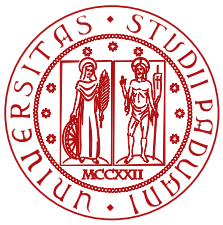


UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DEPARTMENT OF INFORMATION ENGINEERING

MASTER DEGREE IN CONTROL SYSTEMS ENGINEERING

Parameters control of a conveyor belt camera system to improve vision inspection performance in industrial applications

Supervisor
Prof. Cenedese Angelo

Master candidate
Savona Francesco

ACADEMIC YEAR 2023-2024

Graduation date 16 April 2024

Contents

1 Introduction

Industrial automation has brought about increased efficiency and precision in various sectors. Among the components that constitute an automated system, conveyor belts hold a significant position. They are integral to numerous industrial processes, facilitating the smooth transition and handling of goods.
In the context of quality control, conveyor belts work together with other automated systems such as cameras and sensors. These technologies monitor the products as they move along the conveyor belts, inspecting them for any defects or inconsistencies. This real-time monitoring and inspection ensure that only products meeting the set quality standards progress further in the production line, thereby maintaining high quality output and minimizing waste.

This thesis focuses on a specific application that involves the use of a Luxonis camera, specifically the OAK-D Lite and Pro models [23][21], positioned above a conveyor belt system. The use of Smart cameras, such as the aforementioned, offer significant advantages over traditional cameras. They process images in real-time, have advanced features like AI for tasks such as object recognition, and can be integrated with IoT devices. Their adaptability to varying conditions and efficient data management make them ideal for industrial use. Based on this, the primary objective of this setup is to study and implement an algorithm that enhances quality control performance. This includes providing a swift response to light variations and other disturbances, and importantly, controlling the conveyor belt speed to reduce the effects of motion blur for longer exposure times.

The conveyor belt system under consideration here is made by the BMTec [3] and consists of three cascading belts, each approximately $1.5\ m$ in length as can be seen in Figure 1a and 1b. A vibrating plate, in Figure 1c, is a common component in such systems, is used to position small objects like screws, nuts and washers onto the belt. The system is equipped with a control panel, as shown in Figure 1d, which provides comprehensive control over the conveyor belt parameters. This includes the ability to regulate the speed of each of the three conveyor belts individually. Additionally, it offers control over a lighting system that is currently in the process

of being integrated into the system.

(a) *(b)*

(c) *(d)*

Figure 1: BMTec's conveyor belt subject of the study

Industrial conveyor belts operate at varying speeds, in the case of study between $0.03\,m/s$ and $1.8\,m/s$. At these speeds, the exposure time must be significantly low to achieve a shape reconstruction error of, for example, one-tenth of a millimeter.

This requirement presents a unique challenge.

The relationship between conveyor speed (v), exposure time (t), and positional error (e) can be expressed as:

$$e = v \cdot t \tag{1}$$

For an error of $0.1\ mm$ ($0.0001\ m$) and the maximum conveyor speed of $1.8\ m/s$, the required exposure time would be:

$$t = \frac{e}{v} = \frac{0.0001}{1.8} \approx 0.000055\ s \approx 55\ \mu s \tag{2}$$

This calculation shows that even at the maximum speed of the conveyor belt, the required exposure time to maintain the positional error within $0.1\ mm$ is within the camera's exposure time range (10 to $33000\ \mu s$). However, this is quite close to the lower limit of the camera's capabilities, leaving little room for error. This underscores the need for a rapid response to changes in lighting conditions and other disturbances to maintain image quality and accuracy. It also highlights the precision required to address it.

Moreover, the environment where the conveyor belt operates is usually characterized by low light conditions. This typically necessitates a higher exposure time or an increase in the sensor's sensitivity. However, both these adjustments can lead to a significant reduction in image quality. This further emphasizes the need for an effective control algorithm that can maintain optimal image quality under these challenging conditions.

One important consideration of this project is to create a solution that can be easily implemented even on conveyor belts that are not predisposed to. While it would be possible to achieve better performance by creating an enclosure and maintaining a constant light source to avoid problems of shadows and under or over-exposures, the philosophy of the project is to obtain something that does not require too much hardware implementation. This means finding a balance or compromise between the ideal conditions for the camera and the practical conditions of the conveyor belt system.

In addition to the primary goal of this thesis, another crucial objective is to provide a quality parameter for each acquired image. This parameter serves as an indicator of the image's quality, which is fundamental in digital image processing. By ensuring good prepossessing and providing a quality parameter with each image, we can potentially decrease the complexity of the system as it won't have to deal with images of poor quality or disturbances.
Poor quality images could be due to various factors such as low resolution, high noise levels, poor lighting conditions, or motion blur. By controlling these factors, we can ensure that the images fed into the system are of high quality, thereby reducing the chances of errors or inaccuracies in the subsequent stages of processing. For instance, if the subsequent algorithm, which could be a neural network or a simpler object detection/classification algorithm using computer vision, produces some unexpected results, we have an additional parameter (the quality parameter) to verify if the results are reliable or not.

The goal of this thesis, therefore, is to study and implement an algorithm that can effectively control the parameters of the Luxonis cameras in response to varying conditions. By doing so, it aims to improve the performance of the conveyor belt system in applications such as product quality control and product quantity estimation. This endeavor, if successful, could significantly enhance the efficiency and accuracy of industrial automation processes. This approach, therefore, not only enhances the performance of the conveyor belt system but also ensures the reliability of the results produced by the system.

In conclusion, this thesis aims to strike a balance between ideal and practical conditions, and in doing so, hopes to contribute significantly to the field of industrial automation.

2 Cameras in industry

In the modern era, cameras have become an integral part of various industrial applications. They are used in a wide range of applications, from quality control and safety monitoring to process control and automation. With the advancement in technology, cameras have become more sophisticated and capable, making them an indispensable tool in the industry [1].

Cameras in the industry are primarily used for inspection and surveillance purposes [1]. They help in maintaining the quality of products, ensuring the safety of the workplace, and monitoring the efficiency of the production process. With the help of cameras, defects in products can be detected early, reducing waste and saving costs. Surveillance cameras also help in preventing accidents by monitoring the working conditions and alerting the concerned authorities in case of any discrepancies.

2.1 Importance of camera in industry

The importance of cameras in the industry cannot be overstated. They are used in a variety of applications, each serving a unique purpose. For instance, in the manufacturing industry, cameras are used for quality control. They inspect products for defects, ensuring that only high-quality products reach the customers. This not only improves customer satisfaction but also reduces the cost associated with defective products.

In addition to quality control, cameras are also used for safety monitoring. They monitor the working conditions in factories and alert the authorities in case of any safety hazards. This helps in preventing accidents and ensuring the safety of the workers.

Furthermore, cameras are used for process control. They monitor the production process and provide feedback to the control system. This allows the system to make necessary adjustments to the process, improving efficiency and reducing waste.

The advent of cameras, especially those equipped with advanced computer vision algorithms, has indeed brought about a paradigm shift in the field of robotics.

Acting as the ‘eyes’ for robots, these cameras capture high-resolution images or video feeds of their surroundings. This visual data has opened up new avenues for robots to interact with their environment in ways that were once thought to be beyond reach. Consider mobile robots, for instance. They can now recognize minute features, subtle changes, and variances in their surroundings. This ability has empowered them to navigate through challenging terrains, identify and avoid obstacles, and control items with remarkable precision.

In the realm of industrial processes, object detection and classification have become indispensable for maintaining quality control. Cameras are used to capture images of moving objects, which are then classified based on their size, color, and other properties. This technology has found its application in autonomous driving technology as well. Here, it works together with image recognition software to ensure the safe operation of vehicles, detect traffic, create 3D maps, and enable navigation without the need for a driver.

The importance of spatial data analysis, often facilitated by the use of cameras, cannot be overstated in various industries. It has proven to be a game-changer in reducing costs, boosting business revenue, and significantly enhancing machine productivity. Moreover, it has made the work process more efficient.

The advancements and applications elucidated above are integral components of Industry 4.0 and 5.0 that has revolutionized the industrial sector with the integration of digital technologies.

In Industry 4.0, cameras play a crucial role in enabling automation. They are used in conjunction with machine learning algorithms for tasks such as object detection, defect identification, and process monitoring. The data captured by the cameras is processed and analyzed to make informed decisions, thereby reducing human intervention and increasing efficiency.

Industry 5.0 takes the use of cameras a step further by integrating them into a hyper-automated environment. In this setup, cameras are not just passive data collectors but active participants in the industrial process. They are connected to a network of devices and systems that can communicate and make decisions in real-time. This allows for more complex and adaptive industrial processes.

Hyperautomation is a key concept in Industry 5.0. It involves the use of advanced technologies such as Artificial Intelligence (AI), Machine Learning (ML), and Internet of Things (IoT) to automate processes to a degree that was not possible in Industry 4.0. In the context of cameras, hyperautomation could involve real-time image processing, predictive maintenance of camera systems, and adaptive control of camera parameters based on environmental conditions.

While cameras have been an integral part of Industry 4.0, their role is significantly enhanced in Industry 5.0 with the advent of hyperautomation. As we move towards more connected and intelligent industrial systems, the role of cameras as data collectors and decision-makers will continue to evolve.

2.2 Types of cameras for industrial applications

In the realm of industrial applications, a variety of cameras are employed, each bringing its unique set of features and benefits to the Table.

Area Scan Cameras, for instance, capture a two-dimensional image of the entire field of view in one go. This ability makes them a popular choice for applications such as quality control, sorting, and packaging where a comprehensive view of the scene is required [1].

On the other hand, **Line Scan Cameras** work differently. They capture a single line of pixels at a time, gradually building a two-dimensional image line by line. This unique capability makes them ideal for inspecting cylindrical objects and web materials, where a continuous stream of data is needed [1].

In addition there are also the **3D Cameras**. These cameras capture three-dimensional images of the subject, providing valuable information about the shape, volume, and surface profile. This depth of data finds its use in applications like robot guidance, quality control, and logistics, where understanding the spatial relationships between objects is crucial [1].

Infrared (Thermal) Cameras offer a different perspective altogether. They capture images based on the heat, or infrared radiation, emitted by objects. This makes them useful for applications like predictive maintenance, safety monitoring, and process control, where temperature variations are key indicators. Their ability to

capture images in low light or darkness also makes them invaluable in security and wildlife observation [1].

X-ray cameras, on the other hand, are used in industrial radiography to inspect materials and components, locate and quantify defects, and detect degradation in material properties. They also play a crucial role in maintaining security at airports, ports, and borders [1].

In essence, the type of camera chosen for an application depends on the specific requirements of the task at hand. Each type of camera, with its unique capabilities, contributes to the overall efficiency and effectiveness of industrial processes. However, it's important to note that these advanced cameras come with a high price tag. Despite the cost, the efficiency and precision they offer often justify the investment, as they can significantly enhance the quality and productivity of industrial processes.

3 Luxonis cameras

Luxonis [16], a company based in Colorado, is at the forefront of the robotic vision industry. Their mission is twofold: to democratize access to robotics and to enhance engineering efficiency on a global scale. To achieve these goals, Luxonis has developed camera systems that seamlessly integrate artificial intelligence (AI), computer vision, and image processing.

