RIGA TECHNICAL UNIVERSITY

Faculty of Computer Science, Information Technology and Energy

Institute of Applied Computer Systems

Mehmet Sahin Uces

Computer Systems

Student ID No 211ADB056

REAL TIME RGB CAMERA PARAMETERS TUNING FOR TREE SAPLINGS VISUAL ANALYSIS

BACHELOR THESIS

Scientific adviser Dr.sc.ing.

Andrejs Zujevs

RIGA 2024

RIGA TECHNICAL UNIVERSITY

FACULTY OF COMPUTER SCIENCE, INFORMATION TECHNOLOGY AND ENERGY

Institute of Applied Computer Systems

Work Performance and Assessment Sheet of the Bachelor Thesis

The author of the graduation thesis:

Student Mehmet Sahin Uces _____

(signature, date)

The graduation thesis has been approved for the defence:

Scientific adviser:

Dr. sc. ing. Andrejs Zujevs _____

(signature, date)

ABSTRACT

Keywords: Camera, RGB camera, White balance, Color constancy, Exposure

The implementation of automation in greenhouses has become an increasingly popular strategy to reduce manual labor and improve the quality of plants. Machine vision, which uses cameras, plays a crucial role in such automation solutions; however, the performance of cameras is constrained by limitations, which can negatively impact image quality. These limitations are primarily because of the inability of cameras to adapt to environmental changes. These environmental changes, such as sun light, affect the exposure and white balance of the image, lowering image quality.

The objective of this thesis is to develop a software prototype for real-time tuning of an RGB camera to achieve optimal exposure and color. The prototype was required to adapt to changing environmental conditions and operate in real-time.

The thesis explores the working principles of camera, describes the challenges related to environmental changes and discusses algorithms for implementation. For prototyping, a gain algorithm fully and a white balance algorithm partially implemented. Experiments show promising results, however further test should be conducted to refine and validate the prototype.

The volume of the thesis: 58 pages, 33 figures, 5 tables, 5 appendixes, and 68 references.

ANOTĀCIJA

Atslēgas vārdi: Ekspozīcija, Kamera, Baltā balanss, Krāsu noturība, Ekspozīcija

Automatizācijas procesu ieviešana siltumnīcās ir kļuvusi par arvien populārāku pieeju, lai samazinātu manuālo darbu un uzlabotu augu kvalitāti. Šādos automatizācijas risinājumos būtiska nozīme ir datorredzes sistēmām, kuras izmanto digitālās kameras. Tomēr kameru pielietojumam ir savu ierobežojumi, kuri var negatīvi ietekmēt attēla kvalitāti. Šie ierobežojumi galvenokārt ir saistīti ar kameru nespēju optimāli pielāgoties apgaismojuma un ārējās vides izmaiņām. Šīs izmaiņas, piemēram, saules gaisma, ietekmē attēla ekspozīciju un baltās krāsas balansu, samazinot iegūto attēlu kvalitāti.

Darba mērķis ir izstrādāt programmatūras prototipu RGB digitālās kameras parametru kontrolei reālajā laikā, lai panāktu optimālu ekspozīciju un attēla objektu krāsu. Prototips ļauj kamerai jāpielāgojas mainīgajiem ārējās vides apstākļiem, kontrolējot kameras parametrus reālajā laikā.

Darbā ir apskatīti digitālās kameras darbības principi, aprakstītas ar ārējās vides izmaiņām saistītās problēmas un aplūkotas to risināšanas pieejas. Prototipa izveidei pilnībā realizēts attēla gaišuma kontroles algoritms un daļēji realizēts baltās krāsas balansa algoritms. Praktiskie eksperimenti uzrādīja daudzsološus rezultātus, tomēr prototipa pilnveidošanai un validācijai nepieciešams veikt papildus testus.

Darba apjoms: 58 lappuses, 33 attēli, 5 tabulas, 5 pielikumi un 68 atsauces.

ACKNOWLEDGMENTS

This thesis was driven by an ongoing research project focused on the development of automated systems for monitoring the growth, health, and canopy control of nursery plants. The topic and objectives of my research were set in accordance with the goals of this project. I am deeply grateful to the project team and my supervisor for providing the camera and necessary software that were essential for implementing and testing the algorithms presented in this thesis. For more information about the project, please visit project website[1]

Special thanks to my friend David Melamed who gave his best effort to motivate me and help me to start writing my thesis. Without you, I would have been late on finishing my thesis. Also, I would like to thank all my friends and family who listened to my long annoying talks and problems during my thesis and motivated me.

[1] https://www.rtu.lv/lv/universitate/projekti/atvert?project_number=4767

Table of Contents

INTRODUCTION

The forestry industry plays a crucial role in providing a source of renewable energy through fuelwood and industrial roundwood, as well as further processed wood and paper products (European Commission. Eurostat, 2011). Forestry involves three phases: planning, harvesting and planting/maintaining the forest (Forteck Enviro, 2020). Planting the forest in the third phase is a lengthy process, taking years from planting to harvest. To make the process more efficient, it is possible to start growing trees in controlled environment such as greenhouses (Figure 0.1).

Figure 0.1 Example of tree seedling greenhouse (adopted from (Zujevs, 2023))

Greenhouses can provide a regulated water supply, a temperature-controlled environment, protection from external factors, artificial lighting, and more (Savic et al., 2022). Even though perfect conditions might be achievable, there are multiple factors that can affect the health of tree seedlings (Douglas, 2005). Greenhouse caretakers may need to monitor each factor individually to ensure that the seedlings are growing healthily, increasing the amount of human labor required. To reduce the required time and cut human labor, the industry is moving towards technological solutions and agricultural automation.

These solutions and automations can utilize digital cameras as a tool for data acquisition. The digital cameras must be tuned to acquire usable information, which is necessary due to environmental changes such as day-night cycles and changes in the

light conditions. These environmental changes affect the exposure and white balance of the image causing errors in the further processes.

Therefore, the objective of this thesis is **to develop a software prototype for real-time tuning of an RGB camera to achieve optimal exposure and color**. Real-time tuning enables a rapid response to changes in lighting conditions.

To achieve the goal, the tasks of the thesis are:

1. Describe the working principles and architectures of digital camera.
2. Describe the cause of the problem and challenges.
3. Explain potential algorithms and find optimal algorithm to implement in the software prototype.
4. Create a software prototype for real time RGB camera tuning.
5. Evaluate the performance of the proposed software prototype.

The structure of this thesis is as follows: Chapter 1 provides a description and working principles of digital cameras. Chapter 2 delves deeper into the state of the problem. Chapter 3 focuses on algorithms. Chapter 4 covers the development and implementation of the software. Chapter 5 includes experiment details and settings along with criteria. Chapter 6 presents the results analysis and finally conclusion.

1. DIGITAL CAMERA

This chapter explains the key concepts of digital camera and create a basic understanding of the working principles. Subchapter 1 includes history of camera from early stages to digital camera. Subchapter 2 delves deep into architecture of camera and subchapter 3 mentions the uses of camera in forestry and greenhouse.

1.1. Pinhole to digital

A camera is a lightproof object with a lens that captures incoming light and directs the light towards film or in the case of digital camera towards an imaging device (Bellis, 2016). The history of the camera, from its early stages to digitalization, spans from the 10th to the 21st century.

The concept of channeling light through a lens had been known for a long time, but it was only in the 12th century that this information became useful; the introduction of light-sensitive components was not until the end of the 17th century, and the technical knowledge and components necessary for the camera were not available until the beginning of the 19th century (Maître, 2017). This led to the lengthy development of the camera.

Following the article from Mary Bellis (2016), Ibn Al-Haytham, also known as Alhazen, invented the first camera obscura, specifically pinhole camera, around 1000CE. The pinhole camera was used to project the outside scene onto a surface, which could then be traced to 'capture' it (Figure 1.1).

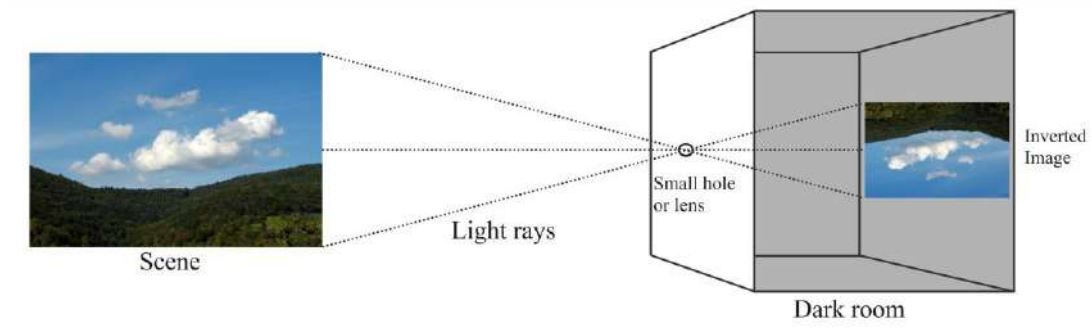


Figure 1.1 Example of camera obscura (Created by the author).

After the invention of the camera obscura, it took until the 19th century to capture the first photographic image. In 1827, Joseph Nicephore Niepce used a

camera obscura and a metal plate coated with bitumen. Niepce's method required eight hours of exposure, and the resulting image would fade with time. Later, Niepce formed a partnership with Louis Daguerre (Bellis, 2016). Daguerre developed a more convenient and effective method, which he named after himself as daguerreotype. This process created the first practical method of photography with a shorter exposure time and images that did not fade away. Daguerre utilized a sheet of silver-plated copper that was coated with iodine, resulting in a light-sensitive surface (Bellis, 2016).

Henry Fox Talbot is credited with inventing the first negative images, which could then be used to produce positive prints (Bellis, 2016). Talbot used paper coated with silver salt solution for his method. After perfecting this process, in 1841 he named it "calotype". In 1889, George Eastman invented photographic media on a flexible base, which gave birth to film and made it possible to produce box cameras, which can take up to 100 pictures (Bellis, 2016).

The charge-coupled device (CCD) was developed in 1934 with the intention of being used as a shift register memory store in a computer (Allen & Triantaphillidou, 2011). After the invention of CCD, In 1975, Steven J. Sasson and his colleagues in Kodak laboratories invented the first digital camera with CCD sensor that can take black and white images in 0.1 megapixel (Maître, 2017). Early issue with CCD was the resolution which caused by the pixel size and sensor area, but with the developments in computer technology resulted in smaller components which allowed to achieve higher resolution (Allen & Triantaphillidou, 2011).

Complementary Metal-Oxide-Semiconductor (CMOS) sensor, developed in 90's, became popular in 21st century due to the less cost of manufacture and power consumption compare to CCD (Allen & Triantaphillidou, 2011).

1.2. Working principles of digital camera

A digital camera is an optical device that captures an image by passing light through optics to the imaging sensor, converting the light information into voltage that can be stored (Nakamura, 2017).

1.2.1. Lens

The camera's lens is the optical component that functions as the camera's eye.

It controls the amount of light entering the camera and, through a series of optical elements, refracts and reflects the light onto a single (Canon, 2019).

Camera lenses typically consist of multiple lenses and are complex structures. To understand why complex lenses are necessary in modern cameras, we must first examine how the pinhole camera and the single lens camera work.

The most basic pinhole camera is constructed from a closed box with a small hole on one side, and the surface parallel to the hole serves as the image plane (Beyerer et al., 2016). The pinhole camera works by allowing a single beam of light from a point in the scene to pass through a pinhole and onto the image plane. This process helps to channel the light and create an inverted image. Because the single light beam can only enter the pinhole in one way, the resulting image is focused on the entire scene (Nayar, 2021c). However, this also means that the amount of light received on the image plane is limited, requiring longer exposure times.

The Figure 1.2 demonstrates how a pinhole camera projects a 3D point onto its image plane.

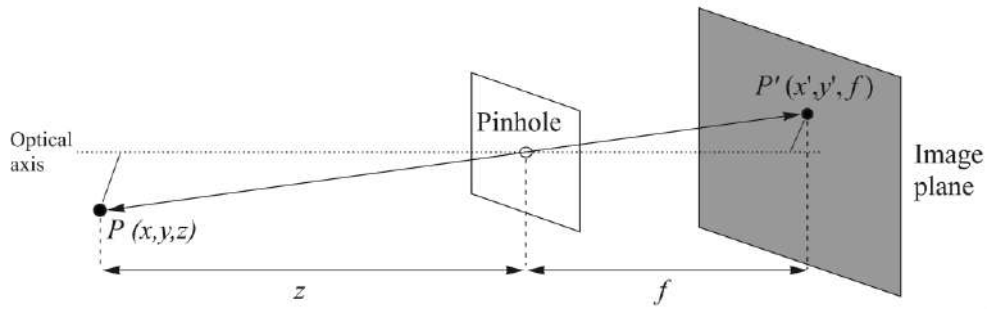


Figure 1.2 Projection in pinhole camera (adapted from (Nayar, 2021c))

The formula for finding the projection mapping from 3D point coordinates (x, y, z) to 2D image coordinates is as follows: (assuming the pinhole is the origin point) (Hartley & Zisserman, 2018):

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} fx/z \\ fy/z \\ f \end{pmatrix}, \tag{1.1}$$

where f – distance between image plane and pinhole (focal length).

Using a lens instead of a pinhole enables the convergence of all light rays from a point in the scene onto the image plane. Increasing the amount of light can reduce the exposure time, but using a lens can capture all available light, resulting in a lack of focus throughout the image (Maître, 2017). As the Figure 1.3 illustrates, a

single lens can only focus on one plane, causing blurring on the image when light from other planes is present.(Nayar, 2021b)

The aperture, which is the size of the opening in the lens, controls the amount of light that enters the system (Barry, 2016). The larger the aperture, the more light passes through the lens, and the smaller the aperture, the less light hits the lens (Maître, 2017).

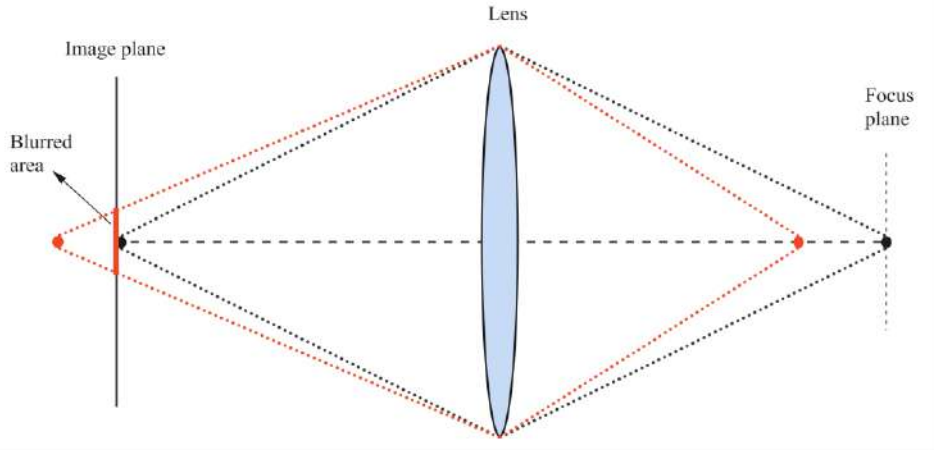


Figure 1.3 Focus problem with single lens (adapted from (Nayar, 2021b))

When the aperture is smaller, like the size of a pinhole, the resulting image will have less blur, achieving a larger depth of field (Nayar, 2021a). However, this comes at the expense of a darker image.

The f-number is the ratio between the aperture diameter and the focal length (Allen & Triantaphillidou, 2011). It is also used to refer to the same concept as the aperture. The f-number is denoted as *f/1*, *f/1.4*, or similar, following ratios of $\sqrt{2}$ dividing the energy received by 2 at each step (Maître, 2017).

The concept of depth of field refers to a lens' capacity to maintain image quality without requiring refocusing, even if the object is moved closer or further away from the focus plane (Edmund Optics, 2021). Aperture has a direct effect on the range of depth of field.

Modern camera lenses are typically composed of a group of lenses, known as a 'centered system', rather than a single lens (Maître, 2017). The use of a centered system can improve image quality by reducing the number of defects and aberrations caused by individual lenses (Maître, 2017).

1.2.2. Image sensor

An image sensor, also known as an imager, is a semiconductor device that captures an image focused by the lens (Allen & Triantaphillidou, 2011; Nakamura, 2017). Different image sensors can detect different wavelengths of light, from X-rays to infrared, depending on their structure and materials (Nakamura, 2017).

