in 3.1.6, only the first and last row in the frame was monitored. Justifications for this was firstly, that the median of that row filtered potential noise triggers. Secondly, the strobe being a consistent band across the entire image most of the time, which significantly affected those specific row values as seen in figure 3.4 in 3.1.2.

When first implemented, the Maintain Delay function monitored 1/36 of the rows in top and bottom, as this gave the algorithm plenty of space to detect the reappearing artefact. But due to the synchronization performed in Find Strobe Bands this proved unnecessary as the artefact moved slowly in frame when reappearing. Therefore, it was decided to only monitor the top and bottom row. However, the background update check still checks 1/36 of the rows.

A problem potential which was not encountered is that the first and last row is somehow obscured or something in that one row specifically blocks the strobe. Now the argument can be made that it is good practice to check more than one row. Perhaps even making sure that it really is the strobe by checking if e.g. 10 rows all exceed the threshold and only then apply the delay. This could be useful alongside potentially cutting these 10 rows from the final image so the strobe will never be seen if everything works as it should.

Before implementing the requirement that the background only could update if the strobe was far enough into the blanking window, there was the ill-timed possibility that the strobe could drift back during the update. This sometimes resulted in the strobe being part of the background. Then, it could no longer distinguish the strobe which resulted in it not applying a delay, and the strobe slowly drifted across the frame. A small reprieve was that, because the background did update, the other side of the frame would be the actual background and when the strobe eventually reached it, a delay would trigger. Additionally, when the real background reappeared, the update background algorithm would trigger as the stored background contained the strobe which resulted in a large difference and triggered the criterion.

5.2 Laboratory tests

From the test results with the unstable trigger, the mean of the last calculated frame rates, and the stable trigger presented in table 4.8 in 4.1.3, a well synchronized frame rate is defined as when at least 12 000 frames between updates have passed on the P1385. This number is decided upon from the data in table 4.8 where the lowest number was 12696 when accounting for outliers and possible failures. This equates to 8 minutes until the reappearance of the strobe at 25 fps. 12 000 might seem high but the mean is larger which provides a high chance of it landing in that vicinity. The goal after all, is to find a well synchronized frame rate.

The laboratory tests show that an unstable strobe benefits from using a smaller α_n along with the mean of the last half of the frame rates. On the other hand, a stable strobe would have sufficed with just the IIR-filter and a larger α_n for a quicker convergence. The final implementation therefore features the former as the strobe stability is unknown and dependent on the environment the camera is in.

5.2.1 Frame rate accuracy tests

While performing the second round of the α_n tests, it became clear that the sample size in the first round of testing was insufficient. As could be seen in table 4.1 in 4.1.1, an α_n value of 0.03 was evaluated as the best. However, testing at a large sample size as shown in table 4.2, showed that the initial tests where susceptible to statistical anomalies skewing the results.

After the second round of tests, an α_n value of 0.03 was chosen despite not having the lowest frame rate standard deviation. It was also very affected by large outliers where it took several hours before the strobe reappeared. This is both an advantage and disadvantage as a well synchronized frame rate is desired but also skews the test results. A more accurate description could be to use a percentage of highs and lows that confine the outliers, resulting in more evenly spread data.

The possibility of the strobe moving out of frame as the frame rate was synchronized with the strobe frequency was present in every test. As the Find Strobe Bands method had not finished yet, this resulted in having to wait for the strobe to slowly move back into the image to finally set the frame rate and properly delay it. It can be argued that this is acceptable as the strobe is out of the picture and when it eventually moves in, it will be properly delayed. This, however, skews the test results as it increases the overall test time and prolongs the time until stable. A consideration was given to if these outliers were to be ignored when showing the data, but it was decided to keep them. For some of the tests this occurred more often, which would alter the number of datapoints between tests if these were considered outliers. The decision was therefore made to keep all the datapoints to show that this issue was larger at smaller values of α_n as shown by the stabilization times in figure 4.5 in 4.1.1.

A solution to this is to find a way to slightly alter the frame rate if it moves out of frame before being eliminated. One possible solution is to use the current band position to determine if it is in frame as this is a vector of one element that is empty without the strobe. Then it could check if this vector is empty and then alter the frame rate a bit. This could be coupled to the proposed solution to the problem where the strobe is at an edge, when the frame rate calculations are prohibited if some of the strobe is outside the image.

5.2.2 Mean of the last calculated frame rates

One could argue that the better alternative is utilizing the mean of an amount of the frame rates when synchronized like this thesis did, in addition to a larger α_n that converges faster but oscillates more around the target frame rate. This concept could be used along with a strobe with a more stable internal trigger due to the lack of noise. Another alternative when using a stable trigger is to simply use the final value that the IIR-filter outputs as in that case, the final frame rate barely changes as shown in figure 4.13 in 4.1.3 since the IIR-filter is already averaging and converging.

5.2.3 Stable trigger with the function generator

When using the function generator as the strobe's trigger, the IIR-filter performed as desired. As seen in figure 4.13, the last calculated frame rate could be set as the final frame rate instead of the mean of the last 100. This mean might contain values from when it was still moving, worsening the result. The implemented IIR-filter already works as an averaging filter before becoming a proper IIR-filter as described in 5.1.4.

To fill out table 4.8 in 4.1.3, a test with the function generator and only the IIR-filter without the mean of the last half of the frame rates would be interesting. Unfortunately, this test was not performed due to time constraints but would be quick to implement to see if only the IIR-filter would suffice with a stable strobe. Furthermore, α_n could probably have been increased which would speed up the system and decrease the chance of the strobe moving out of the image before being delayed as explained in 5.2.1.

The stabilization times in figure 4.5 in 4.1.1 are similar down to around $\alpha_n = 0.05$, after that a drastic increase in the average time required to stabilize can be observed. As a more stable strobe require less time to set the frame rate, it would be preferable to set the $\alpha_n \geq 0.05$.