Their flagship product, the Depth-AI camera, is a testament to their innovative approach. These cameras are designed to deliver high-resolution images, depth vision, and on-chip machine learning. The integration of AI, computer vision, and image processing directly into the camera system significantly reduces the computational load on the robot's computer, making the system more efficient.

This chapter aims to provide a comprehensive overview of the existing cameras and the possibilities offered by Luxonis. While the primary focus of the thesis is on the prepossessing part, it's important to note that the capabilities of Luxonis cameras extend beyond this. The advanced features of these cameras also open up possibilities for implementing sophisticated algorithms after the prepossessing stage.

3.1 Hardware specifications

All Luxonis OAK cameras are equipped with a variety of on-device features, including AI inference, computer vision (CV) functions, video encoding, Python script execution, and more. The OAK-D family of devices stands out with its built-in stereo depth camera pair. This enables stereo depth perception, seamlessly integrated with AI capabilities. As a result, these devices can execute Spatial AI tasks, such as generating 3D spatial coordinates (XYZ) for identified objects and features or assigning a class to each depth point. The Pro version of OAK cameras takes it a step further by incorporating an on-board IR laser dot projector for active stereo and an IR illumination LED for enhanced night vision capabilities.

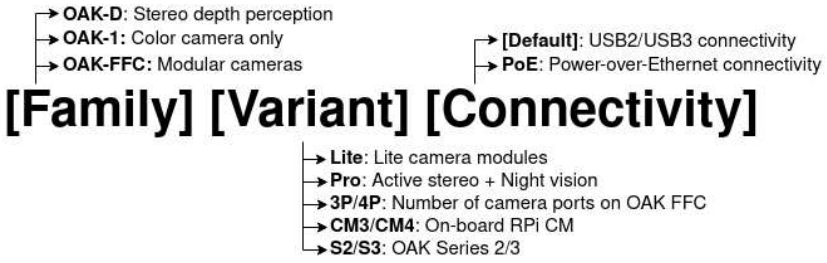


Figure 2: Luxonis cameras nomenclature system - Figure taken from the site [18]

3.1.1 Luxonis family

Luxonis company offers lots of different affordable cameras, each with specific characteristic. The name of the camera specifies its main features, as can be seen in Figure 2.

The Luxonis camera is compose by three main type of camera with different variants. The OAK-D family of cameras consists of three image sensors, one RGB in the centre and two B/W at the sides, which allow the depth of the scene to be calculated with good accuracy under optimal conditions. The OAK-D family comes with different version based on user's requirements, in fact there are cameras that are specific for long or short range depth capabilities (respectively OAK-D LR and OAK-D SR), camera with IR-dot projector and flood illumination (PRO versions) and cameras with smaller or bigger image sensor (respectively OAK-D lite and OAK-D S2/PRO).

The OAK-1 models are cameras that have just one sensor, meaning that they don't provide stereo depth perception but incorporate most of the other features given by the robotic vision core 2 (RVC2).

The OAK FFC line provides an excellent platform for prototype flexibility. Given the modular design of the cameras, users have the flexibility to position them at various stereo baselines. However, this adaptability requires careful consideration: one must decide on the method and location for mounting, followed by the essential step of performing a stereo calibration post-mounting. It's important to note that these cameras do not offer a "plug and play" experience like some other options. Instead, they are better suited for applications that demand a customized approach to mounting, specific baseline configurations, or unique camera orientations.

Most of the Luxonis cameras come with a Power over Ethernet (PoE) version that directs the product towards mainly industrial applications. The main advantages of this characteristic are that these cameras are more robust and above all have the capabilities of work in standalone mode due to the presence of a flash memory inside, thus meaning that they don't need to be connected to a host computer. The host can be also a simple Raspberry, in fact the OAK CM models have an on-board Raspberry Pi Compute Module [17], which is technically the host computer of the OAK SoM, but the device still runs on its own.
All the main features of the Luxonis cameras described above are summarised in Table 1.

3.1.2 Robotics Vision Core 2

The Robotics Vision Core 2 (RVC2), the second generation of Luxonis's Robotics Vision Core, is a powerful core that provides 4 Tera Operations Per Second (TOPS) of processing power, with 1.4 TOPS dedicated to AI tasks. This allows the RVC2 to handle complex AI models, even those that are custom-designed, albeit these models need to be converted to a compatible format first.
The RVC2 supports various encoding formats, including H.264, H.265, and MJPEG, and can handle 4K resolution at 30 frames per second (FPS) or 1080P at 60 FPS. This ensures the images captured by the camera are of high quality and detail.
In terms of computer vision tasks, the RVC2 is capable of performing operations such as resizing and cropping images, detecting edges, and tracking features in real-time. It also supports stereo depth perception, which allows the camera to perceive the 3D world around it. This is achieved through a combination of filtering, post-processing, and RGB-depth alignment.
One of the key features of the RVC2 is its object tracking capability. It can perform both 2D and 3D tracking using the ObjectTracker node. This means that the camera can keep track of objects as they move in space, which is crucial for applications like autonomous navigation and object detection.
Furthermore, the RVC2 provides performance metrics for various models. This allows for the evaluation of an AI model's performance based on factors like FLOPs

OAK model	AI, CV, tracking, encoding, ...	Spatial data	PoE, IP67	Active stereo, night vision	Standalone mode	Coprocessor	Camera
OAK-1, OAK-1 lite	✓						
OAK-D, OAK-D lite	✓	✓					
OAK-D S2	✓	✓					Wide FOV option
OAK-D pro	✓	✓		✓			Wide FOV option
OAK-1-PoE	✓		✓		✓		
OAK-D-PoE	✓	✓	✓		✓		
OAK-D S2 PoE	✓	✓	✓		✓		Wide FOV option
OAK-D pro PoE	✓	✓	✓	✓	✓		Wide FOV option
OAK-D-CM3, OAK-D-CM4	✓	✓			✓	RPi CM	
OAK-D-CM4 PoE	✓	✓	✓		✓	RPi CM	
OAK-FFC-3P, OAK-FFC-3P-OG	✓	✓					Custom

Table 1: Comparative main features

and parameters, which can be very useful when optimizing a model for better performance.

Table 2 provides a comprehensive overview of the features of each camera

model, demonstrating the versatility and power of the RVC2. The models listed in Table 2 were compiled for 8 shaves and were using 2 NN inference threads. The latency includes getting results from the device over USB3. Furthermore, 5 iterations were run for each model and the Frames Per Second (FPS) was calculated as an average. These performance tests were conducted by Luxonis, providing a reliable evaluation of the performance of various models under different conditions.

Model name	**Image size**	**FPS**	**Latency [ms]**
MobileOne S0	224x224	165.5	11.1
Resnet18	224x225	94.8	19.7
DeepLab V3	256x256	36.5	48.1
DeepLab V3	513x513	6.3	253.1
YoloV6n R2	416x416	65.5	29.3
YoloV6n R2	640x640	29.3	66.4
YoloV6t R2	416x416	35.8	54.1
YoloV6t R2	640x640	14.2	133.6
YoloV6m R2	416x416	8.6	190.2
YoloV7t	416x416	46.7	37.6
YoloV7t	640x640	17.8	97.0
YoloV8n	416x416	31.3	56.9
YoloV8n	640x640	14.3	123.6
YoloV8s	416x416	15.2	111.9
YoloV8m	416x416	6.0	273.8

Table 2: Performance Metrics for Various Models

3.1.3 Standalone version

The Standalone mode refers to a configuration where the camera initiates the flashed application upon receiving power, operating independently without the need for a specific host computer connection. This mode proves valuable in scenarios where the camera remains stationary, performing tasks such as environmental inspection and analytics, including people/vehicle counting, License Plate Recognition (LPR),

fall detection, and more.
This mode offers increased stability, particularly in the face of potential instabilities like networking issues that could disrupt the connection between the camera and a host computer. In the standalone mode, the application automatically restarts in the event of such issues.
It's important to note that standalone mode is exclusive to OAK models equipped with onboard flash memory, such as OAK POE and OAK IOT camera models. Moreover it's worth mentioning that if a computer is already onboard (e.g., on a robot or drone), Standalone mode may introduce unnecessary complexity.

For communication with the external environment (e.g., a server), POE cameras can employ the Script node to send/receive networking packets, supporting protocols such as TCP, UDP, HTTP, and MQTT. These models are also compatible with the Robot Operating System (ROS) environment, enabling seamless integration and operation within ROS-based applications, without using a host device.

3.1.4 Technical comparison OAK-D lite/PRO

Figure 3: Luxonis OAK-D lite

Figure 4: Luxonis OAK-D pro

The cameras that are the subject of this technical specifications comparison are the OAK-D lite and the OAK-D pro, shown in Figures 3 and 4 respectively. Both lite and pro versions are based the OAK-D camera that is one of the first series of camera made by Luxonis. Since also this camera is based on the RVC2, they shared approximately the same performances with regard to AI, CV, tracking, encoding, etc. as it is in fact shown in Table 1. To be more specific the OAK-D pro camera is based on the second series of the OAK-D, so the OAK-D S2 but, as it's reported in

the official website [23], the main differences between the first and second version are related to the enclosure (smaller and lighter in the series 2) and in the type of connection that in the newest camera is composed by a single USB-C connector for both power and communication.

The OAK-D pro version is the OAK-D S2 but with the IR laser dot projector and IR illumination LED that make a huge difference in depth perception (thanks to the active stereo function) and in poor lighting conditions.

The OAK-D lite is an upgrade of the OAK-D camera but it is equipped with smaller and cheaper image sensors that don't significantly reduce the performance in the context of AI, CV, tracking, encoding, etc, but make difference in the stereo perception at high distances due to lower resolution cameras. Table 3 shows the main hardware differences among the subjects of this report.

	Color camera		**Stereo pair**	
Camera Specs	**OAK-D Lite**	**OAK-D Pro**	**OAK-D Lite**	**OAK-D Pro**
Sensor	IMX214 (PY014 AF, PY114 FF)	IMX378 (PY004 AF, PY052 FF)	OV7251 (PY013)	OV9282 (PY091 BP @ 940nm)
DFOV / HFOV / VFOV	81° / 69° / 54°	81° / 69° / 55°	86° / 73° / 58°	89° / 80° / 55°
Resolution	13MP (4208x3120)	12MP (4056x3040)	480P (640x480)	1MP (1280x800)
Focus	AF: 8cm - ∞ FF: 50cm - ∞	AF: 8cm - ∞ FF: 50cm - ∞	FF: 6.5cm - ∞	FF: 19.6cm - ∞
Max framerate	35 FPS	60 FPS	120 FPS	120 FPS
F-number	2.2 ± 5%	1.8 ± 5%	2.0 ± 5%	2.0 ± 5%
Lens size	1/3.1 inch	1/2.3 inch	1/7 inch	1/4 inch
Effective Focal Length	3.37mm	4.81mm	1.3mm	2.35mm
Pixel size	1.12µm x 1.12µm	1.55µm x 1.55µm	3µm x 3µm	3µm x 3µm

Table 3: Combined Camera Specifications for OAK-D Lite and OAK-D Pro.

3.2 Software integration

This chapter delves into the software integration aspect of Luxonis cameras. One of the key strengths of Luxonis cameras is their compatibility with popular programming languages such as Python and C++. This compatibility is facilitated through the use of Application Programming Interfaces (APIs) and Software Development Kits (SDKs).