The amount of detail that an image sensor provides is called resolution and is measured in pixels (Mancuso, 2001). Image sensors contain millions of pixels arranged like a grind with a space in between them. Each pixel contains a light-sensitive element, known as the photodiode, and other components (Figure 1.4).

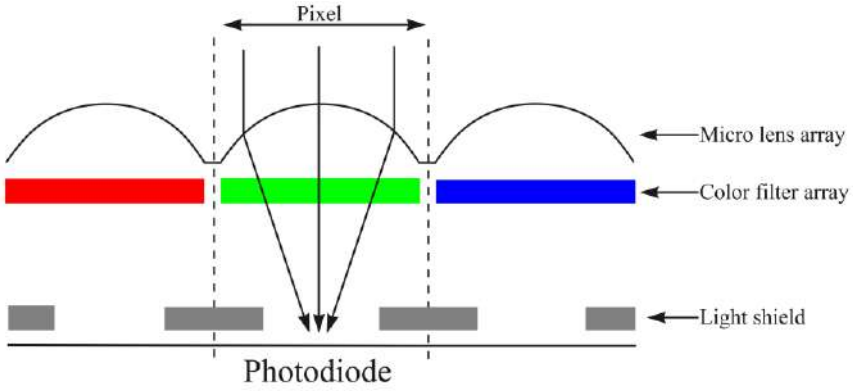


Figure 1.4 Simple pixel structure (adapted from (Nakamura, 2017))

An important characteristic of a pixel is its fill factor, which represents the ratio of its photosensitive area (Nakamura, 2017).

To improve the fill factor, a microlens array can be placed on the top layer of a pixel to focus light on the photosensitive area and increase the fill factor by 30-70% in some image sensors (Allen & Triantaphillidou, 2011).

Photodiodes are monochrome, which means that they can only measure the intensity of light. To be able to collect color information, a layer of a color filter array can be added to the pixel. The Bayer filter, which is commonly used in digital cameras, is composed of macro pixels (Beyerer et al., 2016). Each macro pixel consists of two green pixels, one blue pixel, and one red pixel as seen in Figure 1.5. However, the effective resolution is reduced due to color filter arrays because each pixel is now responsible for only one color (Allen & Triantaphillidou, 2011).

To obtain the other two colors in each pixel, a process called demosaicing, also known as color interpolation, must be performed (Beyerer et al., 2016). This process uses neighboring pixels to estimate the missing color channels.

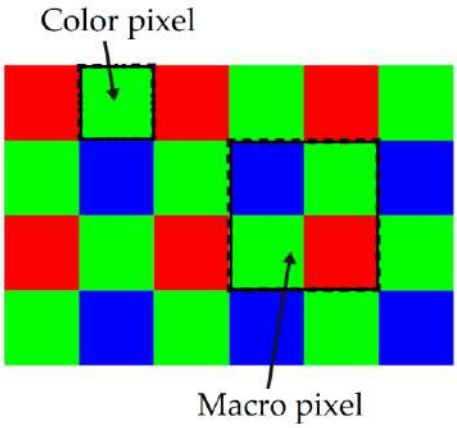


Figure 1.5 Bayer filter arrangement and macro pixel (adapted from (Beyerer et al. 2016))

The most common image sensors today are charge-coupled devices (CCD) and complementary metal-oxide-semiconductor (CMOS). Both sensors are designed in a similar way using semiconductor technology, with the key difference between CCD and CMOS being how signals are read from the pixels (Litwiller, 2001).

Beyerer et al. (2016) describe the following process of charge read in CCD sensor (Figure 1.6): Once the sensor has accumulated charges on the photodiodes, the charges of all pixels are shifted to the vertical shift registers.

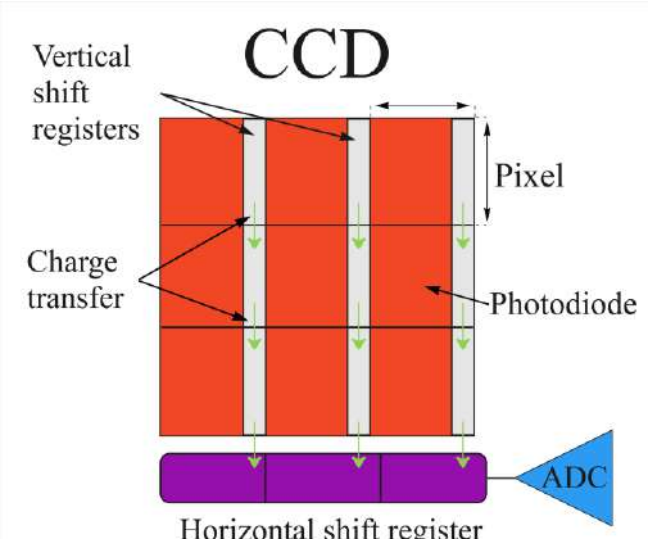


Figure 1.6 Simplified CCD sensor (Created by the author)

These charges are then moved, line by line, into a horizontal shift register at the bottom. The horizontal register sends the charges, one at a time, for amplification and analog to digital conversion. This process repeats until the entire sensor has been read out.

The described readout method is called ‘interline-transfer’, while CCD has other methods, such as ‘full frame’. The main difference on the methods is that charges are either shifted directly or first sent to a the shift registers (Beyerer et al., 2016).

The CMOS sensor differs at the pixel level, with each pixel containing an amplifier, reset, and readout transistor (Figure 1.7).

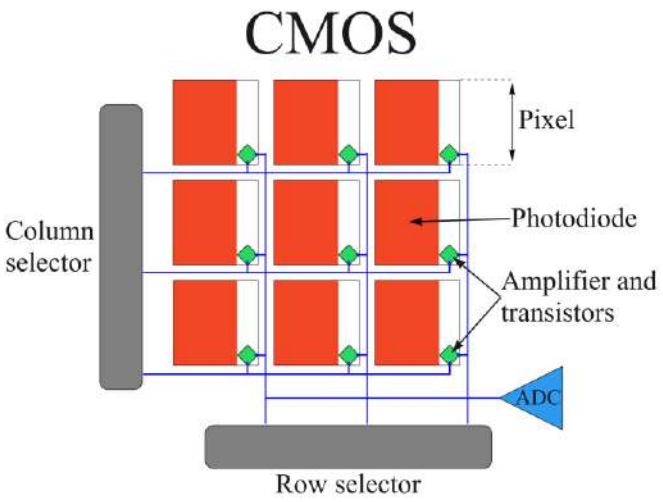


Figure 1.7 Simplified CMOS sensor (Created by the author)

With the row and column selectors, each pixel can be read individually. This also makes it possible to partially read out the image sensor for certain applications. The presence of transistors in each pixel takes up space, decreasing the fill factor of the sensor (Allen & Triantaphillidou, 2011). In contrast to CCD, the use of multiple amplifiers introduces noise due to manufacturing processes (Mancuso, 2001).

Whether on the pixel in CMOS or common in CCD, the amplifier's job is to boost the signal coming from the photodiodes. Amplification is necessary because the voltage that forms in the photodiode is in the range of one quadrillionth of a volt (Allen & Triantaphillidou, 2011).

1.2.3. Image pipeline

The image pipeline is the process that starts with light collection and ends with

image storage or display. Different camera manufacturers use varying image pipelines, which may differ in the order of functions or the functions themselves (Allen & Triantaphillidou, 2011). Since the previous subsection on image sensors already covers the collection of light, this subsection will begin from that point.

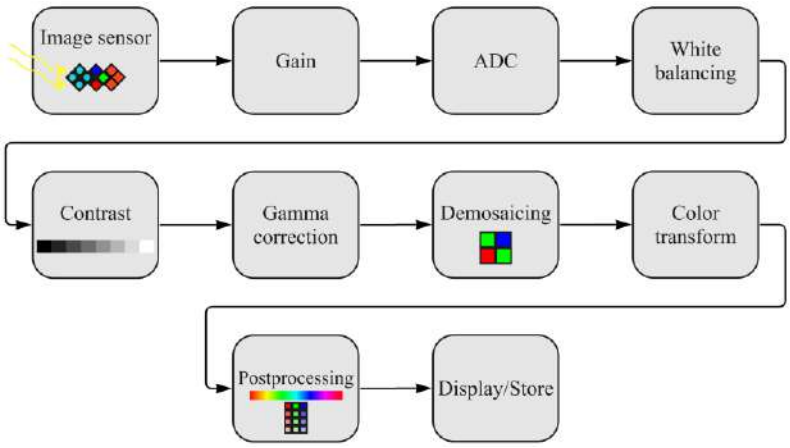
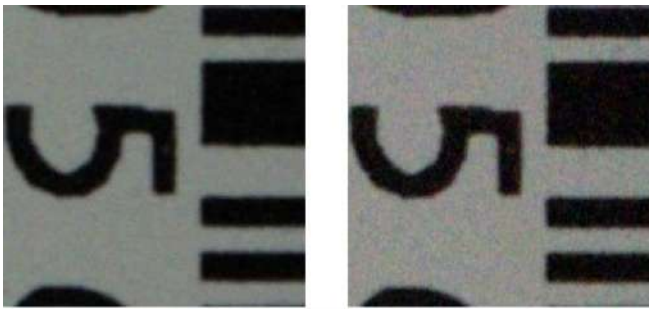


Figure 1.8 Simplified image pipeline (Created by the author)

As seen in the Figure 1.8, gain control comes after the image sensor. Gain control refers to the ability to adjust the amplification scale of a signal (Edmund Optics, 2015b). There are two ways to control gain: directly in the image sensor amplifier or after the ADC, known as 'digital gain'. As gain amplifies the signal, its effects can result in higher image noise.

In the Figure 1.9, noise can be seen as incorrectly colored pixels in white areas and grey pixels in black areas. If the gain is set too high, some areas of the image may become excessively bright and appear completely white. Vice versa, if the gain is set too low, the image may become very dark and appear black.



a) b)

Figure 1.9 Gain effect on noise level: a) higher gain; b) lower gain (Created by the author)

Either situation will result in a loss of detail that might have been useful, so controlling the gain is critical for further processing.

Depending on the pipeline order, before or after gain, signal has to be converted into digital from analog. This process occurs in the analog-to-digital converter (ADC), which scales and converts the input signal. The scaling of the input signal depends on the resolution of the ADC, which refers to the number of bits it can handle (Allen & Triantaphillidou, 2011). Because of the resolution range and number of values for digital signal, errors are introduced into the system with ADC.

The digitized signal can be in various formats called "pixel formats". The pixel formats are named based on the arrangement of the image data.

'Mono' format, short for monochrome, stores the brightness of the pixels. In contrast, the Bayer pixel format, such as 'BayerRG', stores the data in a Bayer filter pattern. The format uses 'RG' at the end to indicate the color of the first row of the macro pixel. For color pixel formats like 'RGB', all color values for each pixel are included. The order of the letters in the name indicates the order of the color data (IDS Imaging Development Systems GmbH, n.d.).

In addition to the names, each pixel format has a number, for example Mono8 and BayerRG12. These numbers indicate how many bits are dedicated to each pixel or color value (in the case of a color pixel format) (IDS Imaging Development Systems GmbH, n.d.). The choice of pixel format type is determined by the color filter layer in use, and the bit numbers are constrained by the ADC.

The next step in the pipeline is white balancing. The purpose of white balancing is to ensure that white objects remain white even under different colored light sources. Incorrect white balance can cause the image to have a warmer (yellow tint) or colder (blue tint) appearance (Figure 1.10) (Zapryanov et al., 2012).

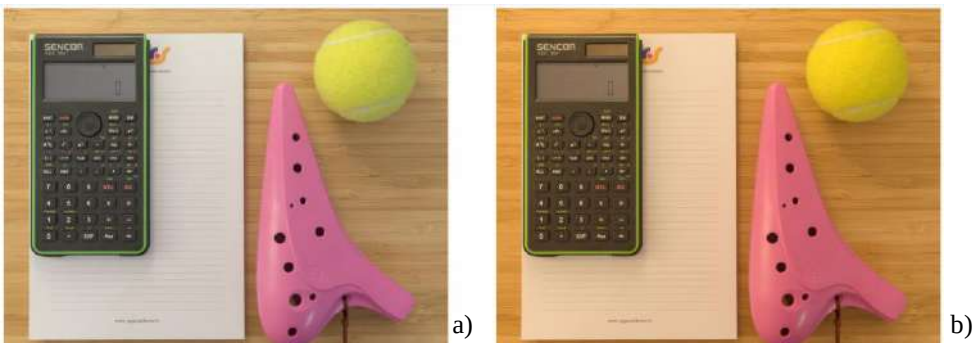


Figure 1.10 Effect of white balance: a) correct white balance; b) incorrect white balance (Created by the author)

White balance can be achieved by adjusting the gain levels of each color channel, red, green, and blue, or by utilizing digital signal processing algorithms.

Contrast refers to the ability to distinguish between blacks and whites. The tones of gray in between should be well-defined and gradual, while black details should remain black and white details should remain white (Edmund Optics, 2015a). There are multiple techniques to change contrasts, including histogram modification, such as histogram equalization, which is widely used (Kotkar & Gharde, 2013).

Image sensors have nonlinear characteristics when converting light into digital signals. These nonlinearities can be represented by a simple function called a gamma curve, and the process of compensating for this curve is known as 'gamma correction' (Nakamura, 2017). Gamma correction adjusts the intensity of all channels and transforms them into a nonlinear signal (Hornberg, 2017).

Next comes demosaicing, which is the process of generating missing color information due to the color filter array, as already mentioned in section 1.2.2. One common demosaicing algorithm is bilinear interpolation. It works by calculating missing colors from the neighboring pixels by averaging their values (Fuentes et al., 2009).

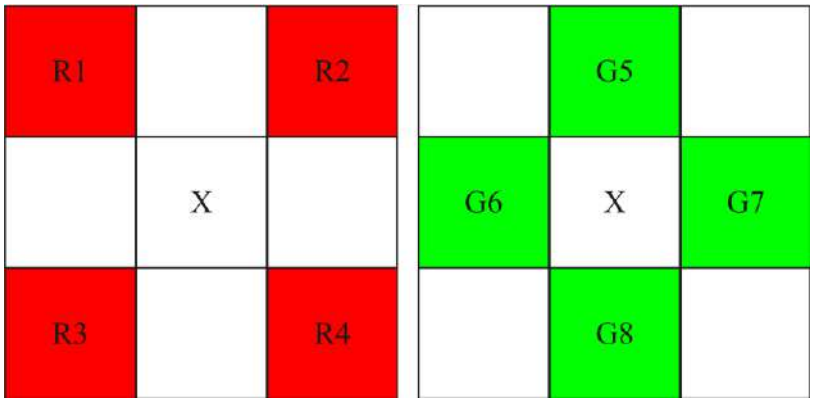


Figure 1.11 Bilinear interpolation (Created by the author)

For the pixel *x* in the Figure 1.11, the red value is calculated by $R_x = (R_1 + R_2 + R_3 + R_4)/4$ and the green value is $G_x = (G_5 + G_6 + G_7 + G_8)/4$.

Color transformation refers to the change in the color space. A color space is a coordinate system that represents colors (Hornberg, 2017). Different color spaces cater to different needs, such as displaying, printing, and image processing. Most computer-based applications use the RGB (red, green, and blue) color space, which employs primary colors, and HSL (hue, saturation, and lightness), which is closer to

human perception (Hornberg, 2017). Color transformation can occur at different stages in the pipeline and may occur multiple times.

Depending on the use case of the imaging system, post-processing can include various procedures. In this case, it's sufficient to focus on sharpening, noise reduction, hue, and saturation for the sake of simplicity.

Due to their optical limitations, the lenses and processes in the image pipeline, including demosaicing, can cause blur in the image (Allen & Triantaphillidou, 2011). Sharpening is commonly achieved through high boost filtering. This process enhances the edges in an image, making them stronger, and then adds them back to a slightly amplified version of the original image. As a result, the picture becomes sharper, and the details become more distinct.

As mentioned in previous chapters, there are multiple causes of noise. Although noise reduction before demosaicing is preferred to reduce incorrect colors, it can also be done before or after (Allen & Triantaphillidou, 2011). In addition to the silicone itself and the ADC, dark current is also a source of noise. Dark current is caused by thermal effects and is related to the temperature of the sensor, independent of the signal (Maître, 2017). Noise reduction algorithms aim to correct these sources of noise.