5.2.4 Issues with the delay command

As mentioned earlier, there were issues when trying to apply delays as the command sometimes failed. The P1388 was abandoned in testing due to this problem occurring too often to collect any valid results. Those issues are also present on the P1387 and P1385 but to a far lesser extent. As mentioned in 4.1.4 and shown in table 4.9, the P1385 managed a failure rate around 1 % when a delay was applied. Although this is an issue affecting the final product of this thesis, it is likely caused by a pipeline issue or a bug somewhere else in the architecture and deemed outside of the scope of this thesis.

With the requirement set in 5.2, 12 000 frames must pass between updates to consider the strobe stable. This equates to 8 minutes between updates, and with a failure rate of 1 %, the expected number of failures in 24 hours is then $\frac{24 \cdot 60}{8} \cdot 0.01 = 1.8 \approx 2$.

5.2.5 Camera differences

As mentioned earlier in 4.1.5 the cameras tested did not behave identically. E.g. the delay command failed to apply a delay before the next exposure at a different rate on the three different cameras. In the case of the P1385 it failed around 1% of the times it was called, but on a P1387 this failure rate was closer to 20 %, and finally on a P1388 the command failed nearly every time which was why testing was excluded from this one. Furthermore, at a given frame rate, the time to read each row differed in the cameras. This is due to the increased number of rows. However, an increase in the number of rows to read did not linearly decrease the time taken to read one row. This led to less blanking time, and a smaller blanking window.

The α_n tests also showed that different values tested performed unevenly on the two cameras, as can be seen in figures 4.1 and 4.3 in 4.1.1. The P1387 was less stable for any given α_n value, possibly due to the decreased blanking time compared to the P1385. Since the P1387 needs to process almost twice as many pixel rows as the P1385 in a cycle, it is likely that the smaller blanking window in combination with the strobe's inconsistency is the cause of the drastically shorter time between the first delay and the artefact reappearing. The P1387 pipeline must also handle more data which could delay certain instructions more.

Another reason is possibly that, as the number of pixels that is processed increases, the delay between a function being called and executed increases. And since the synchronization algorithm calls the function to set a new frame rate every frame, it is suggested that this might cause an offset between the optimal frame rate and the mean frame rate as seen in figure 4.3.

5.3 Field tests

5.3.1 Parking garage

As mentioned in 4.2.1, testing in the parking garage showed that the initial detection algorithm can detect a very weak strobe, as it is enough to detect the strobe two frames in a row to adjust the cameras frequency. Due to the IIR-filter, over time this will be enough to stabilize the artefact. However, as figures 4.20 and 4.21 in 4.2.1 shows, it is questionable whether it is necessary to eliminate an artefact that weak, as it does not impact the quality of the image to a large extent.

5.3.2 The horizon issue

When testing in an outside environment right after dusk with the sky in the image, it was bright enough to fill the entire top half of the image as shown in figures 4.28 and 4.29 in 4.2.2. This maximized the median row intensity values in the top rows which prohibited the algorithm to detect any difference and furthermore, there was nothing for the strobe to reflect off.

Even though the strobe only was discernible in half of the frame, this was till enough to detect and stabilize it in Find Strobe Bands. However, once in Maintain Delay, the strobe could no longer be detected in the top as this was obscured by the horizon. This in turn caused the delay to fail when the artefact reappeared from the side covering the horizon.

One way of handling this issue would be to identify the horizon during the initial stage before strobe elimination, and then if the horizon obscured the top or bottom row, move the monitoring point to where the horizon ends. This would leave that part of the image unmonitored, but since the strobe would not be discernible from the background in these rows, that would be of no concern.

The introduction of the minimum threshold would have stopped the algorithm from applying noise triggered delays. But would in turn be unable to see the strobe which would then travel across the image if moving from the top to the bottom. This is rectifiable by letting the algorithm find the frame rate. Then making sure that it is faster than the strobe frequency by slightly increasing it to only allow it to approach from the bottom as shown in figure 3.7 in 3.1.6.

5.3.3 Nighttime test

When testing in a fully dark environment, all parts of the program worked as intended and no issues where observed. This test was however performed under favourable conditions with low to non-existent background illumination and an intense strobe. Furthermore, the environment in this test provided plenty of objects to reflect the strobe light, making it a near certainty that it would be detected and eliminated correctly.

Due to the lack of any continuous IR-illumination or any other substantial light sources, the strobe artefact appeared with an extreme contrast to the background as can be seen in figure 4.30 in 4.2.3. In a more realistic scenario, the rolling shutter camera would have its own illumination, decreasing the contrast between the artefact and the background. But as previously mentioned in 5.3.1, if the artefact is intense enough to compromise the image quality, it is detected and eliminated.

The nighttime tests were performed the same date as the tests at dusk, with the same version of the algorithm, which has since been updated several times. It is possible that new issues have been introduced, however that is unlikely as the parking garage testing were performed with the latest version of the program and under less favourable conditions. But it is worth mentioning, since subtle edge cases can be difficult to predict.

5.4 Automatic exposure

Regarding automatic exposure, what was preferable was to never touch this setting in the first place and let the algorithm handle it, as sudden exposure changes were noticeable and undesirable. As explained in 3.1.3, Find Strobe Bands was unable to store a correct background as it constantly changed with automatic exposure. The solution to this entailed investigating a way to find a background even though the exposure settings could change.

Firstly, the algorithm just picked a background as a starting point, from there the background could be further updated. Secondly, the algorithm could calculate a rough frame rate which was set a bit lower than the actual frame rate. Now the strobe moved in the image but slowly enough to stay out of the image long enough for the exposure time to settle. This is because the exposure time spiked when the strobe moved out of the image. Lastly, the exposure time was given enough time to settle which could then be used as a criterion for when to store the background. The optimal case was when the now stored background exposure time was as close as possible to the current exposure time which resulted in an exposure time factor of one.