APIs and SDKs serve as the bridge between the hardware (Luxonis cameras in this case) and the software. They provide a set of tools, definitions, and protocols that enable the interaction between the software and the hardware.

An API, or Application Programming Interface, is a set of rules and protocols for building and interacting with software applications. APIs simplify the process of integrating different software components and allow them to communicate effectively. When it comes to controlling Luxonis cameras, APIs offer a more precise control over the camera parameters and functions.

On the other hand, an SDK, or Software Development Kit, is a collection of software tools and programs used by developers to create applications for specific platforms. SDKs for Luxonis cameras typically include APIs, programming tools, and other utilities that assist in the development process.

While both APIs and SDKs play crucial roles in software integration, for more precise control over the Luxonis cameras, it is often recommended to use APIs. This is because APIs provide a more direct interface to the camera's functions and parameters, allowing for a higher degree of customization and control.

3.2.1 depthai libraries

The **DepthAI SDK** is a Python package built on top of the depthai-python API library that improves ease of use when developing apps for OAK devices. It provides an abstraction of the DepthAI API library, containing convenience classes and functions that assist in common tasks while using the DepthAI API.

The **DepthAI API** allows users to connect to, configure, and communicate with their OAK devices. It supports both Python and C++. The API is designed around

the concept of nodes and messages.
In particular the nodes are the building blocks when populating the pipeline. Each node provides a specific functionality on the DepthAI, a set of configurable properties, and inputs/outputs. After creating a node on a pipeline, you can configure it as desired and link it to other nodes.
Linking nodes is achieved through the use of messages. Messages are the only way nodes communicate with each other. They are sent between linked nodes, and the data flows from one node to another in a defined sequence while different operations are performed on the data at each node.
A DepthAI message can be created either on the device by a node automatically, or manually inside the Script node. It can also be created on a host computer and sent to the device via the XLinkIn node.

Let's consider a simple example. Suppose there is a system composed by two nodes, node A and B, and the information (message) passes data from Node A to Node B. This can be represented as:

$$\text{Node A} \xrightarrow{\text{Message}} \text{Node B}$$

In this example, Node A performs some operation and generates a message. This message is then passed to Node B through a link. Node B receives the message, performs its operation, and the process continues.
The complete example represented the OAK device connected to the host using APIs is shown in Figure 5. As can be seen in the host side there are more ways to interact with the camera, the first (depthai demo) is the simplest and less customizable since it is just an executable file; with it is possible to set basic and preconfigured features, The second that stays in the middle is the aforementioned SDK way and finally the more sophisticated APIs.

This system of nodes and messages allows for a flexible and powerful pipeline for processing data on OAK devices. It enables developers to build complex data processing pipelines with ease, leveraging the power of the DepthAI platform.

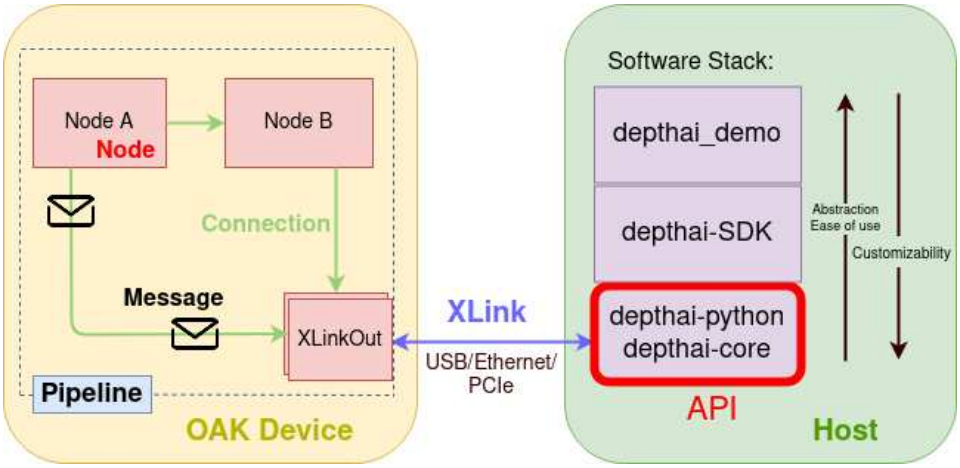


Figure 5: API configuration diagram - Figure taken from the site [18]

3.3 Applications

The high versatility of the OAK-D platform allows this hardware to be integrated into various applications, in view of the fact that both cameras are capable of performing built-in and custom computer vision routines and run built in and custom Neural Network models. Moreover considering the cost of these two cameras when compared to some of the high end cameras used in industrial applications and the fact that the performances offered are of good quality, it is fair to state that these devices can be used in both the prototyping phase and a definitive version of a project.

In addition to the features previously mentioned both cameras present either a 9-axis inertial measurement unit that combines accelerometer, gyroscope and magnetometer (for the OAK-D pro) or a 6-axis inertial measuring unit with gyroscope and accelerometer (for the OAK-D lite camera), thus suggesting the possibility of using such devices for eye-in-hand applications with a robotic manipulator.

4 The Camera Response Function

The Camera Response Function (CRF) is a critical component in digital imaging, representing the relationship between the scene irradiance and the image brightness. This function essentially describes how a camera responds to different light intensities in a scene. In this chapter are described the principles behind this concept. Additionally, since each image sensor has its own response and is essential for the algorithms implemented in this project, the method used to derive the CRF function is illustrated.

4.1 Operating principles of cameras

A camera operates on the fundamental principle of capturing and recording light to produce images. At its core, a camera consists of a lens, an aperture, and a light-sensitive surface.

Light enters the camera through the lens, which focuses the incoming light onto the light-sensitive surface (see Figure 6). The aperture, a small opening within the lens, controls the amount of light that enters into the camera. By adjusting the size of the aperture, the algorithm can control the exposure of the image, determining how much light reaches the digital image sensor.

In the case of this study, the image sensor doesn't have a shutter since it operates differently. Instead, the image sensor continuously accumulates light during the exposure process, and the exposure time is controlled electronically. This allows precise control of exposure duration without the need for a physical shutter. Additionally, the aperture is fixed, as mentioned in Table 1, further simplifying the exposure control process. This fixed aperture ensures consistent light transmission, avoiding use of mechanical components and so reducing size and possible failures. Once the light enters the camera and passes through the aperture, it is captured by the light-sensitive surface. In digital cameras, this medium is a sensor composed of millions of light-sensitive pixels that convert the incoming light into digital signals. These signals are then processed and stored as digital image files.

In addition to controlling the amount of light that enters the camera, it is possible

to adjust the camera's sensitivity to light using the ISO setting. ISO measures the sensor's sensitivity to light, with higher ISO values making the sensor more sensitive and able to capture images in low-light conditions. However, increasing the ISO can also introduce digital noise, affecting the quality of the image. Furthermore, depending on the camera manufacturer and the control system, ISO can be replaced by gain, which represents an increase/decrease in sensitivity in dB. The two parameters are equivalent but are processed differently.

In an ordinary photographic camera, the lens has a fixed focal length. Focusing at various distances is achieved by varying the distance between the lens and the imaging plane [6], where the sensitivity surface is located, see Figure 6.

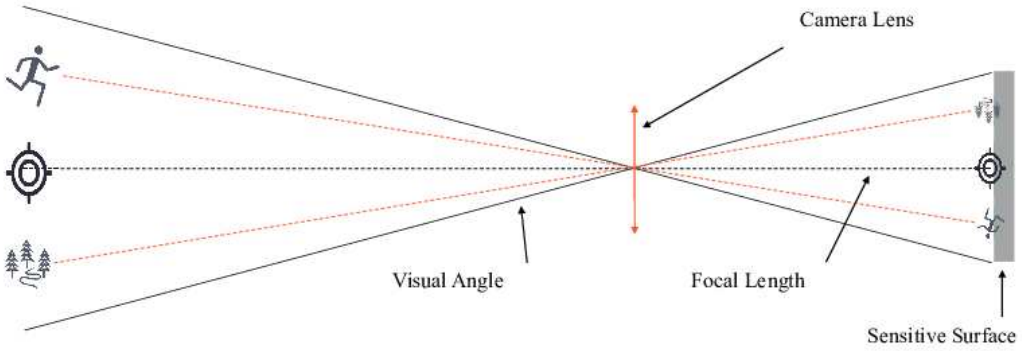


Figure 6: Simple pinhole camera model

In the human eye, the converse is true; the distance between the center of the lens and the imaging sensor (the retina) is fixed, and the focal length needed to achieve proper focus is obtained by varying the shape of the lens. This adjustment is accomplished by the ciliary body, allowing for a range of focal lengths from approximately 14 mm to 17 mm. The retina, particularly the fovea, primarily receives the focused retinal image, enabling perception through the excitation of light receptors and subsequent interpretation by the brain [6] (see Figure 7).

Brightness adaptation and discrimination are crucial aspects of the human visual system when processing digital images. The eye can adapt to a vast range of light intensity levels, from the scotopic threshold to the glare limit, on the order of 10^{10}. Subjective brightness perception follows a logarithmic function concerning incident light intensity. The visual system achieves this wide adaptation range by

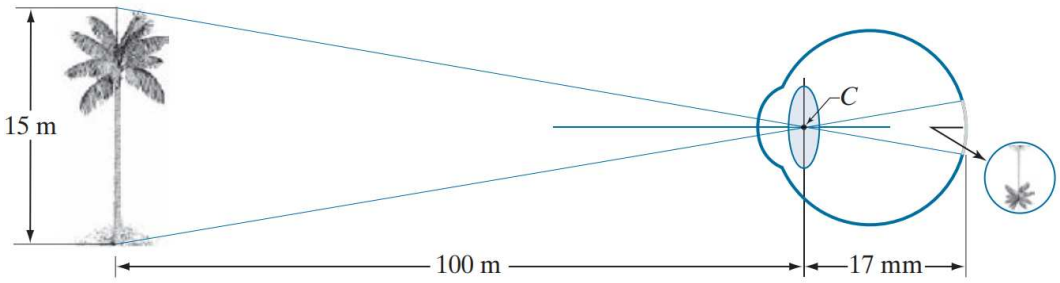


Figure 7: Graphical representation of the eye looking at a palm tree. Point C is the focal center of the lens - Figure taken from the book [6]

changing its overall sensitivity, known as brightness adaptation. However, the eye can only discriminate a relatively small range of distinct intensity levels simultaneously, corresponding to the current brightness adaptation level. This adaptation level determines the range of subjective brightness perception, with stimuli below a certain threshold perceived as indistinguishable blacks [6], some example of brightness sensations are represented in Figure 8.