The term 'hue' refers to the property that allows a color to be classified as red, yellow, green, blue, or a combination of these (Allen & Triantaphillidou, 2011). Changing the hue of an image results in shifting all of its colors. Saturation refers to the intensity of a color in relation to its own brightness (Allen & Triantaphillidou, 2011). Increasing saturation intensifies all colors, while decreasing it causes colors to shift towards grayscale. Adjusting hue and saturation is important for the final image and can result in an artistic or realistic color image.

At the end of the image pipeline, the resulting image can be compressed by converting it to different formats or left in RAW format for storage or direct display.

1.3. Camera in forestry and greenhouse

In the intelligent systems of forestry, greenhouses and plant factories, use of machine vision, to achieve automation, became popular (Tian et al., 2022). To help decision-making and train these systems, there is a huge dependencies on photographic information (Mosslah & Abbas, 2023). Currently, the industry uses

various types of cameras, including color, infrared, and 3D cameras, to collect these datasets.

Satellite imagery, or satellite remote sensing, has been used in forestry since the launch of the first earth observatory satellite, Landsat-1. Satellite imagery is the use of images acquired from space for surface analysis, and It has been proven to be effective in the decision-making process in forestry (Holmgren & Thuresson, 1998). Satellite imagery has been used for various purposes, including land classification, ecological analysis, damage monitoring, forest operation monitoring, and forest regeneration monitoring (Holmgren & Thuresson, 1998).

Another use of cameras in forestry is through unmanned aerial vehicles (UAVs). With the use of camera equipped UAVs, it is possible to collect imagery to reconstruct a 3D structure from overlapping 2D photographs (Dash et al., 2016). These structures can then be utilized to produce digital terrain models and digital surface models to be processed for decision-making (Dash et al., 2016).

The use of cameras in greenhouses and plant factories can provide more benefits than forestry due to the smaller area and smaller distance to the plant.

Some of the use cases in plant factories and greenhouse follows as (Tian et al., 2022):

- estimating the health and growth stage of a plant by capturing images of its color, size, texture, and organs such as flowers, leaves, stems, and fruits;
- disease monitoring using image processing and segmentation based on plant surface characteristics;
- pest control method that relies on machine vision and trained models to identify pests based on their shape, color, and texture;
- determining nutrition levels based on texture and color characteristics, such as hue and saturation combined with the use of machine learning models;
- maturing and quality level of the fruit depending on the growth characteristics and color values.

The use of cameras in forestry and greenhouses is further supported by the examples and methods presented, given the importance of image acquisition in automation and machine vision systems.

2. STATE OF THE PROBLEM

In this chapter, the problem is discussed in details and the factors to achieve the goal of the thesis are explained in.

2.1. Background and objective

This thesis is being developed as part of a bigger project which aims to apply machine vision for assessment of tree seedling using cameras in a greenhouse setting. The goal of the thesis resolves around the conditions in the greenhouse. Sun light combined with roofing material of the greenhouse; high humidity levels creating fog and lower the visibility and day night cycles causing negative effect to the machine vision application. Therefore, the goal of the thesis is to develop a software prototype to control and adjust gain and white balance of an RGB camera in real time, aiming to achieve optimal exposure and color images in greenhouse.

Tasks of this thesis are follows as:

- provide overview of the working principles and architectures of digital camera, which has been done in chapter 1,
- describe the goal of the thesis in detail, chapter 2,
- list possible solutions/algorithms,
- choose one of the solutions to achieve the goal and justify it,
- implement the solution to the RGB camera,
- provide analysis of results of the performance of the chosen solution.

Project is already a work in progress, development of necessary software and hardware being carried out. There is a “camera box”, Figure 2.1, for the project housing an RGB camera, a multispectral camera, and a stereo camera.

Cameras are connected to a computing board located inside the camera box. The computing board is a Nvidia Jetson, running a Linux-based operating system. Communication with the system and data transfer is made by an ethernet cable that is also a power cable that can supply necessary power to the system.

Top camera, RGB camera is the camera that this thesis aims to develop a prototype software. The camera is “Avium 1800 U-1620” from Allied Vision. It has a 16.2-megapixel CMOS sensor. The lens in use does not allow electronic aperture

control means that it as to be set manually. Target frame rate to achieve is 40 frames per second. This makes the exposure time minimum 25ms.

Figure 2.1 Actual camera box used in the project (Created by the author)

2.2. Exposure

Exposure in photography is the amount of light that reaches the image sensor and the camera's exposure is determined by a combination of factors that affect the brightness or darkness of the resulting image (Allen & Triantaphillidou, 2011). These factors include available light, aperture, exposure time, and gain.

In a controlled environment, manipulating the available light is arguably the best way to influence exposure. By adjusting the lighting, one can set other parameters to achieve the desired look in the resulting image. When a controlled environment is not achievable, or lighting is insufficient, other parameters become more important.

As stated in subsection 1.2.1, aperture refers to the opening in the lens and is directly correlated with exposure. Modern lenses allow us to adjust the aperture size and control it. Different aperture sizes have different use cases. For example, a smaller diameter results in a sharper image, but the image will be darker.

Exposure time refers to the duration for which the image sensor collects light. It can range from as high as 30 seconds to as low as 1/4000 of a second. The brightness of an image increases with longer exposure time. However, the appropriate exposure time should be selected based on the object of interest, as longer exposure times can cause blurring of moving objects (Maître, 2017).

Gain refers to the amplification level of the signal, as explained in subsection 1.2.3. Sometimes, gain is confused with ISO. In analog cameras, ISO refers to the sensitivity of the film. In digital cameras, ISO is used as a signal boost level, like gain. The main difference between the two is that gain is measured in dB, while ISO has its own measurement scale ranging from 100 to 6400 or even higher (“Gain and ISO,” 2022).

In this prototyping's setting, it is not possible to control the light as the greenhouse is an outdoor environment. The current camera lens is configured for optimal focus and aperture control, and software control is not available. The prototype's objective is to achieve optimal exposure in a stream of images, and exposure times are linked to the desired frame rate. Therefore, for this case, the only way to control exposure is through gain.

2.3. Color accuracy

Color accuracy can be defined as the ability of an imager to accurately reproduce the true colors of the subject matter. It is important, especially when considering the numerous machine vision applications in greenhouses that rely on an RGB cameras (de Luna et al., 2018; Guoxiang et al., 2016; Kiratiratanapruk & Sinthupinyo, 2011). Some parts of the camera or the image pipeline can introduce color accuracy problems, such as sensor spectral sensitivity, color filter array, gain, ADC, and post-processing.

Spectral sensitivity refers to a system's ability to detect various wavelengths of light. However, image sensors do not have equal sensitivity to all wavelengths due to the materials they are made of (Allen & Triantaphillidou, 2011). They typically have lower sensitivity to blue light and higher sensitivity towards red and infrared (Allen & Triantaphillidou, 2011).

As mentioned in subsection 1.2.2, color filter arrays (CFA) are a layer in the image sensor. Different CFAs have varying patterns and partitions for each color, allowing for the collection of color information in one channel while limiting the other colors within a pixel (Maître, 2017). This diversity, along with the demosaicing required with CFAs, introduces color errors.

Since pixels are responsible for one color channel with CFA, gain and ADC have an effect on how these wavelengths of light are converted into digital signals.

ADC directly affects the color bit depth of an image, defining the number of different colors (Allen & Triantaphillidou, 2011).

Incorrectly adjusting hue and saturation during post-processing can significantly impact the final color of an image. It is important to use accurate hue and saturation settings to avoid altering the colors and their intensities of the image.

2.3.1. Color constancy

Barron (2015) describes the two factors that determine the color of a pixel: the actual color of the object and the color of the illuminant light (light source) reflecting off the object. Humans can naturally differentiate and perceive the color of objects under varying light conditions. However, this task is more challenging for computers, "given a yellow pixel, how can one discern if it is a white object under a yellow illuminant, or a yellow object under a white illuminant?" (Barron, 2015). Tackling this problem is referred to as "color constancy".

Color constancy is the phenomenon where the color of a surface appears to remain constant despite changes in the illuminant (Foster, 2011). Replicating this in digital cameras is referred to as "computational color constancy".

To understand why this is a problem, how image is formed in camera has to be studied. An image represented with *p* can be formulized as (Bianco et al., 2012):

$$p = \int I(\lambda)S(\lambda)C(\lambda)d\lambda \tag{2.1}$$

where $I(\lambda)$ – is illuminant spectral power distribution,
$S(\lambda)$ – is surface spectral reflectance,
$C(\lambda)$ – is sensor spectral sensitivity.

In color constancy, aim is to estimate the illuminant spectral power distribution $I(\lambda)$.

Throughout the history of computational color constancy, multiple methods have been developed across different approaches. Some of the approaches include using a single image, multiple images, different devices, and video feeds (Gijsenij et al., 2011). There are numerous studies and research projects on algorithms for color constancy and each algorithm has its own use cases and requirements (Barnard, Cardei, et al., 2002; Barron, 2015; Barron & Tsai, 2017; Gijsenij et al., 2011; Zapryanov et al., 2012).

Because of the importance of color accuracy in machine vision applications in greenhouses, color constancy algorithms can be used in this area to achieve the required colors. Moving on in the thesis, algorithms related to increasing color accuracy must be studied and selected, taking into consideration real-time use and optimal performance under greenhouse conditions.

2.3.2. White balancing

Similar to computational color constancy, white balance also aims to achieve color constancy, but in a simpler manner. As the name suggests, white balancing ensures that the white point in the scene appears white in the image, regardless of the color of the illuminating light (Allen & Triantaphillidou, 2011). White balance can be set manually, customarily, or automatically.

Manual settings can be adjusted in two different ways: In consumer cameras, you can select a preset for the scene's illumination (such as daylight, cloudy, or tungsten), or you can manually set the color temperature (Maître, 2017).

Custom white balance, also known as color calibration, can be achieved by using a known white or gray spot in the scene or a color test pattern placed in the scene (Allen & Triantaphillidou, 2011). This method allows for the separation of illuminant light and is the most accurate way to achieve white balance (Maître, 2017). However, it requires prior knowledge of the scene.

Automatic white balance uses algorithms to determine the best setting. Multiple algorithms operate in different ways and require computational power. Additionally, each camera company has its own proprietary algorithm.

One of the algorithms is ‘gray world’. The gray world assumption states that all color values in an image will average to gray and corrects the colors according to that (Ramanath et al., 2005). Another algorithm is called 'white patch'. This algorithm assumes that the brightest color in the image is white and then calculates the necessary correction (Zapryanov et al., 2012).

2.4. Related works

Although the author of this paper could not find specific papers related to the

acquisition of real-time color and exposure accurate images in greenhouse conditions, papers on machine vision applications in greenhouses and color constancy, white balance and exposure algorithms can be studied individually.

In the paper “Image segmentation algorithm for greenhouse cucumber canopy under various natural lighting conditions” (Guoxiang et al., 2016), the author's primary objective is to apply a segmentation algorithm to distinguish the plant from the background, while also addressing the issue of shadows caused by factors such as the greenhouse upper beam. To enhance the accuracy of the segmentation and eliminate the mentioned shadow problem, the author used the image enhancement algorithm "multi-scale retinex" and stated that the resulting images were improved.

While the study (de Luna et al., 2018) focuses on disease detection in tomato plants, it presents a promising experimental environment. However, it lacks any mention of camera parameters and color constancy. It is assumed that fixed or automatic settings were used in image capture, which is not suitable for the purpose of this thesis.

The papers (H. Li et al., 2023), (Hassankhani & Navid, 2012) and (Kiratiratanapruk & Sinthupinyo, 2011) uses color to sort or classify. Although the Papers mention white balance, they achieve this through a controlled environment, such as using a 'capturing box'. These boxes consist of one or more light sources to illuminate the object and eliminate shadows. Light sources and surface colors can be arranged to separate the object from the background or highlight specific characteristics of the object of interest, thus not requiring complicated algorithms to achieve color accuracy.

Papers like (Bernacki, 2020) mentions the light meter, a sensor that measures the intensity of light in the scene, and assumes that the camera either has a built-in sensor or an additional sensor is in use. This case is applicable to photography cameras since they come with this sensor but the industrial cameras usually does not include these sensors.

The paper (Zhang et al., 2019) proposes a method that uses dual illumination estimation addressing both under and over exposed areas of an image, but the proposed method is used for changing the images after acquisition. Paper also mentions that the algorithm runs at ~0.8s which meant it is not suitable for real-time use.

Gradient based exposure controls also exists (Kim et al., 2018; Shim et al., 2014) but because of their complexity they will not be considered in this thesis. Due to most of the algorithms that are in use in cameras today are locked behind the manufacturer and the recent public developments are not that different from the previous publishes, this thesis will consider more simple and analytical approaches in the algorithms section.

Recent developments of white balance estimation are revolves around the use of machine learning (Afifi et al., 2019; C. Li et al., 2023). Due to use of machine learning is not applicable for the purpose of this thesis, this thesis will be considering the older algorithms that can be used in real-time operation.

In the “Computational Color Constancy: Survey and Experiments” (Gijsenij et al., 2011) multiple computational color constancy algorithms has been talked and evaluated. The authors only focus on algorithms that use a single image. The paper presents 16 methods and compares them on a couple of image datasets. The authors suggest that combining several approaches can lead to better results in estimating the illuminant for different images. The article (Zapryanov et al., 2012) compares of the basic white balance algorithms. It includes methods derived from gray world and retinex theorem. Some of the more detailed comparisons can be find in (Barnard, Cardei, et al., 2002) and (Barnard, Martin, et al., 2002). Part one of the paper measures algorithm performance using synthesized data, while part two tests are conducted with a large set of calibrated images acquired from a camera.

3. ALGORITHMS

Variant approaches are available for both gain control and white balancing. Due to platform constraints, some algorithms may not be applicable such as algorithms that requires datasets and machine learning. This chapter describes the concepts used in the algorithms and analyzes the algorithms themselves.

3.1. Background concepts

This subchapter describes concepts that are in use in the algorithms that will be discussed.

3.1.1. Histogram

A Histogram is a graph that represents the values to their frequency of occurring. It is made of so-called bins and each bin represents value/level. In the case of a color image, a histogram is calculated for each color channel. Histogram $h(f)$ can be denoted as follows (Allen & Triantaphillidou, 2011):

$$\boldsymbol{h(f_k) = n_k} \tag{3.1}$$

where f_k – is the kth color level value (depending on the channel);

n_k – number of pixels.

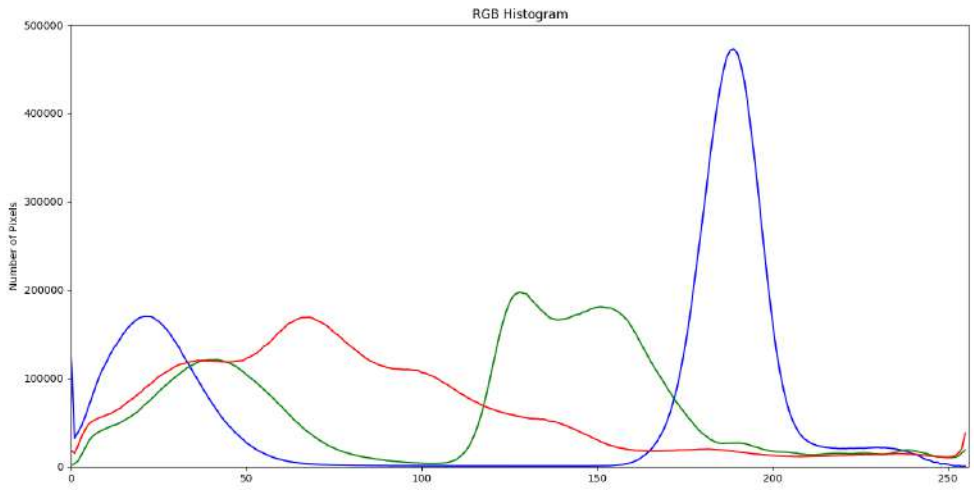


Figure 3.1 Example of an RGB image histogram (Created by the author)

As seen in the Figure 3.1, each color channel represented by its own color. The x-axis represents the range of the color value, and the y-axis represents the total number of pixels in that color value.