This worked when the initial camera frame rate was close to the strobe frequency. But problems arose when the initial frame rate differed too much as the exposure never decreased enough to find a lower image median, resulting in never storing a new background. This implementation was discarded as it pulled resources away from other parts of development.

It is theoretically possible to modify the Maintain Delay method to work with automatic exposure on and an attempt was made as described in 3.1.7, however due to lack of insight in how the exposure algorithm worked at the time, this attempt only compensated for changes in exposure time. If the exposure gain and the lens aperture along with the exposure time was considered when calculating an exposure factor, it should be possible to modify the code to be able to run with automatic exposure on.

Another issue with automatic exposure was the lack of an outport to the exposure code, meaning that if changes were to be made in exposure time, the web user interface service needed to be hijacked and data injected that way. This requires root access and debugging turned on, which is not viable in a consumer product. This was discussed but set aside alongside the attempts with automatic exposure.

Ultimately, automatic exposure contains many variables that are used to control the sensor exposure which were unknown at the time of development. This is something which could be tested further such as the implementation of e.g. a gain factor and an aperture factor beyond the exposure time factor.

5.5 Wide dynamic range

As described in 2.5, WDR utilizes the blanking time between exposures to perform a second exposure with a different exposure time or at a different amplifier gain, which is then used to increase the contrast range of the final image, in practice this leads to twice the normal frame rate and a drastically shorter, if any depending on camera model, blanking time where the sensor is inactive.

There was an attempt to modify the algorithm to work with WDR on and off, and there was potential in creating a working model. However, when testing with the P1387 it was realized that the implementation of WDR on the P1387 made it impossible, as there is close to no blanking time between frames since it is used for a long exposure as illustrated in figure 2.4 in 2.5. The P1385 uses another implementation of WDR as illustrated in figure 2.5 in 2.5, which theoretically makes the strobe elimination possible as long as the exposure time and frame rate is low enough. At slower frame rates (up to around 28 fps), there was space in the blanking window to place the strobe when tested, but this window was shrinking with higher frame rates, and without a reliable blanking window to place the strobe, the algorithm was unable to push the strobe artefact out of frame, therefore the decision was made to exclude WDR from the algorithm.

Another possibility is to not use the second exposure to increase the dynamic range of the image, but instead use it to keep track of the strobe after it has been eliminated. This idea was brought up by the supervisor at Axis in the early stages of this thesis. It was however deemed a complicated solution which would require a fair amount of modification in the sensor's backend code. It was suggested that the focus of this thesis should be on improving the initial detection and the Maintain Delay method.

5.6 Ethical and environmental aspects

Surveillance is a hot topic in today's world, and as a manufacturer of network video solutions, Axis is entangled in all the discussions about privacy and the ethics of surveillance. There are laws in place such as GDPR that regulate companies like Axis from using testing footage for anything other than development of their products. Axis is obliged to destroy any footage of someone that requests it.

The purpose of this thesis is to remove an artefact that reduces the forensic value from cameras. One could argue that this work increases overall surveillance and reduces privacy. On the other hand, one could argue that it benefits the field of forensics and criminology, the robustness of surveillance footage and customer satisfaction.

From an environmental perspective, as explained in the introduction, this work aims to increase the longevity of rolling shutter cameras as global shutter cameras with strobing IR-lights are becoming more prevalent. This opposes planned obsolescence and reduces electronic waste as customers can continue using their existing cameras.

5.7 Future work

A limitation cast upon this work was having only one camera at each resolution. Having several P1385's could have allowed for concurrent comparative testing without having to upload new code and recreating test setups. Having the ability to upload new code whilst field testing would also have been a welcome option. More P1387's and P1388's could have confirmed that it was a code issue that led to inconsistencies and not specific cameras. Furthermore, all testing was performed on only the P1380 series of Axis cameras, all with the same processor. To ensure reliability, a wider range of products require testing and verification. This however is out of the scope of this thesis.

As previously mentioned throughout this chapter, this thesis leaves room for improvement and other implementations, mainly regarding automatic exposure and WDR. These functions are part of the standard functionality of Axis cameras and a customer might expect to be able to utilize them. However, the algorithms in this thesis require them to be restricted or deactivated.

Regarding automatic exposure, as described in 5.4, there was an attempt to factor in the automatic exposure in the calculations, however the exposure gain and aperture was unknown at the time which left that algorithm flawed. By taking all the variables of the exposure into account, it should be possible to adjust the algorithms so that automatic exposure could be active in conjunction with the artefact mitigation.

As for WDR, depending on the implementation of WDR on a specific camera, it is theoretically possible to keep WDR active. But as described in 2.5, there are different ways to increase the dynamic range, and if using the variant with different exposure times, then artefact mitigation is likely impossible by utilizing the blanking window

As mentioned in 5.3, an issue with how the delay is applied does exist and needs to be addressed before these algorithms can be used in a product. While this bug still exists, the best-case scenario is a failure rate of around 1 %, where the artefact reappearing and slowly moving across the frame. That is unacceptable in a finished product, as it reduces the forensic value of the camera. Furthermore, the delay issue made the algorithm unusable on the P1388, which does raise concerns about the generalization of the algorithms developed.

Currently, the frame rate is updated every frame while the artefact is present. To reduce the number of requests sent through the pipeline. This could be re-designed so that data is collected over many frames. The strobe frequency could then be calculated, and the cameras frame rate updated only once as explained in 5.1.4. As other parts of the algorithm described in 3.1 already works in a similar fashion, by collecting data over many frames, this would have a small effect on the time needed to stabilize the strobe artefact.

The significant digits of the row time affect how precisely the frame rate is calculated. Perhaps knowing the number of digits could aid in the frame rate calculations by seeing exactly which frame rate the algorithm calculates every time. Then perhaps another IIR-filter could run in the background which adapts the frame rate continuously instead of setting it only once.