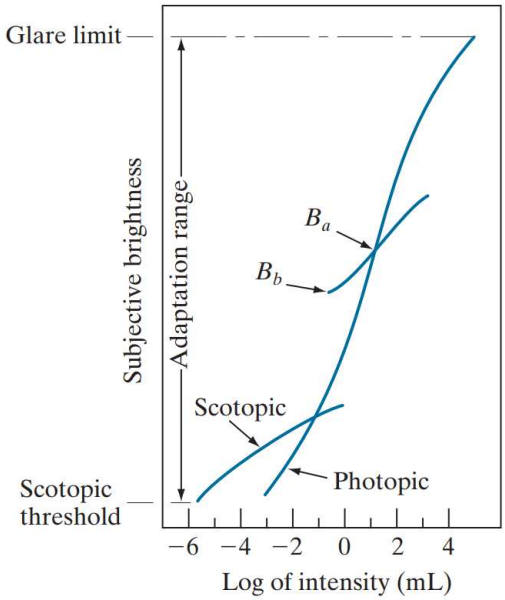


Figure 8: Range of subjective brightness sensations - Figure taken from the book [6]

In digital photography, the Camera Response Function (CRF) and the ISO value serve analogous functions to the brightness adaptation mechanism in the human vi-

sual system. The CRF maps accumulated sensor irradiance to pixel intensities in the final image, similar to how the eye adjusts its sensitivity to light to perceive brightness levels. This adjustment is achieved through a logarithmic function, ensuring a smooth transition between different brightness levels in the image. Similarly, changing the ISO value in digital photography allows the camera to adapt to varying levels of brightness, analogous to the eye's sensitivity adjustment. However, increasing the ISO value can introduce noise and compromise image quality, similar to the eye's limited range of discrimination.

4.2 Image formation process and camera response function

In the process of digital image formation, the radiance L of a scene point is captured by a camera sensor element. If the radiance received by a moving observer is independent of the observer's viewing angle, the scene point exhibits Lambertian reflectance behavior.

The total amount of energy received at a sensor location $\mathbf{x}$ per unit time is termed as irradiance $E(\mathbf{x})$ and it is expressed as (W/m^2). However, for most cameras, a radiometric fall-off of pixel intensities occurs towards the image borders due to vignetting effects. This vignetting effect is represented by a vignetting factor $V : \Omega \rightarrow [0, 1]$, dependent on the spatial location $\mathbf{x}$ of the image sensor [2]. Thus, the irradiance $E(\mathbf{x})$ can be obtained by multiplying the scene point's radiance with the vignetting factor:

$$E(\mathbf{x}) = V(\mathbf{x})L \tag{3}$$

When capturing an image, the sensor irradiance is integrated over a time window specified by the camera's exposure time Δt, resulting in accumulated irradiance X:

$$X(\mathbf{x}) = E(\mathbf{x})\Delta t \tag{4}$$

The accumulated irradiance $X(\mathbf{x})$ is then mapped by the camera response function (CRF) $f : \mathbb{R} \rightarrow [0, 255]$ to an image output intensity. For real cameras, the input of the CRF is limited by the camera's dynamic range. If the accumulated irradiance falls outside the dynamic range, the scene point is under or overexposed, resulting in pixel values of 0 or 255, respectively.

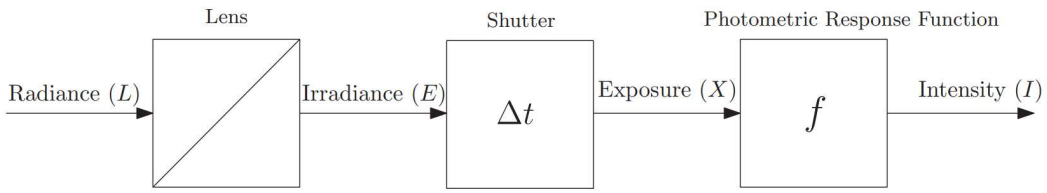


Figure 9: Image acquisition process - Figure taken from the paper [30]

The entire image formation process, mapping a scene point's radiance L to an image output intensity I, is shown in Figure 9 and can be compactly written as:

$$I = f\left(V(\mathbf{x})L\Delta t\right) \tag{5}$$

In this study, the effect of vignetting will not be considered due to simplicity and technical difficulties in deriving the $V(\mathbf{x})$ function for the cameras taken into consideration, also given the scarce documentation in this regard.

The function $f(\cdot)$ is invertible, ensuring that intensity increases monotonically with exposure [30]. For convenience, the inverse response function is defined as:

$$g = \ln f^{-1} \tag{6}$$

which allows us to rewrite the equation 5 as:

$$\tilde{g}(I) = \ln L + \ln \Delta t \tag{7}$$

As mentioned earlier, in digital images, intensities are represented by discrete values within a range $\{0, 1, ..., I_{\text{max}}\}$. Consequently, the inverse response function $\tilde{g}$ is constrained to take values $g(k)$, where k ranges from 0 to I_{max}. These specific values are determined through the analysis of images captured from a static scene under varying exposure times, as elaborated in Chapter 4.3 and explained in [4].

4.3 Estimating CRF using captured images

The algorithm summarised below is used to calculate the CRF function from a set of specific images. The study was first presented by Paul E. Debevec and Jitendra Malik in 1997 [4].

The algorithm is based on the concept of reciprocity in the image. Reciprocity in this context implies that only the product $L \cdot \Delta t$ is important, therefore if this product remains constant, the values of L or Δt can be varied without changing the result.

The input for the algorithm consists of N images obtained with a duration Δt_j where j corresponds to the respective image. In the paper is assumed that the scene is static and that this process is completed quickly enough that lighting changes can be safely ignored. Consequently, the irradiance values L_i for each pixel i are considered constant. Pixel values will be denoted by I_{ij} where i serves as a spatial index over pixels. The starting formula 8 is an extension of equation 7, with conditions added for each pixel i and for each input image j.

$$g(I_{ij}) = \ln L_i + \ln \Delta t_j \tag{8}$$

The key point of the algorithm is to recover unknown elements g and L_i from the known parameters I_{ij} and Δt_j. To do so the basic idea is to minimize the least-squared error obtained from equation 8, by finding the $I_{max} - I_{min} + 1$ values of $g(I)$ and the N values of lnL_i that minimize the quadratic objective function 9 [4].

$$O = \sum_{i=1}^{N} \sum_{j=1}^{P} [g(I_{ij}) - \ln L_i - \ln \Delta t_j]^2 + \lambda \sum_{I=I_{\min}+1}^{I_{\max}-1} g''(I)^2 \tag{9}$$

The first term in the equation ensures that the solution satisfies the set of equations arising from equation 8 in a least squares sense.

The second term acts as a smoothness constraint on the sum of squared values of the second derivative of g, ensuring that the function g is smooth. In this discrete setting, the expression 10 is utilize to approximate the second derivative. The parameter λ should be chosen appropriately based on the expected amount of noise present in the measurements of I_{ij}. To solve the problem the singular value decomposition is used [4].

$$g''(I) = g(I-1) - 2g(I) + g(I+1) \tag{10}$$

As explained in more detail in the paper, the solution to the minimization problem is affected by a bias factor that can scale the result without breaking the constraints. To mitigate this, an additional constraint has been added, assuming that

$g(I_{\text{mid}}) = 0$. This constraint is visible in Figure 12, where it can be observed that the intensity value around 127 corresponds to 0.

Finally, it has been added another constraint since to achieve a better fit in the solution, it's beneficial to anticipate the fundamental shape of the response function. Given that $g(I)$ typically exhibits a steep slope near I_{min} and I_{max}, it has been anticipated that $g(I)$ will be less smooth and fit the data less accurately near these extremes. To do so the function 11 has been integrated in the minimization function 9, obtaining 12.

$$w(I) = \begin{cases} I - I_{\text{min}} & \text{for } I \leq \frac{1}{2}(I_{\text{min}} + I_{\text{max}}) \\ I_{\text{max}} - I & \text{for } I > \frac{1}{2}(I_{\text{min}} + I_{\text{max}}) \end{cases} \tag{11}$$

$$O = \sum_{i=1}^{N} \sum_{j=1}^{P} w(I_{ij})[g(I_{ij}) - lnL_i - ln\Delta t]^2 + \lambda \sum_{I=I_{\text{min}}+1}^{I_{\text{max}-1}} [w(I)g''(I)]^2 \tag{12}$$

After the minimization process the function $g(I)$ and the values lnL are obtained, ready to be used in the subsequent processes.

To get a more precise and complete description refer to [4].

4.3.1 Image capture for CRF estimation

In order to estimate the inverse camera response function $f(\cdot)$, a series of images were acquired using a simple code developed in C++, which facilitated sequential adjustments of exposure and ISO values of the camera. The Luxonis Oak-D Lite camera was utilized for this purpose since it has similar performance to the pro version in this contest. The algorithm systematically adjusted exposure and ISO values across the entire available range, specifically varying ISO from 100 to 1600 and exposure time from 10 μs to 33000 μs, reaching the maximum frame rate of 30 fps. Subsequently, the resulting files were converted to grayscale and saved as .png images.

To ensure consistency and minimize variations in the captured images, the camera was securely mounted on a tripod. This setup effectively eliminated vibrations or movements during the image acquisition process, ensuring the stability of the captured frames, necessary for the estimation process.

Following the acquisition process, four images were selected based on specific criteria. Firstly, the chosen images exhibited uniform differences in exposure between them while maintaining identical ISO values. This selection criterion aimed to provide a diverse set of images for subsequent analysis and processing.
Additionally, the selected images were curated to ensure that a significant portion of pixels covered the entire range of values from 0 to 255.

This approach, as described in Paul E. Debevec and Jitendra Malik's paper [4], ensures comprehensive coverage of irradiance levels, facilitating accurate analysis and comparison.

The selected images are shown in Figure 10 with value of ISO 1000 and respectively exposure time of 4 ms, 13 ms, 23 ms, 33 ms.

(a) Δt: 4 ms, ISO: 1000

(b) Δt: 13 ms, ISO: 1000

(c) Δt: 23 ms, ISO: 1000

(d) Δt: 33 ms, ISO: 1000

Figure 10: Images captured for CRF estimation

4.3.2 CRF estimation

The development involved the creation of a C++ code for estimating the Camera Response Function (CRF) using the OpenCV library. Initially, the set of images choosen in the previous step was uploaded.

Subsequently, the images were aligned using the *cv::alignMTB* class from OpenCV to ensure accurate pixel-to-pixel correspondence. Although minimal camera motion occurred during capture, precise alignment was necessary for quality assurance.
Following alignment, the CRF was estimated using the *cv::CalibrateDebevec* class from OpenCV. The result was then saved in a .csv file in order to be used in the following steps. In light of the fact that the *cv::CalibrateDebevec* class returns the equivalent of f^{-1} as discussed in Chapter 4.2, and given the necessity of the function g for subsequent algorithms, a MATLAB script was employed to obtain the function g and perform other necessary processing steps. The resulting function g specific for the Luxonis OAK-D lite is represented in Figure 11.