By examining the histogram, one can determine if the image is overexposed or underexposed. If the number of pixels is high towards the right edge of the histogram, the image is considered overexposed. Conversely, if there are a high number of pixels towards the left edge of the histogram, the image is considered underexposed (Figure 3.2) .

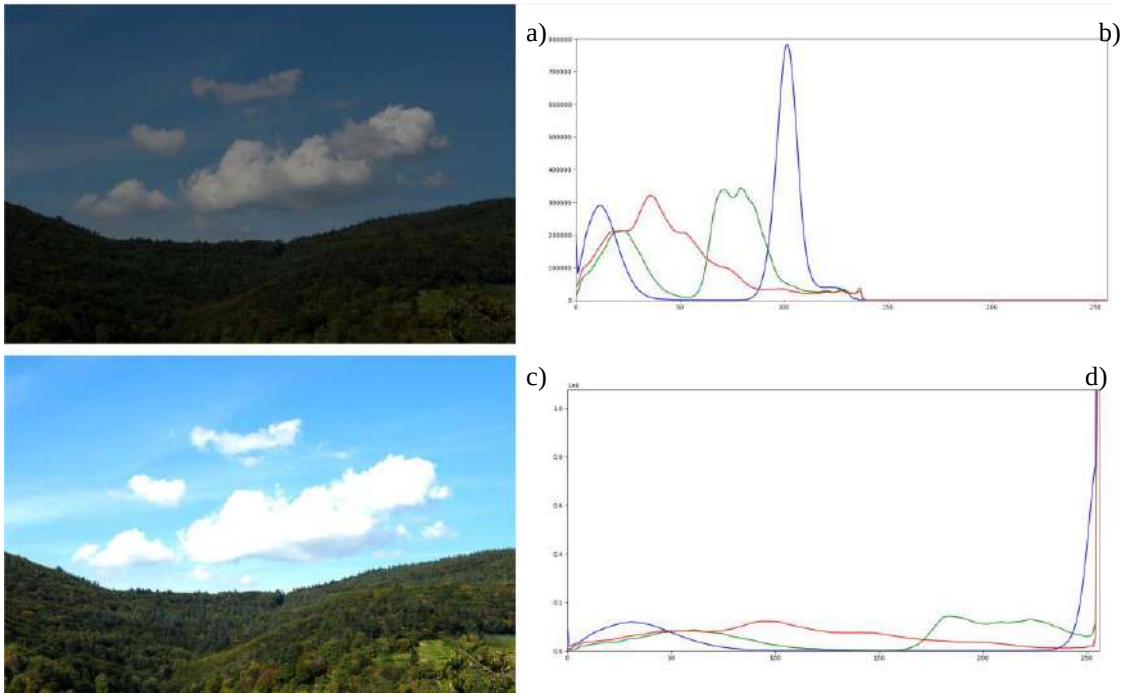


Figure 3.2 Affect of exposure on histogram (Created by the author)

The cumulative frequency histogram, or cumulative histogram in short, calculated based on the histogram of the image and is used in further assessment of the image (Figure 3.3) (Burger & Burge, 2022).

The cumulative histogram H can be calculated for a histogram h in following equation (Burger & Burge, 2022):

$$\boldsymbol{H(i)} = \sum_{\boldsymbol{j=0}}^{\boldsymbol{i}} \boldsymbol{h(j)} \qquad \text{for } 0 \leq i < \text{K}, \tag{3.2}$$

where K – size of histogram.

In a simple way, in cumulative histogram for a value *i*, the *H(i)* is equals to the number of pixels on *i* plus sum of all pixels between 0 and *i*.

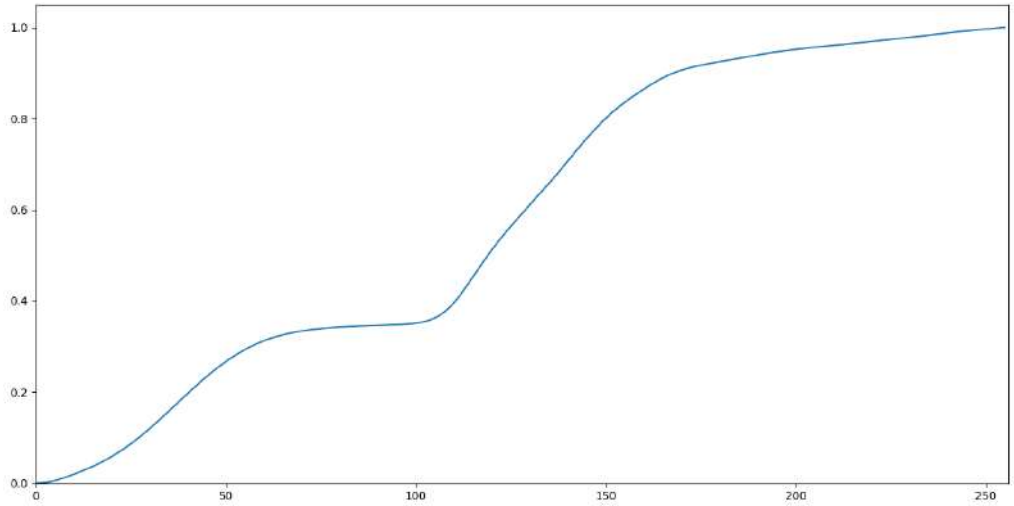


Figure 3.3 Example of cumulative histogram (Created by the author)

Both histograms and cumulative histograms are commonly used in image assessment and enhancement, as well as in some gain control algorithms. Understanding histograms is therefore often helpful in understanding gain control algorithms.

3.1.2. PID controller

The PID controller is an industrial control system that calculates the necessary adjustments to drive the input value towards the reference value (Figure 3.4). The name of the PID comes from the term of the individual letters: P for proportional, I for integral and D for derivative (Johnson & Moradi, 2005).

The proportional part of the controller *P* adjusts the control signal based on the input and the reference. The integral part *I* examines the error over time and changes the control signal. It eliminates the shortcoming of the proportional part (Johnson & Moradi, 2005). The derivative part *D* is based on the rate of change of the error, it keeps the system in at a consistent rate, resisting the change.

PID's can be in a variant structures such as parallel, time constant and series (Johnson & Moradi, 2005). The parallel PID is the "pure" form and it is being called

"textbook PID" according to the Johnson & Moradi (2005) due to not containing any modifications.

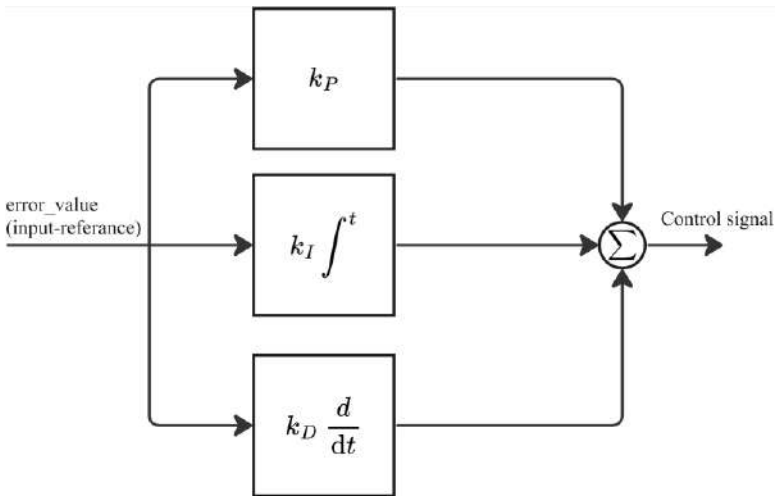


Figure 3.4 Block diagram of PID (adapted from (Johnson & Moradi, 2005))

Parallel PID consists individually coefficients for each part, k_P, k_I and k_D respectively. The formula for parallel PID is follows as (Johnson & Moradi, 2005):

$$u(t) = k_P e(t) + k_I \int^t e(t)dt + k_D \frac{de}{dt} \tag{3.3}$$

where *u(t)* – control signal based on time;
e(t) – error value based on time.

In the context of gain control, PID control can assist in smoothly adjusting the gain value. However, PID alone is insufficient for setting the gain as it is solely a control mechanism and cannot estimate the appropriate gain for a given image. Therefore, another algorithm must calculate the desired gain value and feed it into the PID as a reference value.

3.2. Gain control algorithms

This subsection explains three possible algorithms for prototyping the thesis. All algorithms described here are based on histogram statistics.

First paper by Cho et al. (2010), focuses on histogram equalization with AGC (Automatic Gain Control) aiming to stretch the dynamic range of the image. The authors generate a cumulative histogram based on the histogram of a grayscale image. A transformation function that is acquired by normalizing the cumulative histogram,

so that the gray level ranges between 0 to 255 for an 8-bit image on x-axis and the y-axis is between 0 to 1, is generated. The authors decided to quantize the histogram to reduce the computational complexity. Quantizing means reducing the number of bins in the histogram so that each gray level is represented as an interval (Figure 3.5).

At this point in the paper, the authors mention in the paper that the algorithm does not change the actual gain of the camera, but instead calculates the gain value

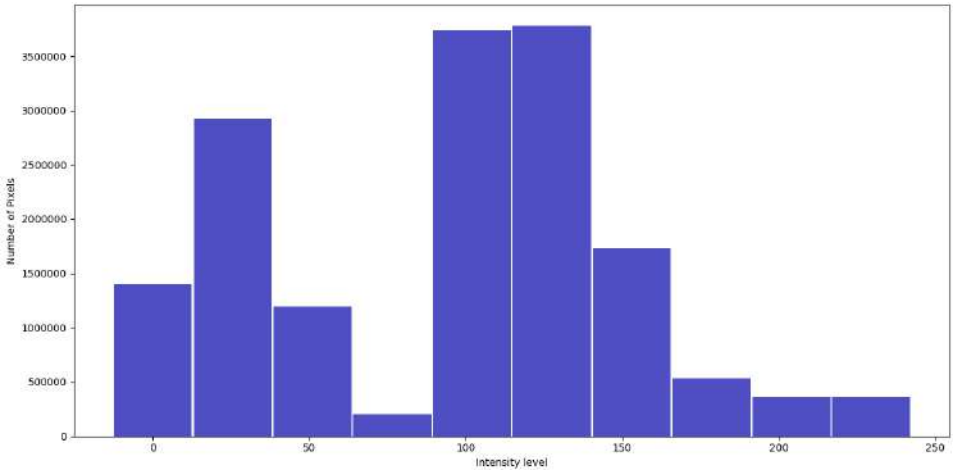


Figure 3.5 Example quantized histogram (Created by the author)

and multiplies it with the image. This is not a problem for the purpose of this thesis. After the histogram calculations and finding the desired value for the gain, the camera gain can be easily adjusted.

The algorithm determines whether the image requires a new gain by analyzing the first and last bins of the histogram. The first bin contains the darkest pixels, while the last bin contains the brightest pixels.

Algorithms does the following steps (Cho et al., 2010):

1) Multiplies the frame with gain (gain is set to 1 for initially);
2) Checks the first bin and last bin against their set threshold;
3) If one of the bins is higher than the threshold, it selects a new gain from a lookup table for that bin (first or last);
4) Multiplies the image with the new gain;
5) If in step 2, the first bin was not satisfying the threshold, then the new multiplied frame's last bin is subtracted by last 2 bins of the original frame and compared against the threshold. Opposite procedure happens if the last bin is not satisfying the threshold;

6) If the comparison in step 5 is satisfied, the algorithm sets the new gain and moves on to the next frame. If not, it goes back to step 3.

As the author Cho et al. (2010) states, the threshold values and the look up table is generated by studying large amount of images.

The result of the proposed algorithm shows visible improvement over the exposure of the image (Cho et al., 2010). The authors also mentioned and provided numerical results on the real-time usability of the algorithm.

The algorithm checks out the requirements for the purpose of this prototyping, but the need for the threshold values and a lookup table to decide the gain does require an additional study on the images taken in the greenhouse environment. This constraint makes the algorithm unable to adapt to different environments, and for the development of this prototype, imaging and studying the images in the greenhouse are not viable option.

Next paper by Rossi et al. (2011) combines histogram analyses with PID controller. The exposure level of the image and the error is calculated by histogram then the error value is fed to the PID controller.

To calculate error, the histogram is divided into 10 bins. Bins from 1 to 5 are considered dark zones and the bins from 6 to 10 light zones. Each bin has a predefined constant as a weight used in the error calculation. Dark zones' constant is positive and the light zones' constant is negative, referring to the need of increase or decrease in gain. As a reference, an average value is calculated. Average value is the total number of pixels divided by 2.

Calculation of error follows by (Rossi et al., 2011):

1) Bins that are exceeding the average value is determined;
2) the difference between number of pixel in the bin and the average value is calculated;
3) the difference then being multiplied by the predefined constant for each bin exceeding average value;
4) all the number then summed, and the resulting number is the error.

The sign and magnitude of the error determines the direction and the amount of the gain change.

In the paper, results on the histogram-based error calculation are not mentioned. This implies that accuracy for the resulting image exposure levels is not known. The results only mention the PID control and how it has been manually tuned.

With the lack of reliable results, predefined constants and reference values, as similar reasons to the algorithm from Cho et al. (2010), this algorithm is not suitable for the development of this prototype.

Third and the final algorithm to consider is from Torres & Menéndez (2015). The paper proposes an algorithm to achieve optimal exposure for video surveillance system in uncontrolled light conditions.

Before explaining the algorithm, some control variables has to be mentioned from the paper (Torres & Menéndez, 2015).

For a given image *B*, with *M* pixels, its histogram *h*, and the *K* as amount histogram bins:

Formula for the brightness or average value of pixel intensity:

$$Br = \frac{1}{M} \sum_{i=0}^{K-1} ih(i) \tag{3.4}$$

Contrast or the variance of pixel intensity:

$$Cr = \sqrt{\frac{1}{M} \sum_{i=1}^{M} (B(i) - Br)^2} \tag{3.5}$$

Algorithm also uses a cumulative histogram distribution *H* which is normalized version of cumulative histogram mentioned in the subsection 3.1.1.

White saturation level given that overexposure range (OER) ∈ [0, K-1]:

$$WSL = H(K - 1 - OER) \tag{3.6}$$

Black saturation level given that noise range (NR) ∈ [0, K-1]:

$$BSL = H(NR) \tag{3.7}$$

The paper then calculates two indexes, overexposure index id_{wsl} and underexposure index id_{bsl}. The id_{wsl} is the minimum number *r* that satisfies the $H(r) \geq 1 - ST$, where *ST* is saturation tolerance. The id_{bsl} is the minimum number *r* that satisfies the $H(r) \geq ST$.

The Author describes the indexes as; larger the id_{wsl} means overexposure, lower the id_{bsl} means underexposure.

This algorithm, like the other two, consists of some constant parameters for tuning. These are: saturation tolerance (ST), maximum contrast (C_{max}), overexposure range (OER) and noise range (NR).

For a given frame, algorithm first checks if the *WSL* is higher than *ST*, if it satisfies then the exposure has to be decreased. If not, then algorithms check for *BSL* greater than *ST* and C_{max} is greater than *Cr*, if satisfies then the exposure has to be increased. If both checks are not satisfied, then there is no need to change exposure.

The paper gives a complicated explanation on the calculation of how much the exposure should change. Simplified version is as follows (Torres & Menéndez, 2015):

When exposure needs an increase, id_{wsl} must be moved towards the boundaries of the *OER* without exceeding *ST* and get new id_{wsl} value. Then E_f, which is exposure value, id_{wsl} and new id_{wsl} goes into a calculation based on the bisection model to estimate a new E_f. When exposure needs to be decreased algorithm first checks if the WSL is above certain threshold, this means image is heavily overexposed and E_f is set to a defined reset value. If the threshold is not passed, then the algorithm calculates the decrease rate in a function. This is not provided in the paper and it is an assumption by the author of this thesis based on the information given in the (Torres & Menéndez, 2015):

$$u = \frac{1}{Ev_{max}} + WSL * \frac{\big((1-ST)Ev_{dec}\big) - Ev_{max}}{Ev_{max}((1-ST)Ev_{dec})} \tag{3.8}$$

where u – is decrease amount;

Ev_{max} – is maximum allowed change in one step;

Ev_{dec} – is decrease rate;

The decrease amount u then applied to the E_f and new exposure value is found. The algorithm then applies the new exposure value and moves on to the next frame.

The description of the algorithm has enough detail to be able to be implemented, but some assumptions, like equation

(3.8), must be made in order to complete the algorithm.