The stabilization time affected the runtime of the Find Strobe Bands method. But the runtime also depended on the implemented frame counters explained in 3.2.3. These can be altered and most likely lowered but this was not a priority and therefore set aside but should be tested in the future as an optimization effort. Another low priority were the different counters that were only quickly tested, these can and should be optimized.

5.8 Conclusion

In conclusion, an algorithm was developed that reliably detects an artefact from a strobe, then eliminates it and keeps it eliminated. The algorithm detects the strobe artefact, synchronizes the frame rate to the frequency of the strobe and delays the exposure to capture frames when the strobe signal is low. It also monitors the top and bottom row of the image for when the strobe eventually reappears and applies a new delay to compensate. This solves problem 1, 2, 3 and 4 expressed in 1.3. Moreover, solutions to the three problems the PIE team raised for future work which were explained at the end of 2.7.3 have all been realized.

Many tests were performed to confirm that the algorithm would work with the unstable internal pulse trigger of the strobe, both in the laboratory and in the field. It worked even better when using a function generator as a high precision trigger. This solves problem 5.

Continuously changing environments from the laboratory to the field aided in recognizing unexpected problems and solving them. This resulted in a program that always worked in the laboratory as well as in many different environments with exceptions of course. The tests in the field always varied in the form of brightness and background changes, changing lighting conditions, moving objects and people, etc. This answers problem 6.

Many issues in the field were discovered and adapted for by in Find Strobe Bands implementing filters such as the median and the moving average, adaptive thresholds, and infinite and finite impulse response filters. In Maintain Delay the adaptations were to check if the frame rate was well synchronized by counting frames, recognizing that the strobe only could appear at one edge at a time and creating a method to check if the background changed and updating it and the thresholds accordingly, which solves problem 7 and 8.

In the field, the algorithm managed to function at varying distances and angles between the strobe and the camera. Although limitations were still present if the strobe was obstructed, too dim or inconsistent in its light output. This answers problem 9.

To summarize, a software solution has been created to solve the problem of an image artefact in the form of a white band from a strobe, which fulfils the goal and purpose of this thesis.

6 References

[1] C. Tormo Lluch, “Energy Efficiency and Power Consumption Improvement of IR Illumination for Surveillance Cameras,” M.S. thesis, EECS, KTH, Stockholm, Sweden, 2018. Accessed: May 8, 2024. [Online]. Available: https://kth.diva-portal.org/smash/record.jsf?pid=diva2%3A1251963&dswid=9146

[2] K. D. Stefanov, “Introduction-what is an image sensor and what does it do?” in CMOS Image Sensors, 1st ed. Bristol, United Kingdom, IOP Ltd, 2022, Ch. 1, sec. 1, pp. 1-1–1-2.

[3] K. D. Stefanov, “MOS transistor” in CMOS Image Sensors, 1st ed. Bristol, United Kingdom, IOP Ltd, 2022, Ch. 1, sec. 8, pp. 1-38–1-40.

[4] M. Wolf, “Image Sensors” in Smart Camera Design, 1st ed. Berlin, Germany, Springer, 2018, Ch. 3, sec. 4, pp. 73–91.

[5] M. Wolf, “Practical Image Capture” in Smart Camera Design, 1st ed. Berlin, Germany, Springer, 2018, Ch. 2, sec. 8, pp. 50–65.

[6] K. D. Stefanov, “Readout Modes” in CMOS Image Sensors, 1st ed. Bristol, United Kingdom, IOP Ltd, 2022, Ch. 5, sec. 2, pp. 5-2–5-4.

[7] “Global & Rolling shutters”, red.com. https://www.red.com/red-101/global-rolling-shutter (accessed April 15, 2024).

[8] Axis Communications AB, Lund, Sweden, Axis P1385 Box Camera. Accessed: 22/04-2024 [Online]. Available: https://www.axis.com/dam/public/f3/76/5c/datasheet-axis-p1385-box-camera-en-US-431061.pdf

[9] Axis Communications AB, Lund, Sweden, Axis P1387 Box Camera. Accessed: 22/04-2024 [Online]. Available: https://www.axis.com/dam/public/98/32/2e/datasheet-axis-p1387-box-camera-en-US-435788.pdf

[10] Axis Communications AB, Lund, Sweden, Axis P1388 Box Camera. Accessed: 22/04-2024 [Online]. Available: https://www.axis.com/dam/public/72/bb/d0/datasheet-axis-p1388-box-camera-en-US-431498.pdf

[11] Axis Communications AB, Lund, Sweden, IR in surveillance, (2023). Accessed: 04/05-2024 [Online]. Available: https://www.axis.com/dam/public/a0/56/91/ir-in-surveillance-en-US-398429.pdf

[12] Raytec Ltd, Northumberland, United Kingdom, Pulsestar VTS. Accessed: 20/04-2024 [Online]. Available: https://www.raytecled.com/wp-content/uploads/2022/05/Datasheet%E2%80%93PULSESTAR-VTS.pdf

[13] P. Stjärneblad, O. Tynélius, M. Scherlund, O. Synnergren, “Sync a rolling shutter sensor to a strobe light to avoid seeing the flash in the image”, in Innovation Week, 2023, unpublished.

[14] Axis Communications AB, Lund, Sweden, Wide dynamic range “WDR solutions for forensic value”, (2022). Accessed: 19/04-2024 [Online]. Available: https://www.axis.com/dam/public/ba/52/11/wide-dynamic-range-en-US-379535.pdf

[15] R. C. Gonzalez, R. E. Woods, “Digital Image Fundamentals” in Digital Image Processing, 4th ed. New York, NY: Pearson, 2018, Ch. 2, sec. 4, pp. 65–71.

[16] K. Vännman, A. Jonsson, “Slumpmässig variation” in Matematisk statistik, 3rd ed. Lund, Sweden: Studentlitteratur, 2022, Ch. 1, sec. 3, pp. 17–30.