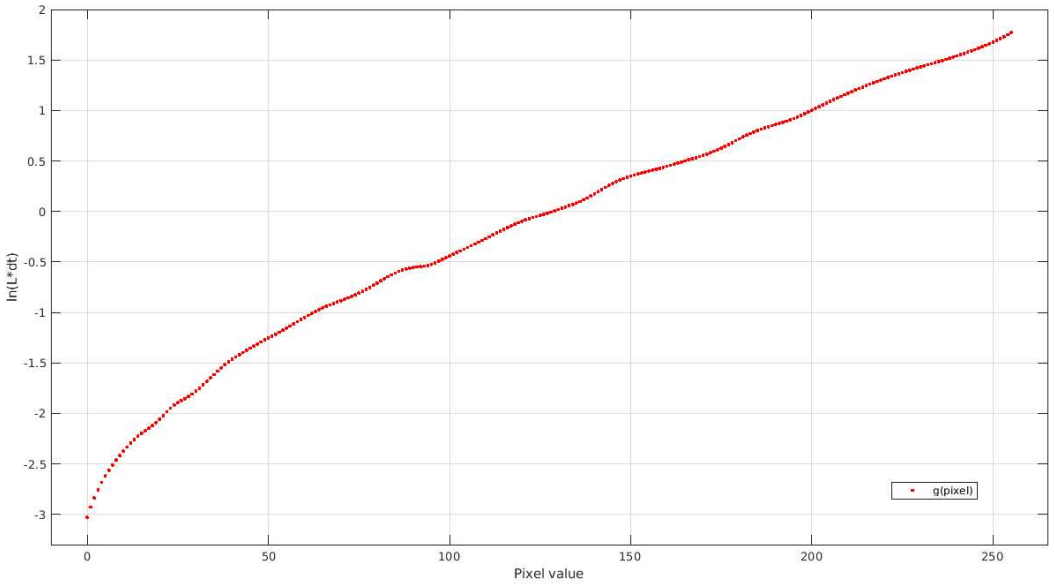


Figure 11: log of inverse CRF from Debevec method on Luxonis OAK-D lite camera

In order to analyze the behavior of the camera with respect the change of light it is better to refer to Figure 12 where the CRF is shown. In particular the steep initial slope indicates that the camera is highly sensitive to light variations in low-light conditions. In other words, small changes in the amount of light ($L * dt$) can lead to significant changes in pixel value. This can be beneficial for capturing details in low-light conditions, but it can also result in overexposure if not careful.
Instead the linear and shallow slope at the end indicates that the camera has a more uniform response to light variations in bright conditions. In other words, an increase

in incoming light will result in a proportional increase in pixel value. This can help prevent overexposure in intense lighting conditions, as the camera will not respond excessively to the increase in light.

In terms of dynamic range, a CRF curve that is steep at the beginning and linear at the end indicates a camera with a wide dynamic range. This is because the camera is capable of capturing details both in low-light conditions (thanks to the steep part of the curve) and in bright conditions (thanks to the linear part of the curve). However, the CRF is just one aspect influencing the dynamic range, other factors include sensor size, pixel count, lens quality, and so on.

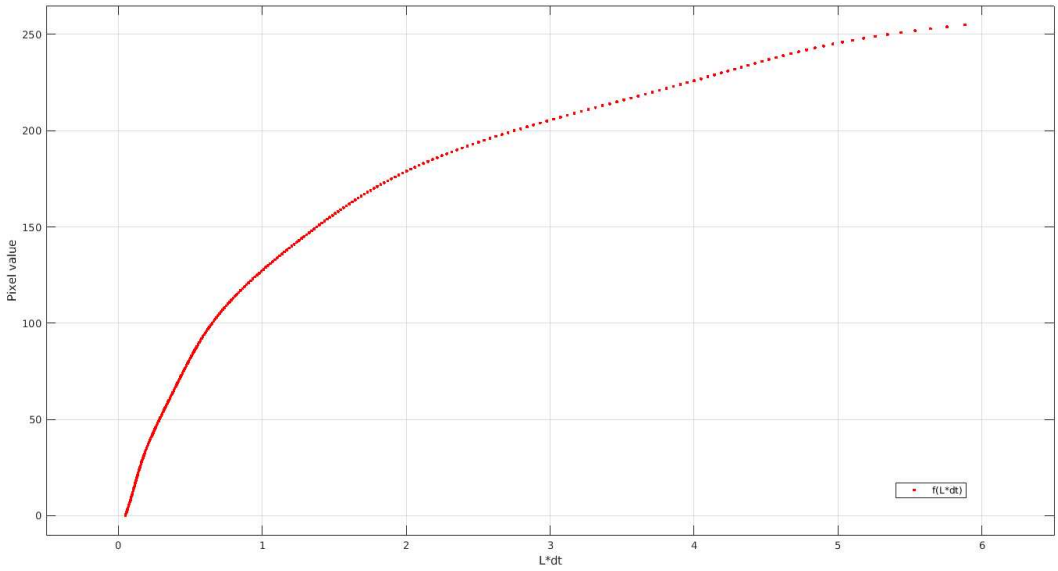


Figure 12: CRF from Debevec method on Luxonis OAK-D lite camera

5 Overview of the literature

Before delving into the conceptual explanation of the algorithms developed in the various selected papers, it is beneficial to clarify some recurring key concepts that form the foundation upon which these algorithms are built.

High Dynamic Range (HDR) imaging captures a broader spectrum of light intensities than standard digital imaging by combining multiple exposures of a scene. This technique is particularly useful in HDR environments, which feature high contrast ratios with intense highlights and deep shadows, preserving details that standard imaging techniques cannot [6].

Visual odometry estimates a camera's position and orientation through image sequence analysis, providing essential real-time navigation data for autonomous vehicles and robotics.

Visual Simultaneous Localization and Mapping (vSLAM) involves constructing or updating a map of an unknown environment while tracking an agent's location within it, integrating visual odometry and SLAM techniques for a comprehensive environmental understanding.

Lastly, The **gradient descent method** is an optimization algorithm used to find the minimum of a function. The core idea is to iteratively move in the opposite direction of the function's gradient, which indicates the direction of steepest ascent, to reach a local minimum. The general formula for parameter update is as follows:

$$x_{n+1} = x_n - \alpha \nabla f(x_n)$$

where x_n is the current parameter value, α is the learning rate, and $\nabla f(x_n)$ is the gradient of the objective function evaluated at x_n [29].

This process is repeated until a convergence condition is met, such as a minimal change in the function value or a maximum number of iterations.

These concepts, together with the CRF explained in Chapter 4, are central to the research reported below.

Subsequent chapters aim to provide an overview of how the various proposed algorithms work. For a more in-depth and comprehensive explanation, please refer to the papers cited [7] [8] [30].

5.1 Camera Attributes Control for Visual Odometry With Motion Blur Awareness

The algorithm developed in the paper under review presents a highly relevant solution for the given case study, primarily due to its meticulous consideration of motion blur. Unlike conventional approaches that prioritize increasing exposure time, this algorithm ingeniously favors boosting ISO sensitivity to mitigate motion blur. By opting for ISO adjustments over exposure time extension, the algorithm effectively tackles motion blur while maintaining image quality.

The algorithm demonstrates innovative concepts such as image synthesis, highlighting its inventive approach to tackling complex challenges in image processing. Unfortunately, despite the algorithm's promising potential, its implementation was not possible due to the unavailability of the project repository at the current date (February 2024).
Nevertheless, in place of implementing the aforementioned algorithm, an alternative approach has been adopted. This alternative algorithm, derived from a study cited in the paper subject of this chapter [8], has similarities with the proposed algorithm. Though not identical, it serves as a viable substitute as elucidated in Chapter 5.2.

5.1.1 Summary

They propose a method for controlling camera attributes by utilizing a weighted sum of image gradient and entropy as quality metrics. The method includes a linear convergence search algorithm to optimize camera exposure, considering the first order derivative of the quality metrics. Scene change speed is estimated using optical flow to determine the maximum exposure time without motion blur. The convergence of the algorithm is accelerated through simulated image generation, allowing for the direct acquisition of nearly optimal camera attributes from overexposed or underexposed images [7].

5.1.2 Metrics adopted

In evaluating image quality, they suggest that the use of image local gradient summation has limitations, particularly within the overexposure range where it fails to effectively distinguish images due to overall increased gradient caused by overexposure. Conversely, image local entropy provides better insight into image information, especially in overexposed scenarios. By combining the advantages of both, they proposed a weighted sum of gradient and local entropy as a more comprehensive image exposure evaluation criterion. The behaviour and the combination of these criteria are shown in Figure 13. The effective ranges of gradient and entropy criteria complement each other, with gradient criterion being more effective for underexposed images, and entropy criterion being more representative for overexposed ones [7].

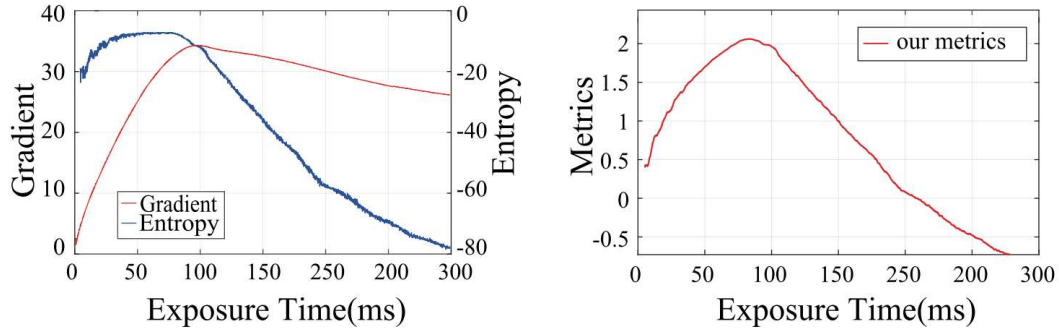


Figure 13: Metric used - Figure taken from the paper [7]

5.1.3 Control architecture

The algorithm comprises three main components: iterative optimization for determining the ideal exposure, estimation of motion state, and control of camera parameters. In the first part, the optimal exposure value is iteratively computed by generating simulated images. During each iteration, the derivatives of the gradient and entropy of the simulated image concerning exposure time are calculated, and the exposure value is updated in the opposite direction of the gradient. The second part involves using the median of optical flow vectors from two consecutive frames to establish the maximum exposure time without causing motion blur. Lastly, cam-

era properties are adjusted based on target exposure and maximum exposure time limits to capture a well-exposed image with minimal motion blur and noise [7].

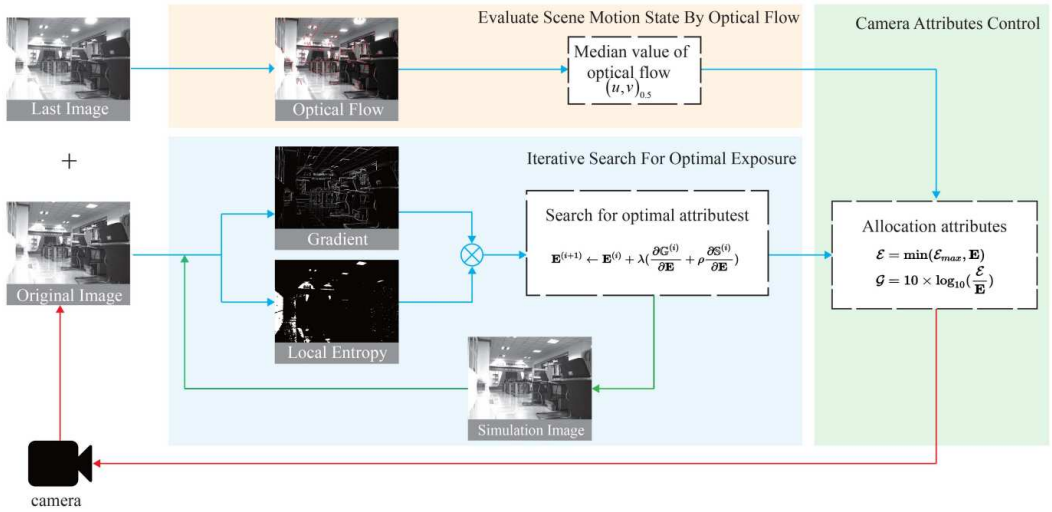


Figure 14: Control architecture of the proposed method - Figure taken from the paper [7]

5.2 Proactive camera attribute control using Bayesian optimization for illumination-Resilient visual navigation

The paper to be briefly outlined in this section was published prior to the one discussed in Chapter 5.1. It shares similarities, particularly in the synthesis of images using the camera response function (CRF). The implemented algorithm is more intricate, which made its implementation challenging, especially since the repository was tailored to a specific camera model. This necessitated significant modifications to adapt the algorithm for the application under consideration.