The proposed algorithm, compared to the other algorithms, has tuning parameters instead of constant target values. The real-time consideration and

uncontrolled light conditions corelates to the goal of this thesis and the results from the paper indicates that the algorithm proposed by Torres & Menéndez (2015) is most promising one compare to the other discussed algorithms.

The following chapter will include implementation of an algorithm to control gain based on Torres & Menéndez (2015).

3.3. White balance algorithms

As it is not known how the human vision system compensates for white balance, the algorithms for white balancing are not fully optimized. Different approaches exists such as machine learning, datasets analyses, use of color checkers and static algorithms (Barnard, Martin, et al., 2002; Gijsenij et al., 2011; C. Li et al., 2023)

Due to the limited computational power and environmental variance, it is not feasible to use machine learning and datasets for the prototype of this thesis. The use of color patches or checkers is also not viable since the camera will be located in a greenhouse and will be part of an automation system. These constraints leave only static algorithms.

The gray-world, white-patch, max-RGB and gray-edge algorithms will be explained and considered in this subsection. Except the gray-edge algorithm, other algorithms are before 2000, this makes it hard to find the original information. That is why these algorithms are studied and described from Gijsenij et al. (2011); Van De Weijer et al. (2007) and Zapryanov et al. (2012).

The gray-world algorithm, also mentioned in 2.3.2., makes a basic assumption that the average color of a scene is gray. The algorithm calculates the mean values for each channel, R_{avg}, G_{avg} and B_{avg}. Then using the mean values, it calculates coefficients: $corrR = \frac{R_{avg}}{G_{avg}}$ and $corrB = \frac{B_{avg}}{G_{avg}}$. Coefficient then used to correct the *R* and *B* color channels.

The white-patch algorithm assumes that the maximum values in each channels are caused by the illuminant on a reflective surface. Algorithm finds the brightest pixels for each channel R_{max}, G_{max} and B_{max}. Then calculates coefficients $corrR = \frac{G_{max}}{R_{max}}$ and $corrB = \frac{G_{max}}{B_{max}}$.

The max-RGB algorithm follows a similar path to the white-patch algorithm but instead of calculating coefficients for each individual channel, it calculates one coefficient based on the maximum value in all channels, then the coefficients is used for all channels.

The grey-edge algorithm proposed by Van De Weijer et al. (2007) is based on analyzing the color change across edges in the image to estimate the illumination color. It assumes that the average color change on edges is a shade of gray.

The paper provides an formula to about the calculation of the average (Van De Weijer et al., 2007):

$$\frac{\int |s_x^\sigma(\lambda, x)| dx}{\int dx} \tag{3.9}$$

Where s_x^σ – spatial derivative at scale σ and spatial position *x*;
λ – wavelength.

Simplified steps for the algorithm as follows:

1) Edge detection based on sharp color changes,
2) Color analyses for the sides of the edge,
3) Averaging the color changes on edges for the entire image,
4) Applying correction based on the average value.

All of the algorithms mentioned above claim to be fast enough to be used in a real-time environment. However, due to their simplicity and assumption-driven nature, their exact performance in a greenhouse environment has yet to be determined.

Based on the results from the papers (Gijsenij et al., 2011; Van De Weijer et al., 2007), grey-edge based algorithm will be implemented for white balance control of the camera.

4. IMPLEMENTATION

The following chapters include implementation procedure for gain control and white balance control as well as information on existing code.

4.1. Tools and technologies

Since the implementation will be developed into an existing project, tools, technologies, and programming language are defined by the project.

The project is based on top of the ROS (Robot operating system). ROS is a open-source for robot applications that offers low-level device control, commonly used functions, process and package management along with tools and libraries aiming to support code reuse in robotic development (*ROS/Introduction - ROS Wiki*, n.d.).

In the project ROS is used to manage each package which contains nodes along with libraries and configurations. Nodes are where the computation happens and nodes can communicate through ROS with each other, inside and outside the package.

The chosen programming language for the implementation is C++, like the rest of the project. C++ is a high-level programming language that was made with performance and efficiency in mind and provides object-oriented programming. Compare to other programming language, C++ achieves the fastest execution time, making it suitable for real-time processes (Alomari et al., 2015).

To make the overall process faster and responsive, multithreading is being used. Multithreading allows a process to divide into threads and allowing them to work asynchronously (Posch, 2017). For this project, a thread can be used to receive frames from the camera and another thread to control gain. This makes it so that neither part has to wait another.

The library “Boost” is used for threading applications. The Boost library is more than a multithreading library, it includes memory management, data structures and more but the main use of the library in the project is multithreading.

A software suit, Vimba SDK, is being used for the purpose of accessing the camera. Vimba SDK provides drivers for camera and provides API to control and receive data.

A library for frame handling and analyses is required. OpenCV (Open-Source Computer Vision Library) is a real-time computer vision library that includes functions such as image manipulation, color space conversion, histogram calculation and more.

Software directly runs on the Jetson; development requires a remote connection using SSH protocol.

4.2. Gain control

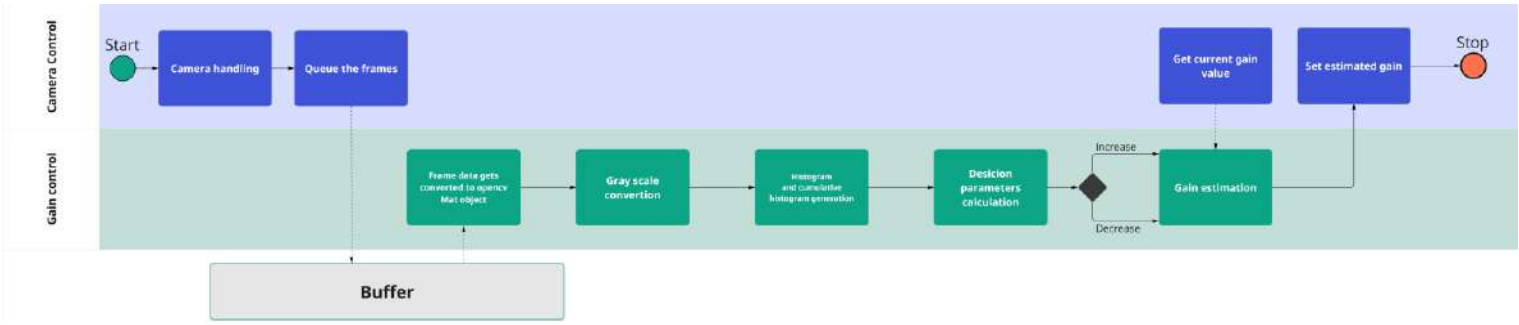


Figure 4.1 Simplified gain control diagram for one frame (Created by Author)

Each lane in the Figure 4.1 runs on a different thread. The “Camera control”, shown as first lane, handles Vimba API and ROS communications. Second lane “Gain control” estimates the optimal gain. And Buffer is where the frames are temporarily stored.

In the Camera control, messages or parameters from ROS get received and camera features read or set. Parameters for the camera are configured in ROS launch config. Camera control receives these parameters from ROS and using Vimba API sets each feature (camera attribute) to the camera in the initialization process.

This thesis does not focus on how Camera control and Buffer work as they are beyond its scope.

Gain control starts with converting the frame data from buffer to OpenCV’s *Mat* class object:

```
cv::Mat color_frame(cv::Size(width_, height_), CV_8UC3, (unsigned char*) &data[0]);
cv::resize(color_frame, color_frame, cv::Size(320, 240));
```

Mat object is an n-dimensional multi-channel array that, in this context, represents a frame. The arguments to initialize the objects are size of the image; type

of data stored, in this case it is 8-bit unsigned integer with 3 channels representing R, G and B; values to populate the Mat object, which comes from the Buffer. Second line does a resizing of the image. This is done so that the required computational power decreases.

Before calculating the histogram of the frame, conversion from RGB color space to grayscale takes place:

```
cv::Mat gray_frame;
cv::cvtColor(color_frame, gray_frame,cv::COLOR_BGR2GRAY);
```

The cvtColor() takes 3 arguments: source, destination and color space. It is important to write correct color space, failing so can cause faulty calculations or even loss of data.

Calculation of the histogram is done by OpenCV function calcHist() by providing gray_frame and the range of histogram.

To calculate the cumulative histogram, a basic function needs to be written as OpenCV does not include one:

```
std::vector<float> cumHist(const cv::Mat& hist, int pixels){
  std::vector<float> cumulativeHist(hist.rows);
  float sum = 0.0f;
  for (int i = 0; i < hist.rows; ++i) {
        sum += hist.at<float>(i);
        cumulativeHist[i] = sum / pixels; // Normalize to range [0, 1]
  }
  return cumulativeHist;
}
```

The function follows the description in the subsection 3.1.1, equation (3.2), additional to the equation, it normalizes the cumulative histogram so that the values range from 0 to 1.

After generating the histogram and cumulative histogram, decision parameters, *Br*, *Cr*, *WSL*, *BSL*, id_{wsl} and id_{bsl} from subsection 3.1.1, are calculated. Implementation of these calculations can be seen in Appendix 1.

At this stage in the implementation, decision about gain increase or decrease can be made with *WSL* and *BSL*. Based on the description of the algorithm, simple if else statement can be used to control the gain:

```
if ( wsl > (1-ST_)){
  gainValue_ +=0.5;
} else if (bsl > ST_ | imagecr < cMax_) {
  gainValue_ -=0.5;
}
```

By controlling in this manner, the estimation process is skipped, but the transition in gain occurs more slowly. Additionally, the algorithm may want to maintain a certain level while still producing overexposed images. A function to estimate the gain should be implemented instead of changing it in increments.

Two separate functions are used to estimate new gain. To increase the exposure the following function is used:

```
float increaseGain(const std::vector<float>& cumhist, float gain, float idwsl,float wsl){
   int index = findIndex(cumhist,wsl)
   float n_idwsl = cumhist[index];
   return (n_idwsl*gain)/idwsl;
}
```

It calculates a new id_{wsl} which is at the border of *WSL*. Then based on the new id_{wsl} and current gain, new gain is calculated.

To decrease the gain, the equation (3.8) is calculated with in a function:

```
float decreaseGain(float wsl){
   return 1/evMax + (wsl * (((1-ST_)*evDec)-evMax)/(evMax*(((1-ST_)*evDec))));
}
```

In the main operation part of the code, an if-else statement checks the conditions for WSL, BSL and Cr of the image to decide if gain has to be increased or decreased:

```
if ( wsl < (1-ST_)){
   std::cout << "decrease exposure" << std::endl;
```

```
        float decAmount = decreaseGain(wsl);
        std::cout << "decreaseamount: " << decAmount << std::endl;
        gainValue_ += decAmount;

    } else if (bsl > ST_ | imagecr < cMax_) {
        std::cout << "increas exposure" << std::endl;
        getGainValue(gainValue_);
        gainValue_ = increaseGain(cumhist, gainValue_, idwsl, wsl);
    }
```

This if-else comparison is different from the description in the paper Torres & Menéndez (2015). After experimenting, decreasing gain if the *WSL* is lower than the *1-ST* and increasing gain if the *BSL* is higher than *ST* found out to be the better approach.

After acquiring the new gain, it should be checked so that the value is always in between 0.5 and 45. These numbers are the gain limitation of the camera in use. Gain control then sends the new gain value to the Camera control where then it is set to the camera. The function that sets the gain can be seen in Appendix 2.

In this state of the gain control algorithm, the optimal exposure is approached, but the gain value is either overshot or undershot, resulting in a continuous decrease or increase, creating an oscillation. This is because the gain value that satisfies both statements is either very precise or does not exist. Furthermore, the minimum increment to the gain value accepted by the camera is 0.1, which reduces the probability of satisfying both cases. So to overcome this oscillation, A simple PID, mentioned in the 3.1.2, is used based on the Beauregard (2017). This code can be found in Appendix 3.

Instead of directly setting the new gain, the current gain is passed to the PID as an input and the new gain is as setpoint. Then PID computes a *step* value to move the current gain towards the new gain in a controlled manner. The algorithm then increases the current gain with *step* and sets the new gain.

4.3. White balance control

Like Gain control, White balance control also works in thread along with Camera control and getting the frame data from the Buffer (Figure 4.2).

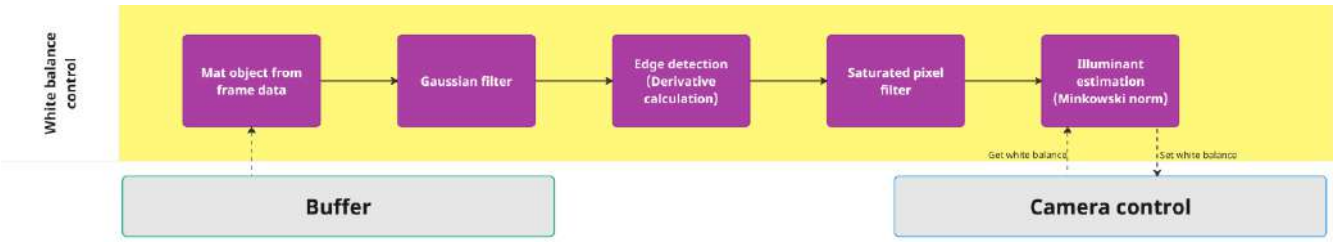


Figure 4.2 Simplified white balance control diagram (Created by Author)

White balance control is implemented with the help of GitHub repository from Sami & Khaled (2019). The repository includes python implementation of the method presented in the Van De Weijer et al. (2007) with other algorithms such as grey-world and max-RGB. The repository serves as a reference for our implementation of the grey-edge algorithm in C++.

Implementation starts with applying a gaussian blur to the frame received from the camera. Gaussian blur is based is a image blurring method on gaussian distribution.

```
cv::Mat gaussImage;
cv::GaussianBlur(image, gaussImage, cv::Size(2 * sigma + 1, 2 * sigma + 1),1);
```

The *cv::Size()* and *sigma* determine the magnitude of the blurring applied.

The Gaussian blur is applied to the image in order to "smooth" it, thereby preventing the edge detection algorithm from generating false positives and allowing it to identify edges where there is a significant change between the sides of the edge.

Next is to apply a edge detection algorithm. For this purpose of the thesis, Laplacian edge detection method will be used. Laplacian takes the 2nd derivative of the image and finds the zero crossings (roots) (Torre & Poggio, 1986).

```
std::vector<cv::Mat> deriv_image(3);
std::vector<cv::Mat> cahn(3);
cv::split(gaussImage, cahn);
for(int i = 0; i < 3; ++i){
        cv::Mat laplance_image;
        cv::Laplacian(cahn[i], laplance_image, CV_64F, 3);
        laplance_image = cv::abs(laplance_image);
```

```
            deriv_image[i] = laplance_image;
    }
```

The *deriv_image* is defined to hold the processed channels of the image. Laplacian has to be applied to each channel(RGB) separately. To do this *gaussImage* is separated to its channels with the function *cv::split()*. Then for each channel, edge detection is applied by the function *cv::Laplacian()*. By the nature of the Laplacian, it can include negative values in its output, to prevent this absolute value of it is taken with *cv::abs()*. The processed channel then is placed in *deriv_image* in the same order as the channels of the input image.

Saturated pixels need to be filtered in order to assure accuracy on the estimation of the grey-edge algorithm. Saturated pixels filtered based on the input image. If the pixel at the input image has a value of 255, which is the maximum value in a 8-bit image, then this pixel is set to 0 in the *deriv_image* (Appendix 4).

Estimation of the illuminant color includes finding Minkowski norm for each channel. Minkowski norm, also known as Minkowski distance, in this algorithm is used to estimate the direction of the illumination light.

```
cv::Vec3d illum;
for (int channel = 0; channel < 3; ++channel) {
   cv::Mat powered;
   cv::pow(deriv_image[channel], mink_norm, powered);

   long sum = cv::sum(powered)[0];
   sum = std::pow(sum, 1.0/mink_norm);
   illum[channel] = sum;
}
```

Each channel is powered by Minkowski norm (*mink_norm*) revealing the magnitude of the change in the *deriv_channel*. Then the channel values are summed to get an overall understanding of the magnitude change. Lastly this sum is rooted by *mink_norm* to normalize it. This value is then stored in the vector *illum*.

After calculating estimation for all channels, one more normalization is applied to the illum to have the values between 0 and 1.