[17] S. W. Smith, “Recursive Filters” in Digital Signal Processing, 1st ed. Burlington, MA, United States of America, Elsevier Science, Ch. 19, pp. 319–332.

[18] R. C. Gonzalez, R. E. Woods, ”Image Segmentation” in Digital Image Processing, 4th ed. New York, NY: Pearson, 2018, Ch. 10, sec. 3, pp. 743–747.

[19] S. W. Smith, “Moving Average Filters” in Digital Signal Processing, 1st ed. Burlington, MA, United States of America, Elsevier Science, Ch. 15, pp. 277–279.

[20] P. Stjärneblad, private communication, Feb–May, 2024

7 Appendix

7.1 Camera output

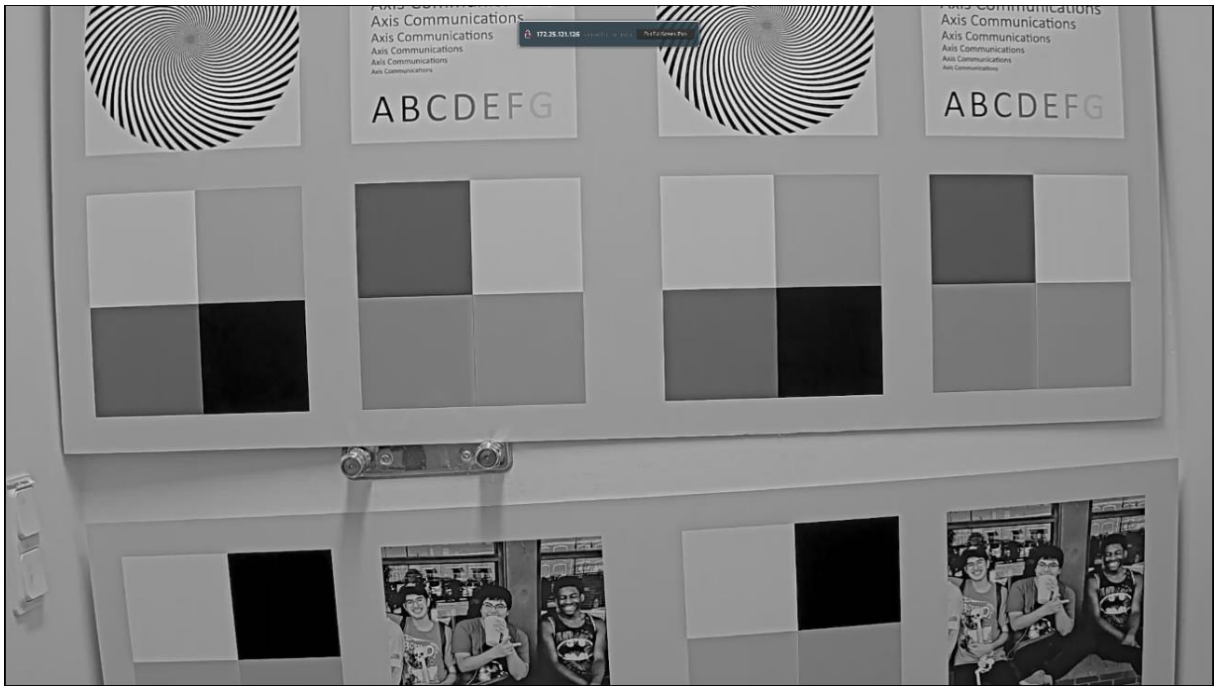

Figure 7.1: Camera output without the strobe in the laboratory.

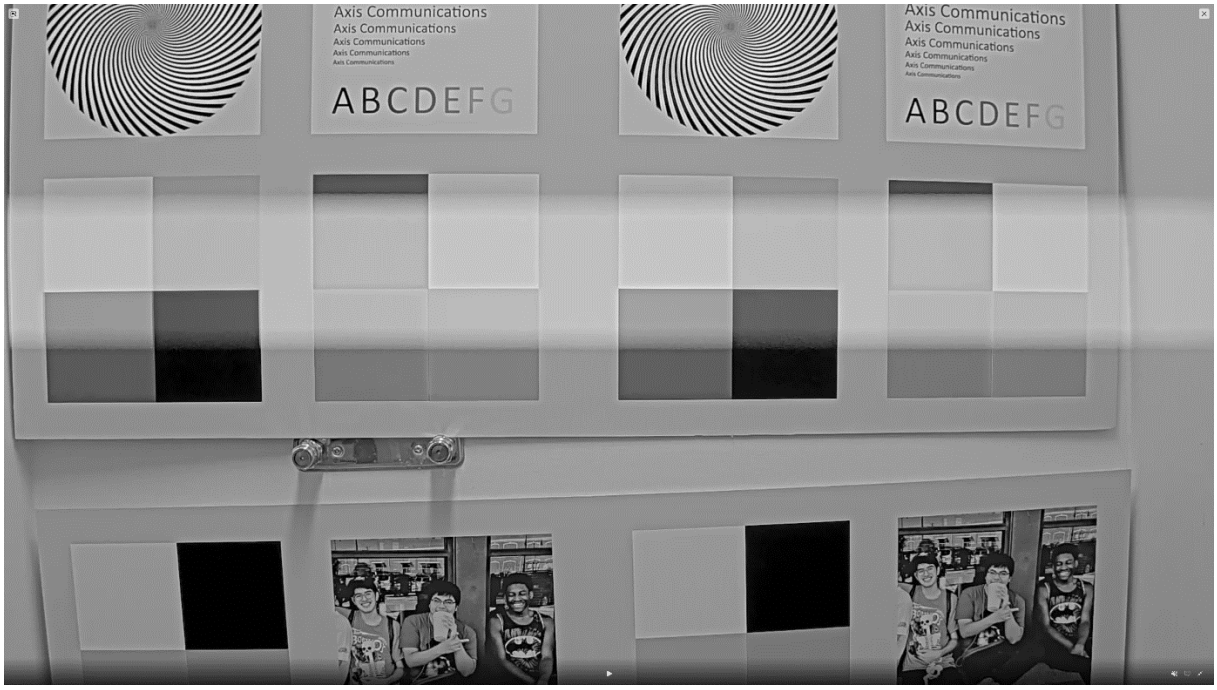

Figure 7.2: Camera output with strobe at 20 % intensity in the laboratory.