5.2.1 Summary

This article introduces a proactive control scheme for managing two key camera settings: exposure time and gain control.

They approach camera attribute control as an optimization problem without prior knowledge of the underlying function. They define a new metric for images considering both image gradients and signal-to-noise ratio simultaneously. Using this

metric, they employ Bayesian optimization (BO) to formulate the attribute control and learn environmental changes from captured images. To reduce the workload of image acquisition and Bayesian optimization, they synthesize images using the camera response function instead of directly capturing frames [8].

5.2.2 Metric adopted

In this article, they expand on their previously developed entropy-weighted gradient (EWG) approach and enhance it by incorporating the noise level measure from the signal-to-noise ratio (SNR). This modification creates a comprehensive image utility measure that considers saturation (entropy), edges (gradient), and noise (SNR) altogether. They use entropy to gauge saturation, aiming to reward local patches with greater diversity while penalizing saturation. This is because even a uniformly colored surface may exhibit slight variations, while saturated regions lack any variation in the local image patch [8]. This evaluation metric is referred to as NEWG both in the academic paper and is also reported under this name in this studio.

5.2.3 Control architecture

The control architecture is outlined in Figure 15. It begins with capturing a seed image to assess the environment, which then triggers the camera attribute controller. This controller utilizes the seed image to generate synthetic images via a synthesizing module, feeding them into the Bayesian optimization (BO) module. The optimal camera attributes determined by the BO module are applied to capture real images until a significant change in global illumination is detected. Each incoming image frame is scrutinized for illumination changes by comparing the image metric. The BO phase is activated only when a notable difference between the synthesized image and the actual frame grab is identified in terms of the metric, indicating an illumination variance [8].

The method comprises three main modules: image metric evaluation, control, and image synthesis. In the initial step, they analyze the seed image by computing both its gradient and entropy. Subsequently, they utilize the entropy-derived weights to diminish noise present in the image gradient. Subsequently, a saturation mask is

generated based on local entropy to compensate the entire metric using an activation function. Once metric calculation concludes, a series of steps are iterated until a Gaussian process (GP) identifies optimal attributes. Synthetic images are produced using the camera response function (CRF) and gain functions for the subsequent query image to be learned in the Gaussian process [8].

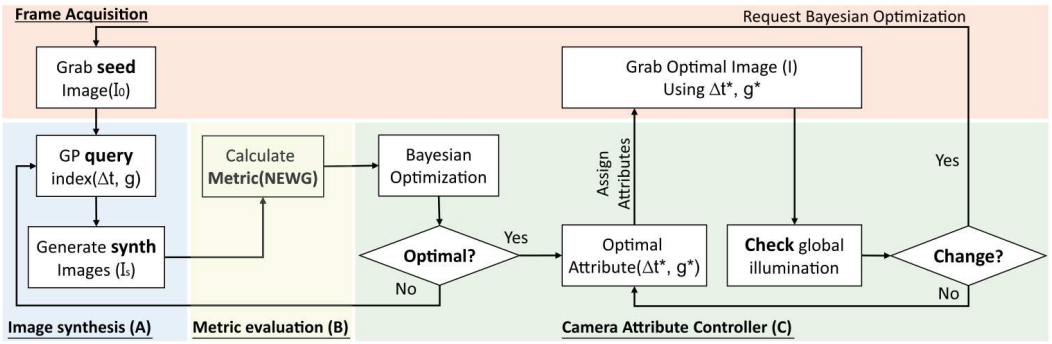


Figure 15: Control architecture of the proposed method - Figure taken from the paper [8]

5.3 Active exposure control for robust visual odometry in HDR environments

The algorithm proposed in this paper introduces a simpler calibration and implementation process. One of its advantages is that the libraries developed by the authors facilitate testing with various control metrics, all based on the gradient descent method for optimization. This enables the evaluation of different control options, allowing for the selection of the most suitable for the desired application. Compared to other studied algorithms, this one boasts faster execution times during the optimization process, thereby enabling, for instance, the possibility of increasing camera frames per second (fps) if required.

5.3.1 Summary

In this article they propose an active exposure control method aimed at enhancing the robustness of visual odometry in HDR environments. Their method assesses the appropriate exposure time by maximizing a robust gradient-based image quality

metric. This optimization is accomplished by leveraging the camera response function of the camera, in fact, with the latter, they were able to evaluate the derivative of their metric concerning the exposure time. Such information allows them to apply mathematically grounded methods, such as gradient descent, in exposure control. [30].

5.3.2 Metric adopted

The metrics developed in this paper for the optimisation process are based on the gradient calculation. They define two metrics, the first is a percentile metric which takes a certain percentile of all gradient magnitudes of the image:

$$M_{perc}(p) = percentile(G(\mathbf{u}_i)_{\mathbf{u}_i \in I}, p)$$

where p indicates the percentage of the pixels whose gradient magnitudes are smaller than M_{perc}, $G(\mathbf{u}_i)$ is the gradient at pixel $\mathbf{u}_i$ and I represent the image. The second metric is called $M_{softperc}$ that is a weighted sum of the sorted gradient magnitudes in ascend order.

$$M_{\text{softperc}}(p) = \sum_{i \in [0,S]} W_{\text{ith}}(p) \cdot G_{\text{ith}}$$

One of the advantage of the soft percentile metric over the percentile metric is that it changes smoothly with the exposure time, which ensures that the estimation of its derivative is more accurate, necessary to apply the gradient descent method [30].

5.3.3 Control architecture

The authors demonstrated, as reported in the paper, that the soft percentile metric $M_{softperc}$ serves as a robust indicator of image quality. Hence, the objective of their exposure control is to maximize $M_{softperc}$ for future images. To accomplish this objective, the exposure time is adjusted based on the most recent image obtained from the camera driver using a gradient ascent approach. Specifically, given an image I and its corresponding exposure time Δt, the desired exposure time for the next image is calculated as follows [29]:

$$\Delta t_{\text{next}} = \Delta t + \gamma \frac{\partial M_{\text{softperc}}}{\partial \Delta t} \tag{13}$$

Here, the derivative of $M_{softperc}$ is computed with the aid of the derivative of the camera response function, which plays a crucial role in the optimization process. The sole calibration parameter of the system is γ that represents the magnitude of the update step. Subsequently, the newly determined exposure time is transmitted to the camera driver, and then the formula 13 is executed on the subsequent image [30].

6 Implementation of the exposure control algorithms

In the initial stages of the project, various programming experiments were conducted using the Python language for controlling the cameras. However, as the development progressed, the need for more comprehensive and fine-grained control over the camera functionalities became apparent. Consequently, the decision was made to transition to C++ programming, as it provides greater control and more efficient performance compared to Python.

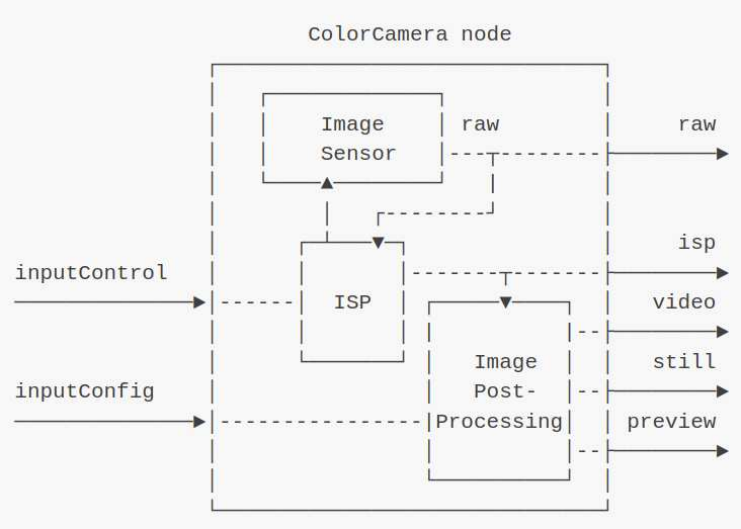
The shift to C++ not only allowed for more precise control over the cameras but also facilitated easy integration of third-party algorithms and utilization of more efficient image processing libraries. This opened up new avenues for development and significantly enhanced the system's capabilities.

Moreover, the C++ language offers greater flexibility in code optimization and implementation of advanced features, enabling the achievement of more satisfactory results in terms of performance and functionality.

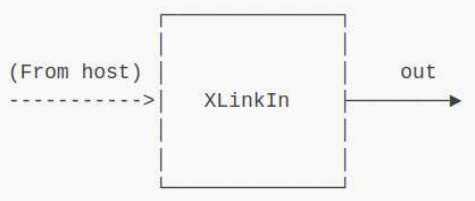
6.1 Integration into the camera control system

The API structure is based on the depthai-core libraries for C++ and depthai-python as explained in section 3.2.1. The API comprises two main components: the host and the device, in the latter is upload the pipeline.
The pipeline represents the complete workflow on the device side. In the case of study the pipeline is composed by three nodes: *colorCamera* node, *xLinkIn* and *xLinkOut* as shown in Figure 16. Each node encapsulates a specific functionality, have inputs and outputs, and their behavior can be customized through configurable properties.
The *colorCamera* node is the most important in this contest, it has the task of capturing the images based on the configurations given by the *inputControl* message. The latter is linked to the *xLinkIn* node that connects the host with the camera, instead the video output message is linked to the *xLinkOut* node providing the captured frames to the host.
The other messages wasn't linked since they weren't useful for the purpose of this

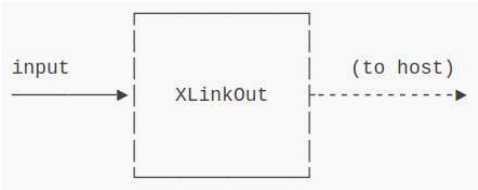
thesis. In particular *inputConfig* can be used for cropping, warping, rotating, resizing, etc. an image in runtime. Furthermore raw data is the primary information that needs to be converted into a more easily readable format if one does not require this type of file. The conversion can be performed directly by the ISP module or by the host, but it is more convenient from a speed standpoint to perform it locally. *Still* is a type of message that collects the frames that are requested to be saved by the host via the *inputControl* message. *ISP*, *video* and *preview* deliver the same type of information but based on the configuration can be used for different purposes.
In the case of this study the *video* message was used since provides more flexibility with regard the resolution settings, since the *preview* message is more used as input to neural networks or other computer vision tasks that required small squared size images.



(a) ColorCamera node



(b) xLinkIn node



(c) xLinkOut node

Figure 16: Nodes used to implement the algorithms

Listing 1 presents the basic code used to define and load the pipeline into the

camera, the code is placed inside the main, before the control and acquisition process. In particular, the first step is to create the pipeline object (line 2) and then populate it with the necessary nodes (lines 6, 7, 8). After this procedure, the *streamName* is set for each message, which allows interfacing with the device from the host, and the nodes are configured. In this case, it was necessary to configure only the *colorCamera* node.