At this stage of the white balance control, it is able to give an estimation of the illuminant color but changing camera settings is not achieved.

The current camera in use uses channel ratios to control white balance. These ratios are red-to-green and blue-to-green, while green channel has fixed ratio. After some experimentation, it has been determined that utilizing the output of the grey-edge algorithm, which is the illuminant color estimation, in conjunction with the ratio setting provided by the camera is a complex issue that necessitates a more advanced approach than simply taking ratios of the estimation.

As the paper by the Bianco et al. (2012), estimation can be used to correct the image with matrix multiplication in the pipeline of the camera. In the context of this thesis, it is not possible to change the pipeline and applying multiplication to the frame is not desired since the aim is to tune the camera in real-time.

Given the situation, a correlation between the estimated illuminant color and the ratios of the camera must be found.

In this stage of the implementation, problems about hardware related to the project thus to thesis had occurred. The Nvidia Jetson that was mentioned in the section 2.1 had to be returned. The fact that all the code and configurations were located in the Jetson meant that the development of the prototype was brought to a halt. Solution was found as containerizing the Jetson but the process took two and a half week.

Due to the unfortunate circumstances that have arisen, the author of the thesis has decided to not pursue the further implementation of white balance control, given the constraints on time.

5. EXPERIMENTATION

This chapter provides information on experimentation settings and mathematical evaluation criteria.

5.1. Acquisition

Due to the hardware issues and the resulting time loss, the experiments could not be conducted in a greenhouse environment. As a result, the experiments were conducted in an office environment where the possibility of utilizing natural light from the window and the ability to control the blinds to simulate varying light levels were available. Furthermore, the office was equipped with two sets of warm (yellowish) LED lighting allowing different configurations to conduct test.

Although the implementation of the white balance control is not finished, the current state of the implementation does run the grey-edge algorithm which gives the illuminant estimation. To provide a comparison of the grey-edge algorithm to the built-in algorithm, frames can be acquired with a fixed setting and then a diagonal matrix transformation can be applied using the illuminant estimations. This will provide an estimation of what could be achieved if the white balance control were fully implemented.

For the testing of the gain algorithms, 4 settings have been used along with the built-in automatic white balance. These settings are:

- Dark (D): where all light sources blocked or switched off;
- Dark natural (DN): blinds are set to half and lights are off;
- Natural (N): blinds are open and light are off;
- Full (F): blinds are closed and one set of lights are on.

In all test settings for gain, the camera is set to take 10 frames per second, the exposure time is set to 100ms, and the lens aperture is set to f/1.8 with manual focus.

Another test setting with 2 sets of lights has been tested but, because the system only changes gain, the resulting image highly overexposed even though the gain is set to lowest possible value (Figure 5.1).

Figure 5.1 Problem of controlling only gain in strongly lit environment

To make it easier to refer the algorithms and test settings, “Built” for built-in mode and “Imp” for implemented gain control followed by the name of the setting will be used (e.g. “BuiltD” for built-in mode in dark setting or “ImpF” implemented algorithm in full setting) (Figure 5.2).

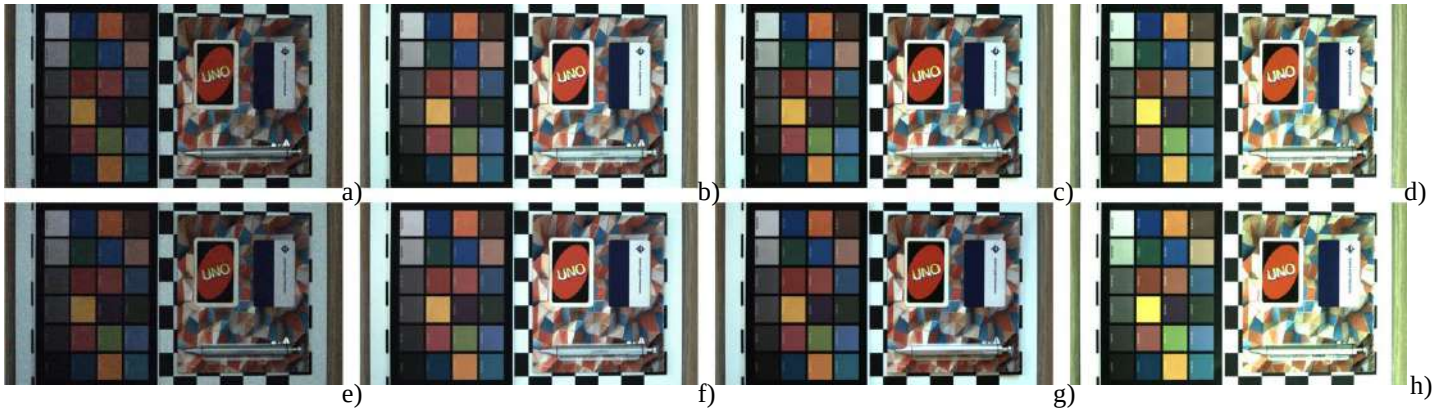


Figure 5.2 Acquired test images; a) *BuiltD*, b) *BuiltDN*, c) *BuiltN*, d) *BuiltF*, e) *ImpD*, f) *ImpDN*, g) *ImpN*, h) *ImpF*

In addition to the acquired images, a reference image is generated for each setting through the process of editing the image's exposure with image processing tools (Figure 5.3). This involves manually adjusting the exposure to achieve the realistic exposure for that setting.

For white balance, test are only conducted in the natural (N) and full (F) conditions with built-in gain control enabled. As previously stated, the grey-edge algorithm is partially implemented. To facilitate a potential comparison with the built-in white balance, the images are

Figure 5.3 Test images compare to reference image; a) *BuiltDN*, b) *ImpDN*, c) *RefDN*

acquired with a fixed white balance setting, after which the illuminant estimation from the grey-edge is applied to the image with a diagonal matrix transformation (Figure 5.4).

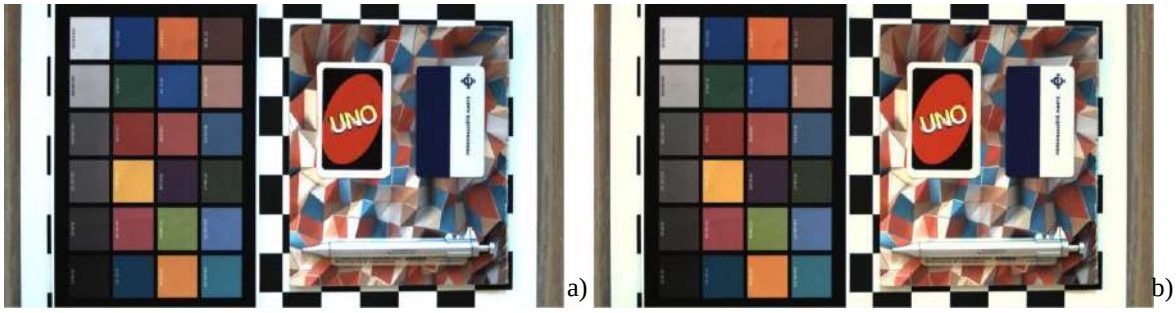


Figure 5.4 Grey-edge transformation: a) before transformation; b) after transformation

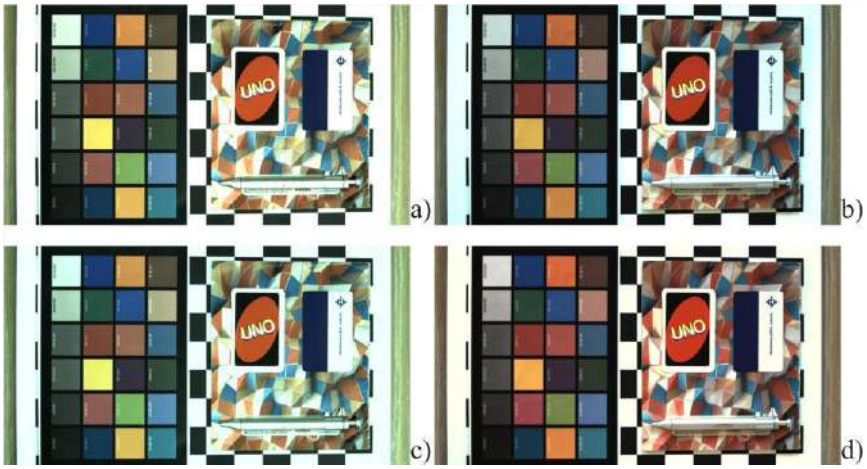


Figure 5.5 White balance tests: a) *BwF*, b) *BwN*, c) *ImpwF*, d) *ImpwN*

Similar to gain algorithms, for built-in white balance “Bw” and for implemented grey-edge “Impw” along with setting name will be used (Figure 5.5).

References for white balance algorithms are also generated in the similar way to the gain references, using a image processing tool to match the color to real life looks (Figure 5.6).

Figure 5.6 White balance tests compared to reference: a) built-in, b) grey-edge, c) reference

5.2. Evaluation criteria

The images acquired with different algorithms will be compared in two distinct ways: visually and mathematically.

A visual comparison is a straightforward process whereby the images and their respective luminosity histograms are evaluated based on observations. Furthermore, to compare the gain control algorithms, the images will be divided into distinct regions representing the highlights and darks. This will allow for a more comprehensive understanding of the results.

Mathematical comparison will be different for gain control algorithms and white balance algorithms.

For gain control algorithms quality assessment measures Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM)(Wang et al., 2004), Visual Information Fidelity (VIF)(Sheikh & Bovik, 2006) and Universal Quality Image Index (UQI)(Zhou Wang & Bovik, 2002) will be used. These metrics are widely used

for image quality assessments (Athar & Wang, 2019; Sheikh et al., 2006; Wang & Bovik, 2006). While they are not typically employed in the comparison of image exposure, when a reference image is provided, they yield a score indicating the distance between the test image and the reference.

PSNR is the ratio between signal power and noise. It is mainly used in comparing original image to compressed image. The reason to use in this comparison is to include “more traditional” measure. Higher values of PSNR refers to image is closer to the reference image.

SSIM is a index that is based on luminance, contract and the structure of the image. SSIM also considers the human visual system (Wang et al., 2004). The SSIM index closer to 1 indicates higher similarity to the reference image.

UQI models the image in three factors: loss of correlation, luminance distortion and contrast distortion (Zhou Wang & Bovik, 2002). Closer UQI to 1 refers to closer to reference image.

VIF is an index based on natural science statistic and model of human visual system (Sheikh & Bovik, 2006). A higher VIF index indicates the test image is closer to reference image.

For white balance algorithms, a color difference equation CIEDE2000 is used. CIEDE2000 is an equation that has been approved by the International Commission of Illumination (CIE) that aims to represent color differences as perceived by the human visual system (Luo et al., 2001). Given two colors, it yields a number indicating the distance of the colors with smaller values means closer to the reference this better white balance. For the purpose of this thesis, the CIEDE2000 equation will be applied to each pixel within a specified region of the image (Figure 5.7).

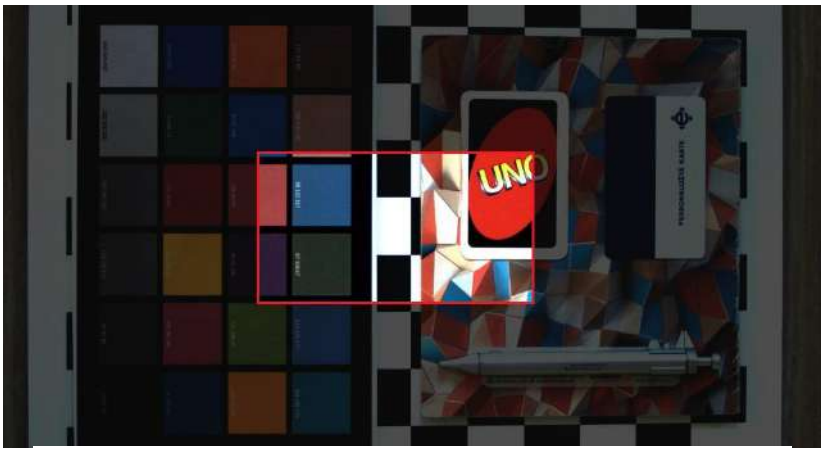


Figure 5.7 Region of color difference equation

The resulting values will then be averaged to obtain an average color distance. For the tests gain algorithm's tuning parameters are set as seen in the Table 5.1 and grey-edge algorithm parameters in Table 5.2.

Table 5.1

Gain Control Parameters

Parameters	**Values**
ST	0.06
C_{max}	100.0
OER	10.0
NR	40.0
Ev_{max}	24.0
Ev_{min}	0.6

Table 5.2

Grey-Edge Parameters

Parameters	**Value**
Minkowski norm (*mink_norm*)	5.0
Sigma	5.0

6. RESULT ANALYSIS

This chapter includes comparisons and results of gain and white balance algorithms based on the experiments done in the chapter 5.

6.1. Gain algorithms comparison

Experimental results in a *dark* and *full* setting demonstrate that the built-in and implemented algorithms yield highly comparable outcomes. This similarity can be clearly seen in the histogram of the test images (Figure 6.1).

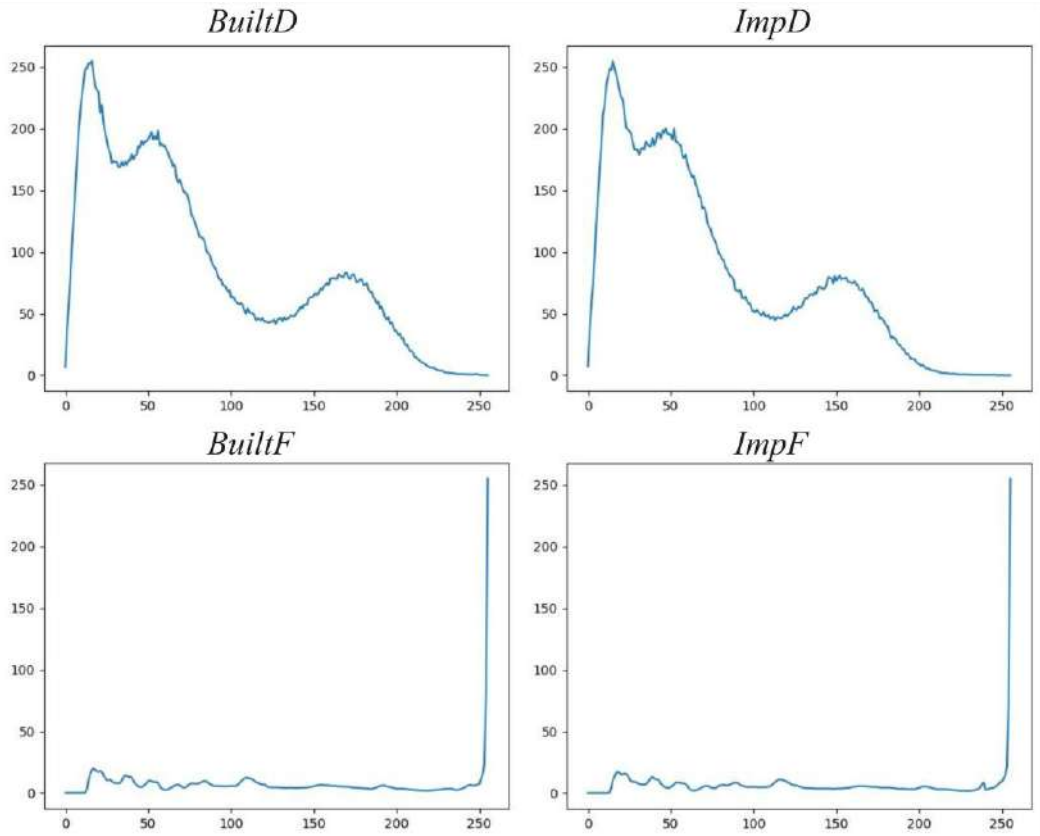


Figure 6.1 Histogram comparisons in *dark* and *full* settings

This similarity may be perceived as the performance of both algorithms being identical. However, these similarities can be largely attributed to the fact that both algorithms approached the camera's gain range limits. Therefore, it is unjustified to conclude that their performance is similar. What can be observed in these settings is that both algorithms are capable of understanding the dark environment and increasing the gain to the limits and understanding an over lit environment and decreasing the gain to the minimum.

The actual difference between the algorithms can be seen in the *natural(N)* and *dark natural(DN)* settings.