7.2 Residual data

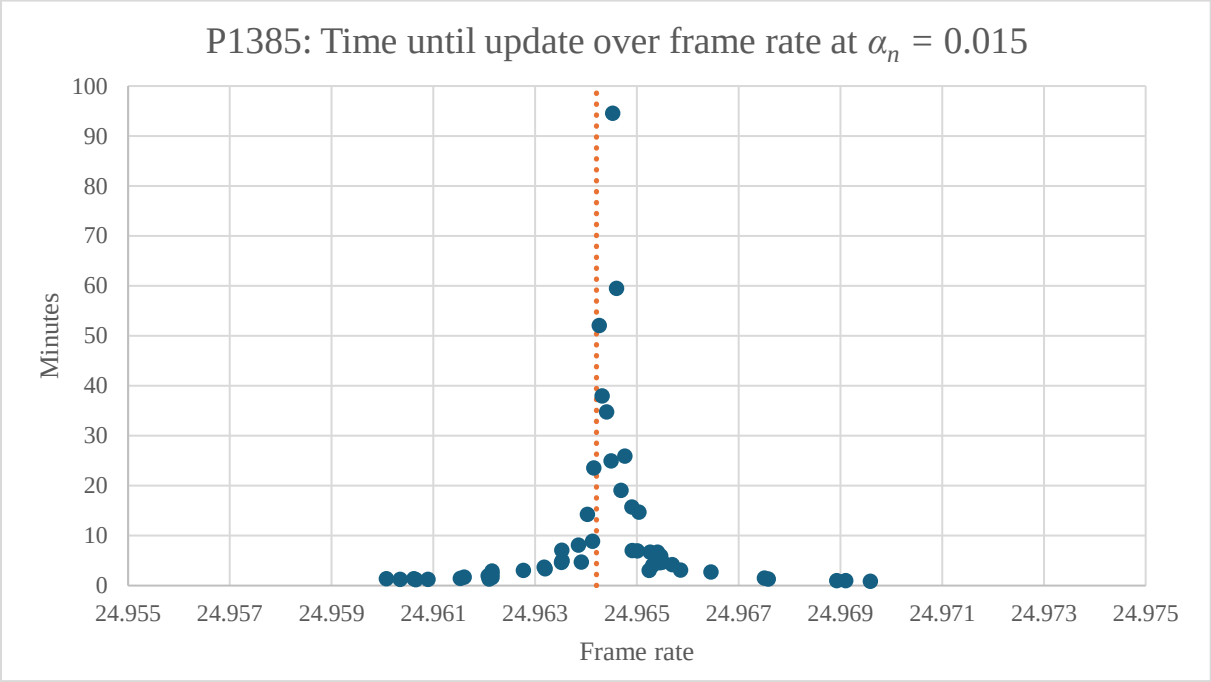

Figure 7.3: Shows the amount of time in minutes until update for each calculated frame rate at $\alpha_n = 0.015$ for the P1385. The dotted line represents the mean calculated frame rate.

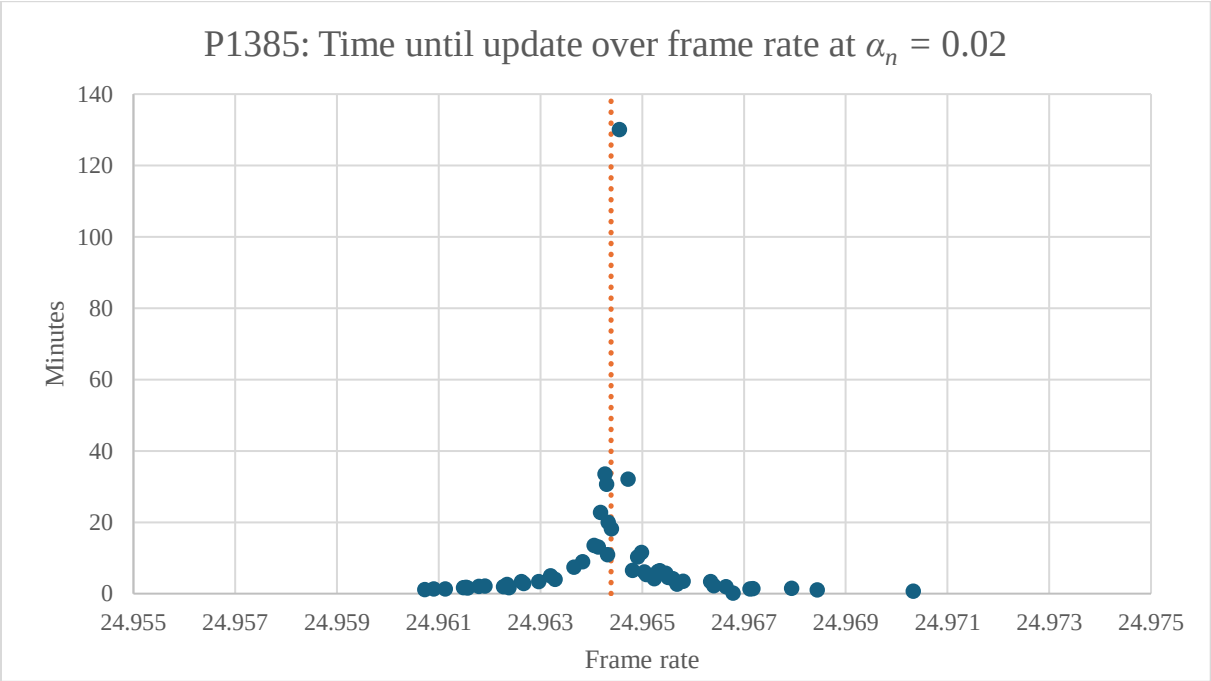

Figure 7.4: Shows the amount of time in minutes until update for each calculated frame rate at $\alpha_n = 0.02$ for the P1385. The dotted line represents the mean calculated frame rate.