Among the main steps, there is the linking between the various nodes (lines 21 and 22), which allows the transmission of messages between them. Finally, the pipeline is loaded into the device, and during this phase, if there are connection problems between the host and the camera during the program's run, an error is reported.

Once the loading phase is complete, the queues containing the outgoing and incoming messages are defined. The system operates through FIFO type buffers. At line 29, it is possible to note the presence of a 1 and a false parameter among the settings. These settings refer to the buffer size and the blocking parameter. In particular, it was chosen to set the buffer to a single element with the possibility of being overwritten if not emptied beforehand. This choice was made to have better control over the frames acquired by the camera and to simplify the acquisition phase by the host, as explained in the following chapters.

The last setup performed before starting to capture and adjust control parameters is setting the manual focus. There are primarily three possible configurations: the first is continuous automatic focus, which continuously adjusts focus throughout the acquisition process based on internal camera algorithms; the second allows for automatic focusing only at the beginning of the acquisition; and finally, the third is manual setup.

The first option was not chosen because during the focus point search, the camera lens, as explained in Chapter 4.1, moves, and to find the correct focus point, the entire available range is traversed before stopping at the optimal point. The issue with this type of mechanism is that during the process, the acquired images are inevitably out of focus, resulting in unusable frames.

The second option might be valid, but since automatic focusing in these cameras depends heavily on good exposure, it's necessary for the exposure to be optimal

during the initial acquisition phase. However, this cannot be guaranteed, resulting in sub-optimal focus values that then affect the entire subsequent phase.

On the other hand, the third option was chosen for the reasons described earlier and also based on the fact that the camera is positioned at a fixed distance from the conveyor belt. Therefore, once set correctly, it doesn't need further adjustment.

```
// Create pipeline
dai::Pipeline pipeline;

// Define sources and outputs -> 3 nodes: colorcamera
// and xlinkin/out nodes
auto camRgb = pipeline.create<dai::node::ColorCamera>();
auto controlIn = pipeline.create<dai::node::XLinkIn>();
auto videoOut = pipeline.create<dai::node::XLinkOut>();

// Define the names of the messages that travels
// between host and camera via xlink
controlIn->setStreamName("control");
videoOut->setStreamName("video");

// Properties
camRgb->setBoardSocket(dai::CameraBoardSocket::CAM_A);
camRgb->setResolution(
    dai::ColorCameraProperties::SensorResolution::THE_1080_P);

// Linking
camRgb->video.link(videoOut->input);
controlIn->out.link(camRgb->inputControl);

// Connect to device and start pipeline
dai::Device device(pipeline);

// Get data queues
auto controlQueue = device.getInputQueue("control");
auto videoQueue = device.getOutputQueue("video",1,false);

// Set manual focus
```

```
32 dai::CameraControl ctrl;
33 ctrl.setManualFocus(0); //0-255 far - near
34 controlQueue->send(ctrl);
35
36 cv::waitKey(300);
```

Listing 1: Pipeline creation for Luxonis camera

6.1.1 Control and acquisition process

To control the camera parameters, it is necessary to create an object of the class *cameraControl* found in the *deptai-core* library. Subsequently, the function *setManualExposure* is used to set the instances of the *cltr* object to the desired values *expTime* and *sensIso*. Finally, the previously created object *controlQueue* (line 28, Listing 1) is used to write the desired settings into the control buffer. Listing 2 shows an excerpt of the code that performs this operation. Specifically, at line 1, there is a command that, when associated with a variable, allows for the acquisition of a list of messages present in the *xlinkOut* buffer; in the case of this application, the only message present is ”video”. The list returned by the *device.getQueueEvent()* function presents a single element as the buffer size has been set to one element.

In this case study, this function was chosen to be inserted at this specific point as it allows synchronization with the camera’s workflow.
Another fundamental aspect of this function is that it waits for the arrival of a new message, meaning that if it is called at the beginning of the acquisition of a new frame in the camera workflow, the algorithm will pause for at most 33 ms, corresponding to the camera’s 30 fps acquisition rate. This is necessary to perform optimization operations in the time slot between the acquisition of one frame and the next, approximately 33 ms. This concept is fundamental to have a precise control of the frames and of them settings, in fact the exposure control algorithms need of the acquired images and the corresponding exposure settings to be able to estimate the optimum values for the next frames. This can’t be done if there isn’t this kind of synchronisation.

```
1 device.getQueueEvent();
```

```
2
3 dai::CameraControl ctrl;
4 ctrl.setManualExposure(expTime, sensIso);
5 controlQueue->send(ctrl);
```

Listing 2: Set exposure parameters code

The final step, as is shown in Listing 3, involves acquiring the message from the buffer using the *get* function on the *videoQueue* object previously created in Listing 1. Subsequently, a *cv::Mat* variable is created, to which the frame contained within the *videoin* message is assigned. From this point forward, it is possible to utilize all the functionalities provided by the OpenCV library on the newly acquired image.

```
1 auto videoIn = videoQueue->get<dai::ImgFrame>();
2 cv::Mat init_img = videoIn->getCvFrame();
```

Listing 3: Frame acquisition code

6.2 Problems of the development phase and solutions

Subsequent chapters address common issues encountered in the algorithm's implementation process. In Chapter 7, instead, there is a more detailed analysis for each individual algorithm.

6.2.1 Gain to ISO conversion

One of the challenges encountered is that the tested control algorithms modify the sensitivity of the image sensor by setting a value in dB, known as gain. In the case of the algorithms under consideration, this value oscillates between 0 dB and 12 dB, representing the maximum and minimum values, respectively. In contrast, the camera, as already explained, is controlled by ISO, from 100 to 1600.

The term 'dB' stands for decibel, which is a logarithmic unit used to express the ratio between two values of a physical quantity, often power or intensity. The decibel is one-tenth of a bel, a unit named after Alexander Graham Bell. The formula to calculate the ratio in decibels (dB) is given by:

$$\text{Ratio in dB} = 10 \cdot \log_{10}\left(\frac{P_1}{P_0}\right) \tag{14}$$

where P_1 is the power level of interest, and P_0 is the reference power level. For amplitude quantities such as voltage or sound pressure, the formula is modified to:

$$\text{Ratio in dB} = 20 \cdot \log_{10}\left(\frac{V_1}{V_0}\right) \tag{15}$$

where V_1 and V_0 are the amplitude values. The decibel scale is a convenient way to express large or small numbers. For power quantities, a 10 dB increase means that the power ratio is 10 times greater. For amplitude quantities, a 10 dB change would correspond to a square root of 10 change in amplitude, and a 20 dB change would represent a tenfold change in amplitude

The progression of ISO values exhibits an exponential behavior, thus matching the utilization of a decibel (dB) scale. Indeed, by doubling the preceding ISO value, one achieves the equivalent of doubling the light exposure on the sensor. The challenge arises from the discrepancy between the utilized scale and the one operable in the camera. Specifically, the ISO values that can be employed range from 100 to 1600, following the sequence: 100, 200, 400, 800, 1600. As elucidated earlier, the sensitivity doubles at each increment.

In this particular case study, the development of a function was imperative to align the decibel (dB) scale with the pre-established ISO values. It was specifically chosen to double the ISO values at every 3 dB increment as in formula 14, as opposed to the standard 6 dB increment for amplitude quantities, as in formula 15. This calibration enabled a correspondence of ISO 100 to a gain of 0 dB and ISO 1600 to a gain of 12 dB, contrary to how it is considered in the algorithms implemented and explained in Chapter 5, which instead refer to the formulation 15.

A straightforward function, as shown in Listing 4, was developed to integrate both exposure time and ISO control, converting the gain value derived from the control algorithm into an equivalent ISO setting. Additionally, modifications were made to libraries that required linear values rather than decibel measurements. This was particularly pertinent for the libraries involved in image synthesis, referenced in section 5.2 [8].

The final formulas used for the conversion are shown in 16, where *min_ISO* corresponds to the minimum ISO value, which is 100.

$$\text{gain\_} = 10 \cdot \log_{10}\left(\frac{\text{sensISO\_}}{\text{min\_ISO}}\right) \qquad \text{sensISO\_} = \text{min\_ISO} \times 10^{\left(\frac{\text{gain\_}}{10}\right)} \tag{16}$$

In this function, a further `clamp` function is invoked to guarantee that the desired values do not surpass the established boundaries defined as the function's arguments. Finally the exposure control message is send as explained in Listing 2.

```
void setExp(int expTime_, double gain_,
    std::shared_ptr<dai::DataInputQueue> controlQueue_) {

    double a = min_ISO;
    double b = pow(10.0,0.1);
    double sensIso_ = a*pow(b,gain_);
    expTime_ = clamp(expTime_, min_exp_t, max_exp_t);
    sensIso_ = clamp((int)sensIso_, (int)min_ISO, (int)max_ISO);

    dai::CameraControl ctrl_;
    ctrl_.setManualExposure(expTime_, sensIso_);
    controlQueue_->send(ctrl_);
}
```

Listing 4: Set exposure function

6.2.2 Control delay

After implementing the algorithms proposed by the papers, an analysis was performed on the operation of the programs since both did not show the anticipated results. A thorough evaluation of all parameters led to the hypothesis that the execution of the message in `inputControl` was not immediate, or at least not carried out in the frame following the one in which the message was dispatched.

To validate this hypothesis, a C++ software was crafted to ascertain the time interval between dispatching the control message and the actual implementation of the settings. This was accomplished by analyzing the mean value of the absolute difference between the current and preceding frames. Operating under the premise that environmental conditions are stable throughout the test, the absence of variations should yield a value approximately equal to zero. Conversely, a change in settings

should prompt a variation in the parameter being scrutinized.

The test was executed by applying this concept to two distinct programs. The first was expressly developed to conduct the aforementioned test, while the second entailed altering an example within the depthai-core library. This phase was pivotal in determining whether the issue could indeed be ascribed to the company or to programming errors. Moreover, the test was performed on both the OAK-D lite and pro cameras, yielding equivalent outcomes.

The test yielded results that undermine the actual applicability of the algorithms under study due to a delay ranging from approximately 220 ms to 270 ms. Consequently, any employed algorithm cannot achieve optimal results. Assuming the optimal exposure value is attained within 3 to 4 frames with an average delay of 250 ms, the overall time spans between 750 ms and 1000 ms, which are excessively high for a control system. Furthermore, this issue introduces significantly greater complexity in algorithm application, as the synchronization between the setting of exposure values and the corresponding frame is not assured.