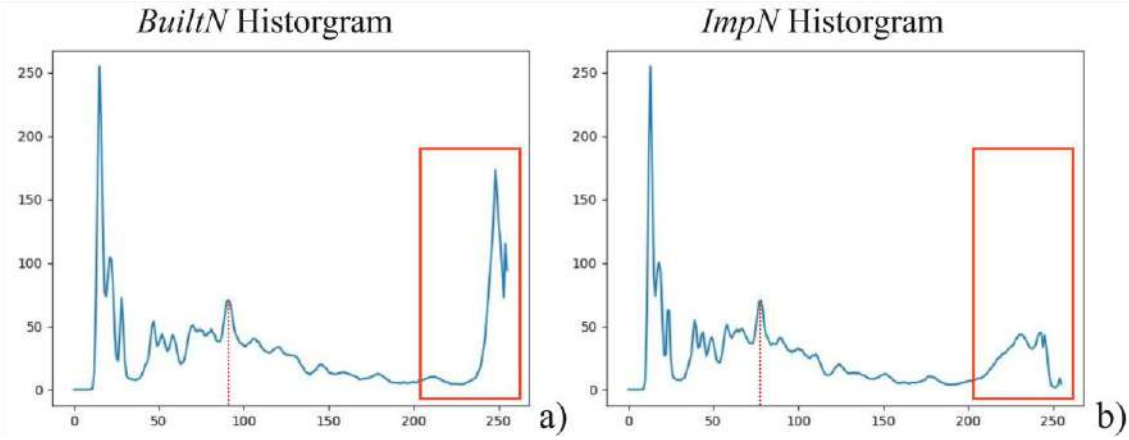


Figure 6.2 Histogram of both algorithms in *natural* setting: a) built-in algorithm, b) implemented algorithm; red marks indicate differences

As seen in the Figure 6.2a and Figure 6.2b, there is a clear visual difference in the red box region. This region, situated on the right side of the histogram, represents brighter pixels. The image taken with built-in algorithm has more bright pixels than the image taken with implemented algorithm. This indicates that the implemented algorithm functions as intended, effectively try to prevent overexposure, as stated in the original paper being a primary goal. Furthermore, the red dotted line is shifted to the left. This further demonstrates that the implemented algorithm selects a lower gain level to prevent overexposure of the image, thus shifting the histogram to the left.

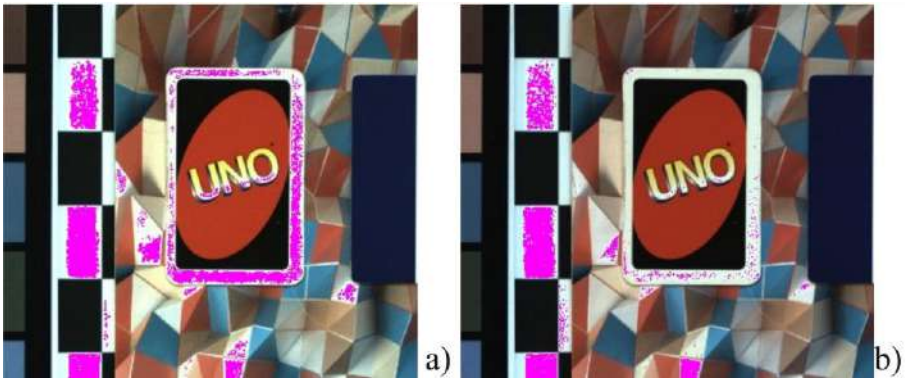


Figure 6.3 Test images with overexposed areas marked with purple: a) built-in, b) implemented

The difference seen in Figure 6.2 can be supported by Figure 6.3, which shows that the implemented algorithm produced fewer overexposed areas in the image. Similar result can also be seen with the setting *dark natural(DN)*(Appendix 5).

Table 6.1

Gain Algorithm's Mathematical Measures - 1

Settings	Algorithms	PSNR	SSIM	UQI	VIF
Dark (D)	Built-in	19.171	0.438	0.898	0.191
	Implemented	19.917	0.933	0.918	0.877
Full (F)	Built-in	12.818	0.435	0.819	0.207
	Implemented	12.172	0.423	0.796	0.203

Mathematical comparison of both algorithms in the *dark* and *full* setting can be seen in the Table 6.1. It is important to remember that these comparisons are not being made to one another, but rather to a reference image.

The built-in algorithm in the *dark* setting exhibits considerably lower values for *SSIM* and *VIF* index scores compared to those of the implemented algorithm in the same setting. The observed results are somewhat unexpected, given that the images and their respective histograms appear to be quite similar. This is highly likely because of the high gain value in the dark setting caused significant amount of noise on the image. Since these noise patterns differ for each image, it could be that the reference image has similar pattern to the *ImpD* thus calculated indexes can be favoring implemented over the built-in algorithm (Figure 6.4).

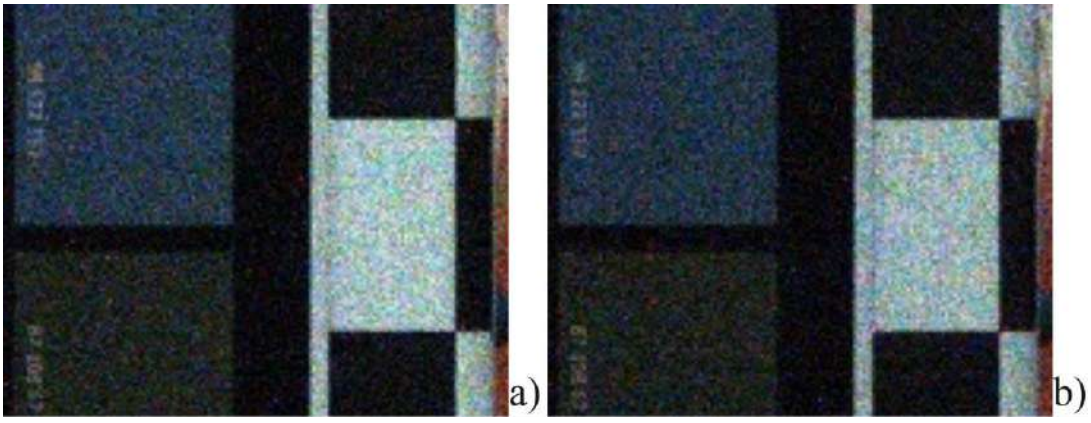


Figure 6.4 Noise patterns of test images: a) *BuiltD*, b) *ImpD*

Considering that the algorithms reached maximum value of gain in both settings and *dark* setting yields biased measures, *full* and *dark* setting will not be considered in the following analyses for this thesis.

Table 6.2

Gain Algorithm's Mathematical Measures - 2

Settings	Algorithms	PSNR	SSIM	UQI	VIF
Dark natural (DN)	Built-in	18.067	0.937	0.915	1.074
	Implemented	24.502	0.985	0.981	1.025
Natural (N)	Built-in	18.241	0.943	0.917	1.067
	Implemented	30.9	0.996	0.995	1.004

As the Table 6.2 shows, algorithms have similar results on the different settings with implemented algorithm having a higher score except the *VIF*. Upon examination of the index values, it can be observed that the implemented algorithm yielded a higher *PSNR*, indicating a lower in noise within the image. As previously stated, the implemented algorithm produced an image with less overexposure areas using a lower gain value. Given that the gain is directly correlated with noise, the observed difference in *PSNR* index can be attributed to this.

The SSIM and UQI indexes both indicates that the image from implemented algorithm resembles slightly closer image to the reference than the built-in algorithm.

In terms of *VIF*, small difference can mean that the built-in algorithm captures visual information slightly better than the implemented algorithm.

In general, both algorithms produced similar outcomes, with the implemented algorithm lower on the gain value and exhibiting fewer overexposed areas.

6.2. White balance algorithms comparison

To analyze the potential performance of the grey-edge and compare it to the built-in white balance algorithm, histogram of the images can be looked. It is important to remember that the white balance control is not fully implemented, and images here are produced by using grey-edge algorithms illuminant estimation.

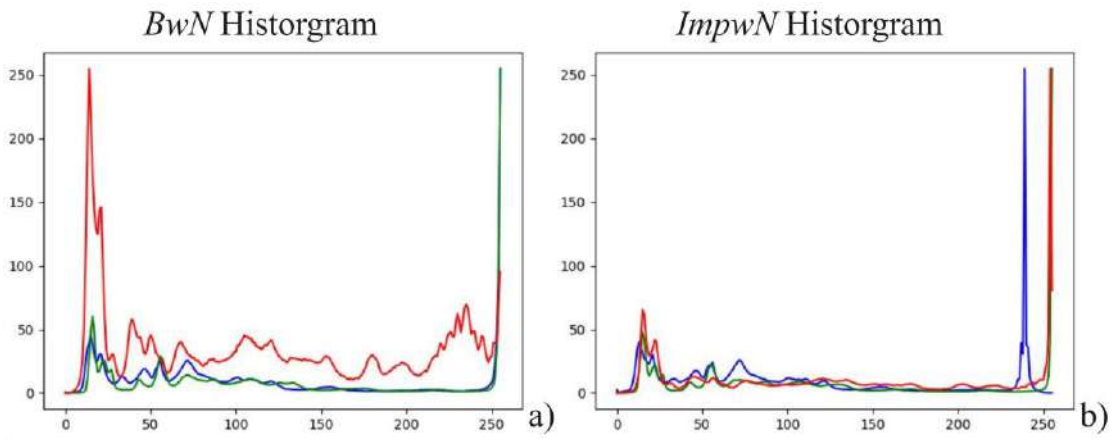


Figure 6.5 Histogram comparison of white balance algorithms: a) built-in, b) implemented (grey-edge)

Examining the histograms in Figure 6.5 reveals that the built-in white balance opts for brighter green and blue values while lowering the red tones. Meanwhile, the grey-edge decides on brighter reds and greens and slightly lower blues. As it is seen in the Figure 6.5, the big difference is on the red channel. This move also can be seen on the images in Figure 6.6.

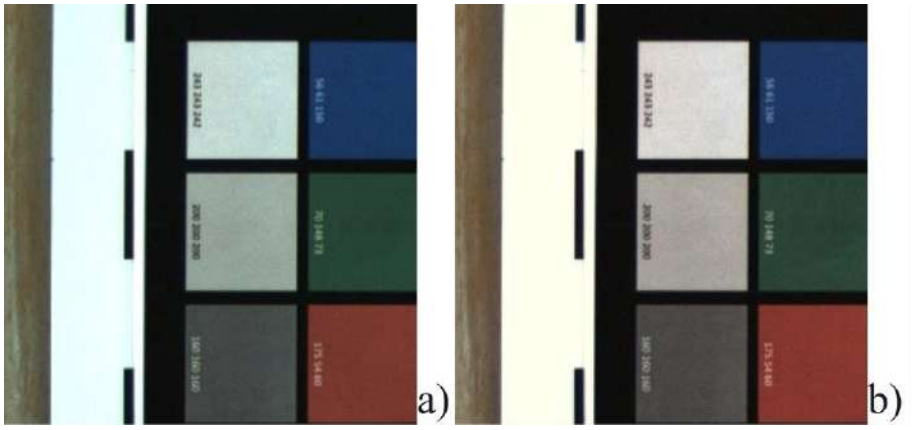


Figure 6.6 Color difference on the setting *natural*: a) built-in, b) grey-edge

Since the conducted experiments are not many in quantity, it is not sufficient to get a conclusion from visual inspects.

By analyzing Table 6.3, white balance algorithms can be further investigated. In the setting *natural*, grey-edge achieves to be closer to the reference image colors by a small margin. Although it is a small difference on the CIEDE2000 index, Figure 6.6 proves it to be a noticeable difference on the actual image.

Table 6.3

White balance algorithms CIEDE2000 Measures

Settings	Algorithms	CIEDE2000
Natural (N)	Built-in	5.861
	Grey-Edge	5.583
Full (F)	Built-in	8.220
	Grey-Edge	10.001

In the other hand, built-in white balance had surpassed the grey-edge algorithm in the *full* setting. It is relatively straightforward to suggest that the grey-edge algorithm performed better at natural condition while the built-in algorithm performed better in full condition. However, given that only a single experiment was conducted for each setting, it is not possible to come to such a conclusion.

CONCLUSION

The objective of this thesis was to develop a software prototype for real-time tuning of an RGB camera to achieve optimal exposure and color in a greenhouse environment. The prototype was required to adapt to changing environmental conditions and operate in real-time.

The thesis began with the history and working principles of digital camera, with an emphasis on their use in forestry and greenhouse applications. The subsequent section examined the role of exposure and white balance as key components for achieving optimal image quality. Then, several algorithms were considered, with one gain algorithm fully implemented and grey-edge white balance algorithm partially implemented.

The results of the experiments show promising results, particularly for the gain algorithm.

While the gain algorithm does function, oscillations still occur. It is recommended that the PID and the algorithm itself be tuned properly, with tests conducted both indoors and outdoors, and if possible, in a greenhouse environment. Furthermore, another tuning should be conducted in the deployment environment of the camera.

For white balance, a correlation should be found between the illuminant estimation of the gray-edge algorithm and the white balance ratio settings of the camera. The simplest way to find this would be to test the algorithm in a controlled environment and change the camera ratios to see the changes in illuminant estimation, but this approach can be time consuming, so another approach may be considered, or changing the white balance algorithm to something more appropriate for this camera may also be considered.

In conclusion, while the initial outcomes are promising, further research and fine-tuning are necessary to fully realize the potential of the developed prototype.

LIST OF REFERENCES

Afifi, M., Price, B., Cohen, S., & Brown, M. S. (2019). *When Color Constancy Goes Wrong: Correcting Improperly White-Balanced Images*. 1535–1544. https://openaccess.thecvf.com/content_CVPR_2019/html/Afifi_When_Color_Constancy_Goes_Wrong_Correcting_Improperly_White-Balanced_Images_CVPR_2019_paper.html

Allen, E., & Triantaphillidou, S. (2011). *The manual of photography* (10th ed). Elsevier/Focal Press.

Alomari, Z., Halimi, O. E., Sivaprasad, K., & Pandit, C. (2015). *Comparative Studies of Six Programming Languages*. https://doi.org/10.48550/ARXIV.1504.00693

Athar, S., & Wang, Z. (2019). A Comprehensive Performance Evaluation of Image Quality Assessment Algorithms. *IEEE Access*, *7*, 140030–140070. https://doi.org/10.1109/ACCESS.2019.2943319

Barnard, K., Cardei, V., & Funt, B. (2002). A comparison of computational color constancy algorithms. I: Methodology and experiments with synthesized data. *IEEE Transactions on Image Processing*, *11*(9), 972–984. https://doi.org/10.1109/TIP.2002.802531

Barnard, K., Martin, L., Coath, A., & Funt, B. (2002). A comparison of computational color constancy Algorithms. II. Experiments with image data. *IEEE Transactions on Image Processing*, *11*(9), 985–996. https://doi.org/10.1109/TIP.2002.802529

Barron, J. T. (2015). *Convolutional Color Constancy*. 379–387. https://openaccess.thecvf.com/content_iccv_2015/html/Barron_Convolutional_Color_Constancy_ICCV_2015_paper.html

Barron, J. T., & Tsai, Y.-T. (2017). *Fast Fourier Color Constancy*. 886–894. https://openaccess.thecvf.com/content_cvpr_2017/html/Barron_Fast_Fourier_Color_CVPR_2017_paper.html

Barry, C. (2016, June 25). *A Beginner's Guide to Aperture, Shutter Speed, and ISO in Photography*. PetaPixel. https://petapixel.com/aperture-shutter-speed-iso-beginners-guide-photography/

Beauregard, B. (2017, June 20). *Improving the Beginner's PID – Introduction*. http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/

Bellis, M. (2016). *History of photography and the camera*. *95*.