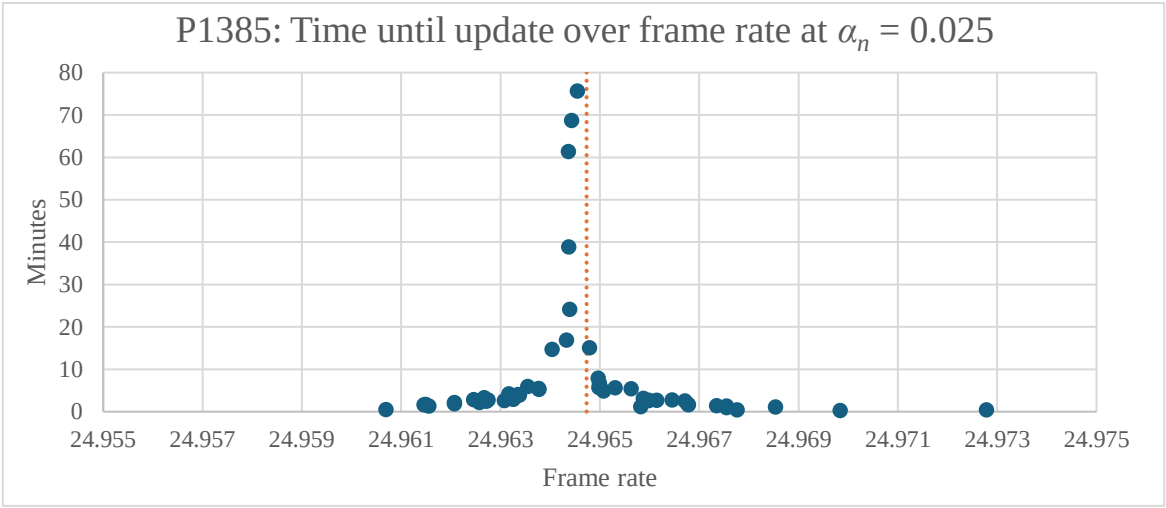

Figure 7.5: Shows the amount of time in minutes until update for each calculated frame rate at $\alpha_n = 0.025$ for the P1385. The dotted line represents the mean calculated frame rate.

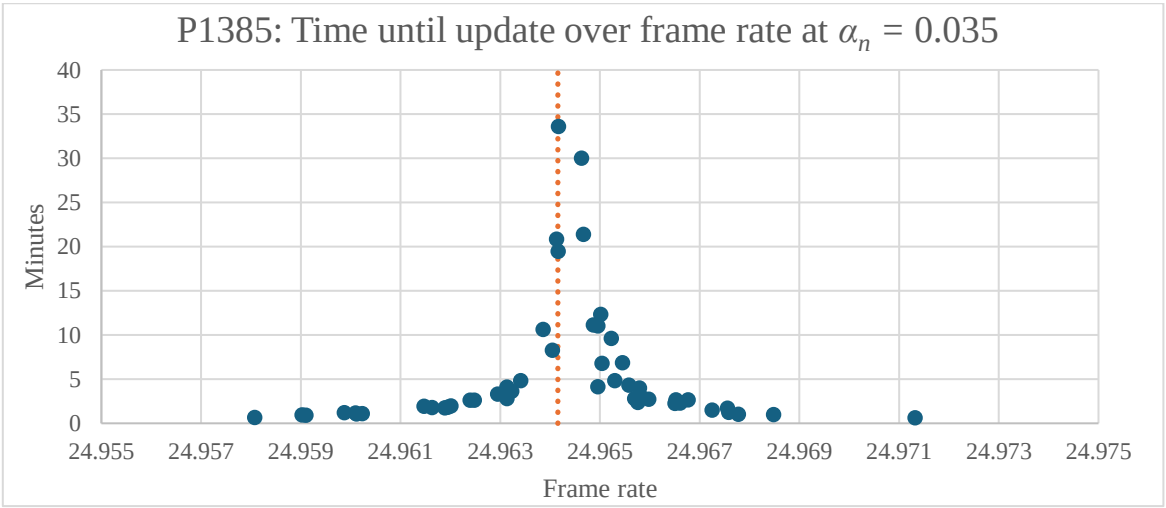

Figure 7.6: Shows the amount of time in minutes until update for each calculated frame rate at $\alpha_n = 0.035$ for the P1385. The dotted line represents the mean calculated frame rate.

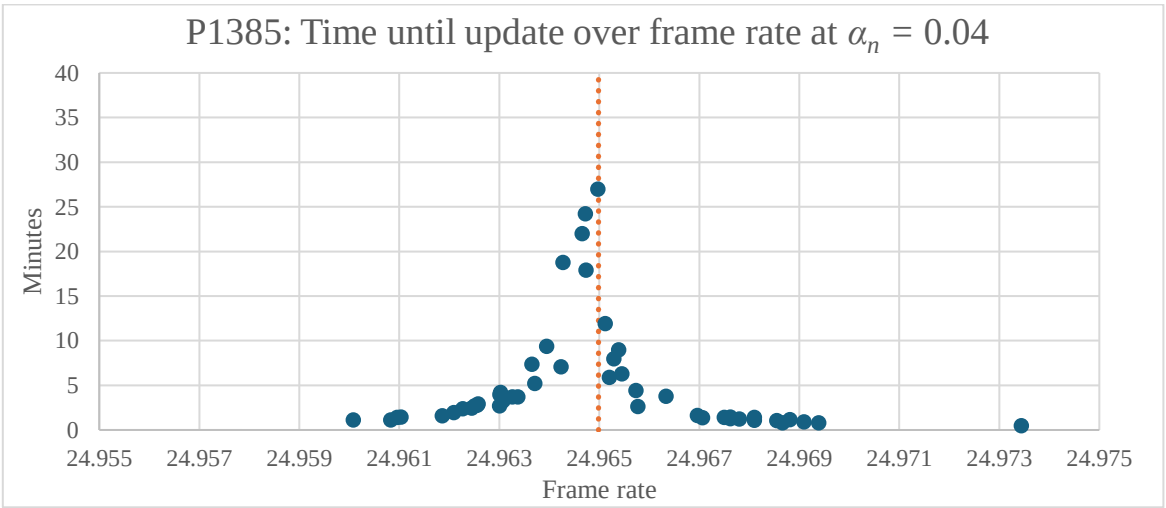

Figure 7.7: Shows the amount of time in minutes until update for each calculated frame rate at $\alpha_n = 0.04$ for the P1385. The dotted line represents the mean calculated frame rate.

7.3 Thesis Poster

Bachelor's Thesis

Temporal Artefact Mitigation in Rolling Shutter Videography

By: Marcus Remnélius Bou, Frans Wallinius

Supervisors: Philip Stjärneblad, Axis; Bernard Schmidt, LTH

1. Introduction

Rolling shutter cameras often suffer from distortions when capturing fast-moving objects or flashing light sources, such as LED or strobing IR-lights. These distortions occur due to the rolling shutter effect which is a result of the sensor reading the pixels in the image row by row. An issue was discovered that afflicted existing Axis cameras in the form of a band that move across the image due to strobing IR-lights. With the rise of global shutter cameras comes the use of strobing IR-illumination to increase image quality and reduce energy consumption.

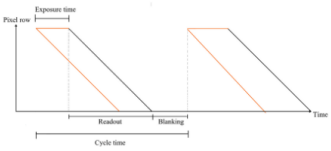

Figure 1: Exposure diagram of a rolling shutter camera.

Figure 2: An example of the strobe artefact.

2. Background

This issue was recognized by our supervisor and his department at Axis. A proof-of-concept solution was developed to verify the possibility of addressing the issue in software. They developed algorithms that detected the artefact, synchronized the camera's framerate to the strobe's frequency, and applied a delay to the exposure window.

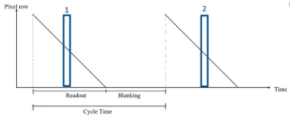

Figure 3: Exposure diagram of a rolling shutter camera with a stabilized artefact.

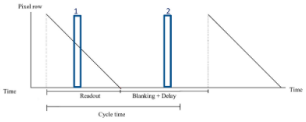

Figure 4: Exposure diagram of a rolling shutter camera with artefact eliminated.

3. Goals

- **Software Development:**
  - Improve the strobe artefact detection algorithm.
  - Develop a method to keep this artefact out of the frame after elimination.
- **Environmental Testing:**
  - Test the algorithms in both laboratory and real world environments to verify reliability and effectiveness.

4. Method

Our algorithm improved the existing solution in a number of ways. Firstly statistics was used to determine the background which to compare subsequent frames against. This led to a more reliable detection, with few false positives.

Secondly, we implemented adaptive thresholds based on the strobe artefact's peak intensity and the noise level in the image. This made our solution automatically adjust to lighting conditions without any manual input. In addition to these improvements, we've applied noise reducing filters to reduce the risk of detecting false positives even further.

Thirdly, the main improvement in our algorithm is a method called Maintain Delay. Here, the top and bottom row of the image is monitored for the reappearance of the artefact. When the artefact eventually reapperas due to a discretized framerate, a threshold is exceeded and the exposure is delayed again, resulting in a clear image.

Finally, to ensure reliability, we've implemented checks to detect changes in the background which update the appropriate values in order to adapt to environmental changes.

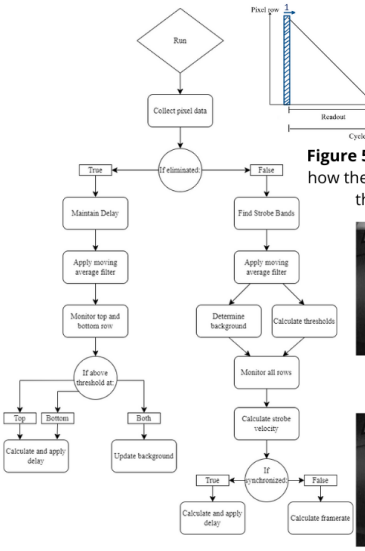

Figure 6: Flowchart visualizing our solution.

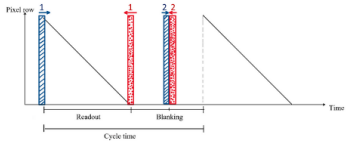

Figure 5: Exposure diagram visualizing how the Maintain Delay method keeps the artefact out of frame.

Figure 7: Strobe artefact present in the top of frame.

Figure 8: Delay applied.

5. Results

We have developed an algorithm that reliably detects a strobe artefact, even under challenging conditions. Our algorithm is sufficiently robust that any artefact intense enough to affect the forensic value of an image is detected. The second part of our software keeps the artefact out of frame at a success rate of 98% when used on an Axis P1385 camera.

6. Conclusion

A software solution has been produced that can be implemented on existing surveillance systems. This enables users of Axis cameras to keep employing their existing rolling shutter cameras while installing new global shutter cameras with their strobing IR-illumination. This will hopefully lead to reduced waste as existing systems can remain in service.

AXIS COMMUNICATIONS

Figure 7.8: Thesis poster.