Bernacki, J. (2020). Automatic exposure algorithms for digital photography. *Multimedia Tools and Applications*, *79*(19), 12751–12776. https://doi.org/10.1007/s11042-019-08318-1

Beyerer, J., Puente León, F., & Frese, C. (2016). *Machine Vision: Automated Visual Inspection: Theory, Practice and Applications*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-47794-6

Bianco, S., Bruna, A., Naccari, F., & Schettini, R. (2012). Color space transformations for digital photography exploiting information about the illuminant estimation process. *Journal of the Optical Society of America A*, *29*(3), 374. https://doi.org/10.1364/JOSAA.29.000374

Burger, W., & Burge, M. J. (2022). *Digital Image Processing: An Algorithmic Introduction*. Springer International Publishing. https://doi.org/10.1007/978-3-031-05744-1

Canon. (2019, May 27). *Camera Lenses Explained*. Canon Australia. https://www.canon.com.au/get-inspired/camera-lenses-explained-jenn-cooper

Cho, J., Jin, S., Kwon, K., & Jeon, J. (2010). A Real-Time Histogram Equalization System with Automatic Gain Control Using FPGA. *TIIS*, *4*, 633–654. https://doi.org/10.3837/tiis.2010.08.0011

Dash, J., Pont, D., Brownlie, R., Dunningham, A., Watt, M., & Pearse, G. (2016). Remote sensing for precision forestry. *New Zealand Journal of Forestry*, *60*, 15–24.

de Luna, R. G., Dadios, E. P., & Bandala, A. A. (2018). Automated Image Capturing System for Deep Learning-based Tomato Plant Leaf Disease Detection and Recognition. *TENCON 2018 - 2018 IEEE Region 10 Conference*, 1414–1419. https://doi.org/10.1109/TENCON.2018.8650088

Douglas, S. M. (2005). *COMMON TREE HEALTH PROBLEMS*. https://portal.ct.gov/-/media/CAES/DOCUMENTS/Publications/Fact_Sheets/Plant_Pathology_and_Ecology/COMMONTREEHEALTHPROBLEMSpdf.pdf

Edmund Optics. (2015a, September 23). *Contrast*. https://www.edmundoptics.com/knowledge-center/application-notes/imaging/contrast/

Edmund Optics. (2015b, September 24). *Imaging Electronics 101: Basics of Digital Camera Settings for Improved Imaging Results*. https://www.edmundoptics.com/knowledge-center/application-notes/imaging/basics-of-digital-camera-settings-for-improved-imaging-results/

Edmund Optics. (2021). *Imaging Optics* (V213E ed.).

European Commission. Eurostat. (2011). *Forestry in the EU and the world: A statistical portrait*. Publications Office. https://data.europa.eu/doi/10.2785/13022

Forteck Enviro. (2020, September 22). *Stages Of Forestry*. Forteck. https://www.forteck.ca/post/stages-of-forestry

Foster, D. H. (2011). Color constancy. *Vision Research*, *51*(7), 674–700. https://doi.org/10.1016/j.visres.2010.09.006

Fuentes, I. O. H., Bravo-Zanoguera, M. E., & Yanez, G. G. (2009). FPGA Implementation of the Bilinear Interpolation Algorithm for Image Demosaicking. *2009 International Conference on Electrical, Communications, and Computers*, 25–28. https://doi.org/10.1109/CONIELECOMP.2009.47

Gain and ISO. (2022). In *Video Production Handbook*. Laney College, City College of San Francisco, San Francisco State University, Los Angeles City College. https://human.libretexts.org/@go/page/124312

Gijsenij, A., Gevers, T., & van de Weijer, J. (2011). Computational Color Constancy: Survey and Experiments. *IEEE Transactions on Image Processing*, *20*(9), 2475–2489. https://doi.org/10.1109/TIP.2011.2118224

Guoxiang, S., Yongbo, L., Xiaochan, W., Guyue, H., Xuan, W., & Yu, Z. (2016). Image segmentation algorithm for greenhouse cucumber canopy under various natural lighting conditions. *International Journal of Agricultural and Biological Engineering*, *9*(3), 130–138. https://doi.org/10.25165/ijabe.v9i3.2102

Hartley, R., & Zisserman, A. (2018). *Multiple view geometry in computer vision* (2. edition, 17.printing). Cambridge Univ. Press.

Hassankhani, R., & Navid, H. (2012). Qualitative Sorting of Potatoes by Color Analysis in Machine Vision System. *Journal of Agricultural Science*, *4*. https://doi.org/10.5539/jas.v4n4p129

Holmgren, P., & Thuresson, T. (1998). Satellite remote sensing for forestry planning—A review. *Scandinavian Journal of Forest Research*, *13*(1–4), 90–110. https://doi.org/10.1080/02827589809382966

Hornberg, A. (2017). *Handbook of Machine and Computer Vision: The Guide for Developers and Users*. John Wiley & Sons, Incorporated. http://ebookcentral.proquest.com/lib/rtulv-ebooks/detail.action?docID=4821050

IDS Imaging Development Systems GmbH. (n.d.). *IDS peak manual* [User manual]. Retrieved March 19, 2024, from https://www.1stvision.com/cameras/IDS/IDS-manuals/en/index.html

Johnson, M. A., & Moradi, M. H. (Eds.). (2005). *PID Control*. Springer-Verlag. https://doi.org/10.1007/1-84628-148-2

Kim, J., Cho, Y., & Kim, A. (2018). Exposure Control Using Bayesian Optimization Based on Entropy Weighted Image Gradient. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 857–864. https://doi.org/10.1109/ICRA.2018.8462881

Kiratiratanapruk, K., & Sinthupinyo, W. (2011). Color and texture for corn seed classification by machine vision. *2011 International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS)*, 1–5. https://doi.org/10.1109/ISPACS.2011.6146100

Kotkar, V. A., & Gharde, S. S. (2013). *REVIEW OF VARIOUS IMAGE CONTRAST ENHANCEMENT TECHNIQUES*. *2*(7).

Li, C., Kang, X., Zhang, Z., & Ming, A. (2023). SWBNet: A Stable White Balance Network for sRGB Images. *Proceedings of the AAAI Conference on Artificial Intelligence*, *37*(1), 1278–1286. https://doi.org/10.1609/aaai.v37i1.25211

Li, H., Zhang, R., Zhou, W., Liu, X., Wang, K., Zhang, M., & Li, Q. (2023). A novel method for seed cotton color measurement based on machine vision technology. *Computers and Electronics in Agriculture*, *215*, 108381. https://doi.org/10.1016/j.compag.2023.108381

Litwiller, D. (2001). Ccd vs. Cmos. *Photonics Spectra*, *35*(1), 154–158.

Luo, M. R., Cui, G., & Rigg, B. (2001). The development of the CIE 2000 colour-difference formula: CIEDE2000. *Color Research & Application*, *26*(5), 340–350. https://doi.org/10.1002/col.1049

Maître, H. (2017). *From Photon to Pixel: The Digital Camera Handbook* (1st ed.). Wiley. https://doi.org/10.1002/9781119402442

Mancuso, M. (2001). *An Introduction to the Digital Still Camera Technology*. *2*(2).

Mosslah, A. A., & Abbas, A. H. (2023). An Analysis of Image Processing In Forestry and Agriculture Review. *IOP Conference Series: Earth and Environmental Science*, *1202*(1), 012003. https://doi.org/10.1088/1755-1315/1202/1/012003

Nakamura, J. (Ed.). (2017). *Image Sensors and Signal Processing for Digital Still Cameras*. CRC Press. https://books.google.lv/books?id=UY6QzgzgieYC

Nayar, S. (Director). (2021a, February 28). *Depth of Field*. https://youtu.be/v5OE90eVIXo?list=PL2zRqk16wsdr9X5rgF-d0pkzPdkHZ4KiT

Nayar, S. (Director). (2021b, February 28). *Image Formation using Lenses*. https://youtu.be/7LX-19v_9ns?list=PL2zRqk16wsdr9X5rgF-d0pkzPdkHZ4KiT

Nayar, S. (Director). (2021c, February 28). *Pinhole & Perspective Projection*. https://www.youtube.com/watch?v=_EhY31MSbNM

Posch, M. (2017). *Mastering C++ Multithreading: A comprehensive guide to developing effective multithreading applications in C++* (First published July 2017). Packt Publishing.

Ramanath, R., Snyder, W. E., Yoo, Y., & Drew, M. S. (2005). Color image processing pipeline. *IEEE Signal Processing Magazine*, *22*(1), 34–43. https://doi.org/10.1109/MSP.2005.1407713

ROS/Introduction—ROS Wiki. (n.d.). Retrieved April 10, 2024, from http://wiki.ros.org/ROS/Introduction

Rossi, D. L., Bonato, V., Marques, E., & Lima, J. M. G. P. D. B. (2011). A PID Controller Applied to the Gain Control of a CMOS Camera Using Reconfigurable Computing. *2011 International Conference on Reconfigurable Computing and FPGAs*, 141–145. https://doi.org/10.1109/ReConFig.2011.2

Sami, M., & Khaled, M. (2019). *MinaSGorgi/Color-Constancy* [Python]. https://github.com/MinaSGorgi/Color-Constancy

Savic, D., Ilin, Z. M., Savic, D., & Ilin, Z. M. (2022). Advantages of Growing Vegetable Crops in Modern Greenhouses. In *Vegetable Crops—Health Benefits and Cultivation*. IntechOpen. https://doi.org/10.5772/intechopen.101469

Sheikh, H. R., & Bovik, A. C. (2006). Image information and visual quality. *IEEE Transactions on Image Processing*, *15*(2), 430–444. https://doi.org/10.1109/TIP.2005.859378

Sheikh, H. R., Sabir, M. F., & Bovik, A. C. (2006). A Statistical Evaluation of Recent Full Reference Image Quality Assessment Algorithms. *IEEE Transactions on Image Processing*, *15*(11), 3440–3451. https://doi.org/10.1109/TIP.2006.881959

Shim, I., Lee, J.-Y., & Kweon, I. S. (2014). Auto-adjusting camera exposure for outdoor robotics using gradient information. *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1011–1017. https://doi.org/10.1109/IROS.2014.6942682

Tian, Z., Ma, W., Yang, Q., & Duan, F. (2022). Application status and challenges of machine vision in plant factory—A review. *Information Processing in Agriculture*, *9*(2), 195–211. https://doi.org/10.1016/j.inpa.2021.06.003

Torre, V., & Poggio, T. A. (1986). On Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-8*(2), 147–163. https://doi.org/10.1109/TPAMI.1986.4767769

Torres, J., & Menéndez, J. M. (2015). *Optimal camera exposure for video surveillance systems by predictive control of shutter speed, aperture, and gain* (N. Kehtarnavaz & M. F. Carlsohn, Eds.; p. 94000S). https://doi.org/10.1117/12.2083182

Van De Weijer, J., Gevers, T., & Gijsenij, A. (2007). Edge-Based Color Constancy. *IEEE Transactions on Image Processing*, *16*(9), 2207–2214. https://doi.org/10.1109/TIP.2007.901808

Wang, Z., & Bovik, A. C. (2006). *Modern Image Quality Assessment*.

Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, *13*(4), 600–612. https://doi.org/10.1109/TIP.2003.819861

Zapryanov, G., Ivanova, D., & Nikolova, I. (2012). *Automatic White Balance Algorithms forDigital StillCameras–a Comparative Study*.

Zhang, Q., Nie, Y., & Zheng, W.-S. (2019). Dual Illumination Estimation for Robust Exposure Correction. *Computer Graphics Forum*, *38*(7), 243–252. https://doi.org/10.1111/cgf.13833

Zhou Wang, & Bovik, A. C. (2002). A universal image quality index. *IEEE Signal Processing Letters*, *9*(3), 81–84. https://doi.org/10.1109/97.995823

APPENDIXES

Appendix 1

Decision parameters functions

```
float imageBr (const cv::Mat& hist, int pixels){
        float sum;
        for(int i = 0; i<256; i++){
          float pixelValue = hist.at<uchar>(0,i);
          sum += i*pixelValue;
        }
        return sum/pixels;
}
float imageCr (const cv::Mat& image, float imageBr, int pixels){
        cv::Size image_size = image.size();
        float sum;
        for(int r = 0; r < image_size.width; r++){
          for(int c = 0; c < image_size.height; c++){
            int pixelValue = image.at<uchar>(c,r);
            float deviation = pixelValue - imageBr;
            sum += deviation * deviation;
          }
        }
        float variance = sum / pixels;
        return sqrt(variance);
}
float findWSL(std::vector<float>& cumHist, int oer){
        return cumHist[255-oer];
}
float findBSL(std::vector<float>& cumHist, int nr){
        return cumHist[nr];
}

float idwsl = cumhist[findIndex(cumhist,1-ST_)];
float idbsl = cumhist[findIndex(cumhist,ST_)];
```

The findIndex() is a simple binary search function that returns the index of the target value.

Appendix 2

Gain set function

```
bool setGainValue(const double value) {
VmbErrorType errType;
if (camera_ == nullptr){
    std::cout << "camera_ is nullptr" << value <<std::endl;

    return false;
}
if( VmbErrorSuccess == camera_->GetFeatureByName("Gain", feature_ ))
{
    errType = feature_->SetValue(value);
    if( errType != VmbErrorSuccess )
    {
        std::cout << "Error in set Gain's value= " << value << ". Error ID =  " << errType <<
std::endl;
        return false;
    }

    return true;

} else {
    std::cout << "AlliedCamera has NOT recognized the feature 'Gain'" << std::endl;
    return false;
  }
}
```

This function is based on the VimbaSDK API documentation.

Appendix 3

PID Controller

The following code is adopted from the Beauregard (2017):

```
class PID_Controlier {
//from http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/
  public:
  double Input, Output, Setpoint;
  double errSum, lastInput;
  double kp, ki, kd;

  double compute(){
     double error = Setpoint - Input;
     errSum += error;
     double dInput = (Input - lastInput);
     Output = kp * error + ki * errSum - kd * dInput;

     lastInput = Input;
     return(Output);
   }

  void reset(){
     errSum = 0.0;
     lastErr = 0.0;
     lastInput = 0.0;
   }

  PID_Controlier(double Kp, double Ki, double Kd){
     kp = Kp;
     ki = Ki;
     kd = Kd;
     errSum = 0.0;
  }
};
```

Appendix 4

Filtering saturated pixels

```
for (int channel = 0; channel < 3; ++channel) {
    cv::Mat deriv_channel = deriv_image[channel].clone();
    cv::Mat image_channel;
    cv::extractChannel(image, image_channel, channel);

    for (int row = 0; row < image_channel.rows; ++row) {
        for (int col = 0; col < image_channel.cols; ++col) {
            if (image_channel.at<uchar>(row, col) >= 255) {
                deriv_channel.at<double>(row, col) = 0.0;
            }
        }
    }

    deriv_image[channel] = deriv_channel;
}
```

This code part goes through the every channel of the input image and for every pixel that has a value bigger or equal to 255, it sets the same pixel in the *deriv_image* to 0.

Appendix 5

Histogram comparison on *dark natural* setting

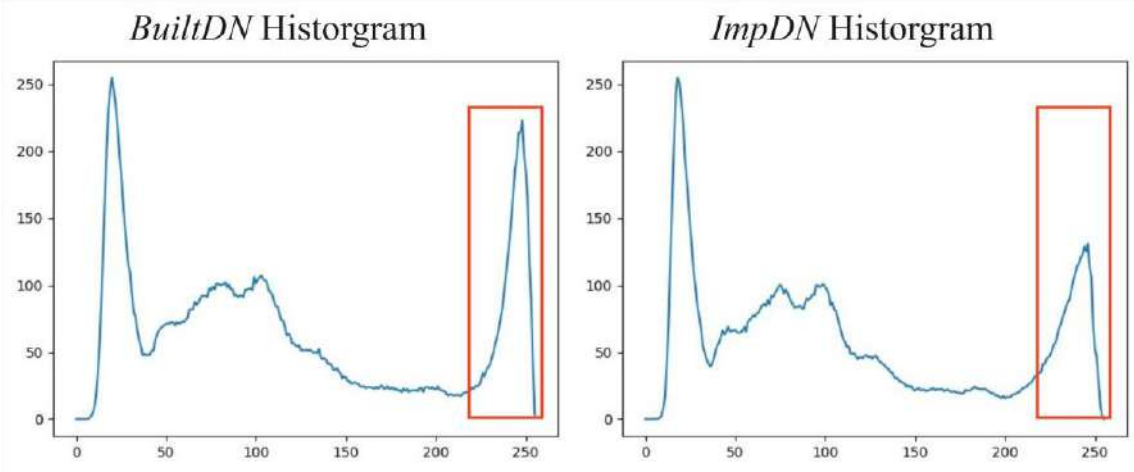


Figure 0.1 Histogram comparison of gain algorithms in *dark natural(DN)* setting

Similar to the *natural* setting, implemented algorithm decides on a lower gain value to decrease the overexposed areas.