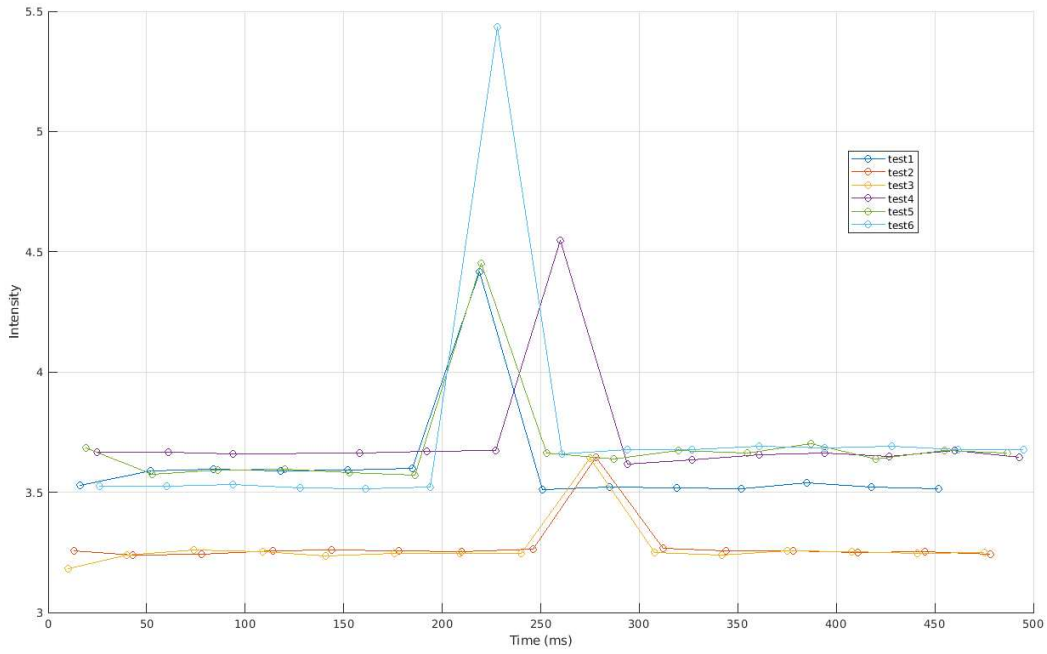


Figure 17: Mean difference between two consecutive frames

The outcome of the test is illustrated in the graph of Figure 17, which was executed multiple times as evidenced by the data. The graph of Figure 17 exhibits the

mean intensity value of the difference between the current and preceding frames. A discernible peak manifests, evidencing the actual moment of setting modification. The transmission of the message containing new exposure time and sensitivity settings is designated at the zero point on the graph, correlating with the program's initiation of the timer upon message dispatch. The amplitude of intensity variation is not markedly significant, as the test was performed on a library example that permitted only minimal changes upon each message transmission, notably 100 ms and 50 ISO, leading to the similarity between successive frames. However, this did not hinder the verification of the hypothesis, as it is verifiable that the change in exposure transpires between 219 ms and 278 ms subsequent to the dispatch of the control message. Various solutions aimed at enhancing the synchronization between the transmitted messages, the control algorithm, and the frame capture were explored to resolve this issue. Nevertheless, none led to substantial outcomes, as the challenge is associated with the communication between the host and the device, especially the `ColorCamera` node. Regrettably, although the tests corroborate the issue, there is an absence of documented information that might have predicted the problem or elucidate a potential resolution. It is likely requisite to employ an alternative camera type, switch brands, or evaluate the Power over Ethernet (PoE) cameras from the same producer.

7 Analysis of the implemented control algorithms

In this chapter, various algorithms derived from the papers that have been previously analyzed are discussed. An additional algorithm has been introduced which allows for the testing of the image evaluation part and the calculation of the conveyor belt speed, with implementation in ROS. Various aspects of the different controllers have been analyzed and, where necessary, useful information has been acquired and displayed to explain the characteristics, performance and problems of the various codes.

7.1 Proactive camera attribute control using Bayesian optimization for illumination-Resilient visual navigation

The algorithm proposed in the paper [8] is hosted in a GitHub repository, which also contains separate modules for independently testing the different components of the full algorithm.
In this case study, the complete algorithm was implemented, with tests conducted on the various individual modules. Notably, the primary modules pertaining to Bayesian optimization and image synthesis are thoroughly elaborated upon in the subsequent chapters, which also include a discussion of the encountered problems.
A subsection is dedicated to explaining the approach employed for the potential calibration of the optimization process parameters, which are instrumental in identifying the optimal exposure parameters. As explained in Chapter 6.2.2, the delay between setting the desired exposure and the frame with the correct settings does not allow for better performance compared to the auto-exposure system present within the Luxonis camera. Despite this, the algorithm has been implemented to maintain $30\ fps$ by synchronizing the frames, even though the response time is around $250\ ms$. In particular, the algorithm developed for this application was structured in order to set the exposure time and sensitivity for each available frame, therefore 30 times per second. This was feasible because the algorithm presented in the paper relies on a seed image with a fixed exposure value, which is also its main flaw as will be explained later.

The strategy involves alternating the seed exposure settings with those proposed by the optimization process, thus creating, at steady state, a buffer that alternates between a seed image and an image with the proposed settings. The latter is the one displayed and used for subsequent processes and the seed image is fed into the optimization process.
Despite an inherent delay, this method allows for the utilization of idle times to process images. It should be noted that the acquisition of the correct images occurs at $15\ fps$, as the other $15\ fps$ are allocated for the seed images.

7.1.1 Synthetic image generator

The designated module synthesizes from a seed image all the exposure time and gain values supported by the utilized camera. For each synthesized image, it computes the corresponding evaluation metric, termed *NEWG* as explained in 5.2.2. Upon completion of the calculations, it selects the optimal parameter and exhibits the synthesized image with the exposure settings that garnered the highest evaluation.
Adaptations were made to the image synthesis module to accommodate the Luxonis camera's specifications. Modifications were particularly focused on the sections involving the conversion of gain to ISO value, as delineated in 6.2.1. Displayed in Figure 18 is a chart elucidating the variation of *NEWG* in relation to the set exposure time and ISO. In Figure 18, a red star marks the point corresponding to the optimal exposure parameters, chosen at an exposure time of $4\ ms$ and ISO 400.

Figure 19 displays the seed image (a) and the resulting synthetic image (b). Conversely, Figure (c) depicts the same test context in terms of lighting and scene but with the optimal settings applied base on the calculation done in the previous step. The seed image was set with minimum exposure values of $4\ ms$ and ISO 100 ($0\ dB$ gain). This decision was made because, for shorter exposure times, the image was completely underexposed, and consequently, the algorithm could not synthesize the image. The synthesis process involves multiplying two factors derived from the desired exposure time and gain value [8]. The first factor is calculated based on the camera response function, while the second follows the analysis explained in

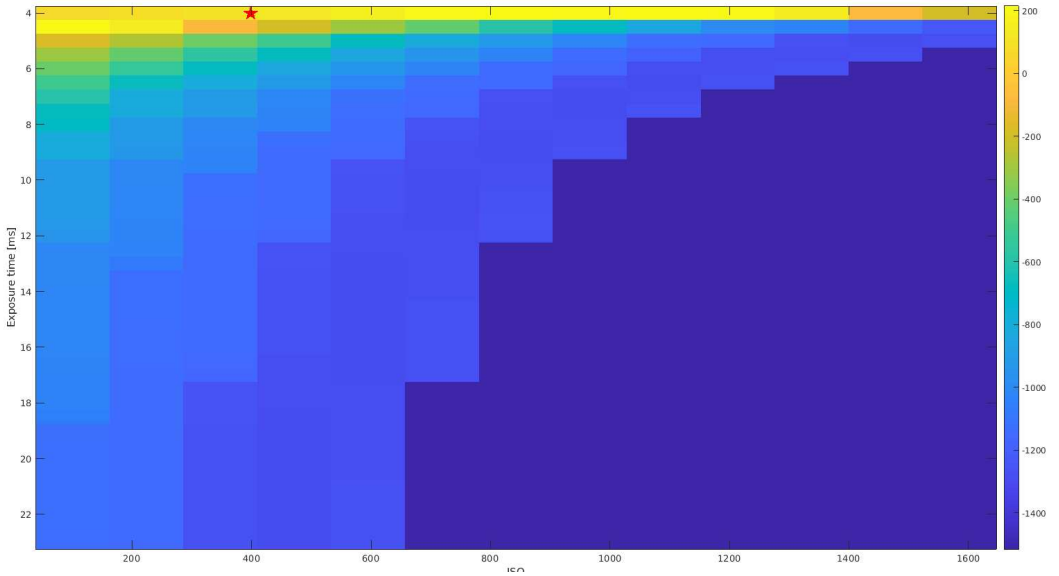
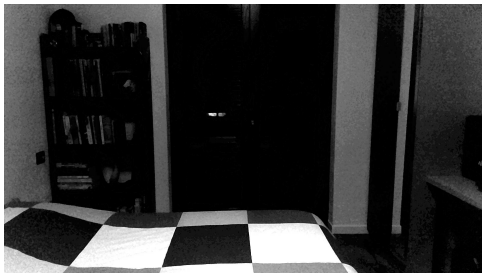


Figure 18: NEWG result for each synthetic image

Chapter 6.2.1. Essentially, for every $3\ dB$ increase, the values of the individual pixels are doubled.

The issue encountered with this approach is that if the seed image, which has fixed exposure values, is in an environment with insufficient lighting, it will not be capable of capturing any detail. Consequently, the resulting frame will be composed of pixels with a value of zero. Multiplying these zero-valued pixels by the synthesis parameters will only yield a result of zero. This issue results in a loss of information during the frame acquisition process, rendering the frame inadequate for the scene's requirements, this concept is more clear in Chapter 7.1.2. Despite the aforementioned issue, under optimal conditions, such as those recreated during testing and visible in Figure 19, the algorithm demonstrates potential. It can recreate a synthetic image upon which the *NEWG* evaluation metric is applied to estimate the scene's optimal exposure time and gain parameters. Indeed, Figure (c) reveals that the image captured with the optimal settings closely resembles the synthesized image (b), though certain areas like the central patio door or the left bookshelf are not accurately synthesized due to the problems described earlier.

(a) Seed image



(b) Optimal synthetic image

(c) Captured image with optimal exposure parameter

Figure 19: Images used in the synthetic image generator

7.1.2 Bayesian optimization

The module related to the Bayesian optimization process facilitates the calibration and testing of the optimal parameter search process without interacting with the rest of the algorithm. This allows for a more rapid and accurate calibration, as modifications to the algorithm can be made without concern for impacting the entire program.

Bayesian optimization is a strategy for optimizing objective functions that are costly to evaluate. It constructs a probabilistic model of the function, often employing a Gaussian process as the surrogate model. A Gaussian process is a stochastic process where every point in some continuous input space is associated with a normally distributed random variable [29] [5]. Within Bayesian optimization, this model is utilized to predict the location of the function's maximum in the case of study, with the Gaussian process providing a measure of uncertainty in these predictions.

Given the complexity of the optimization process, the separate module helps in test-

ing and calibration of the optimization parameters. The module in question is divided into two subprograms: one in MATLAB that also integrates image synthesis, and one in C++ based on a *.csv* file, which allows for a greater focus on evaluating the performance of the individual optimization process and for which there is a specific focus in the next chapter 7.1.3.

Figures 20, 21, 22, and 23 refer to the MATLAB program. In fact, the algorithm for synthesizing images explained in Chapter 7.1.1 is implemented to obtain the *NEWG* for each exposure setting using a seed image. However, in this case, the search for the optimal value is not conducted by finding the maximum value in the *NEWG* value matrix but by using the Bayesian optimization process. Particularly in this instance, a maximum number of iterations was set to five steps, which, as can be seen in Figures 20 and 22, amply allows reaching the optimal *NEWG* value. This confirms the correct and efficient functioning of this approach, as in real conditions, it is not necessary to evaluate every synthesized image and then find the exposure parameters that obtain the maximum *NEWG* value. It is sufficient to refer to the Bayesian optimization process, synthesizing only the required images that will be used to train the algorithm until the optimal value is found or the maximum number of iterations is reached.

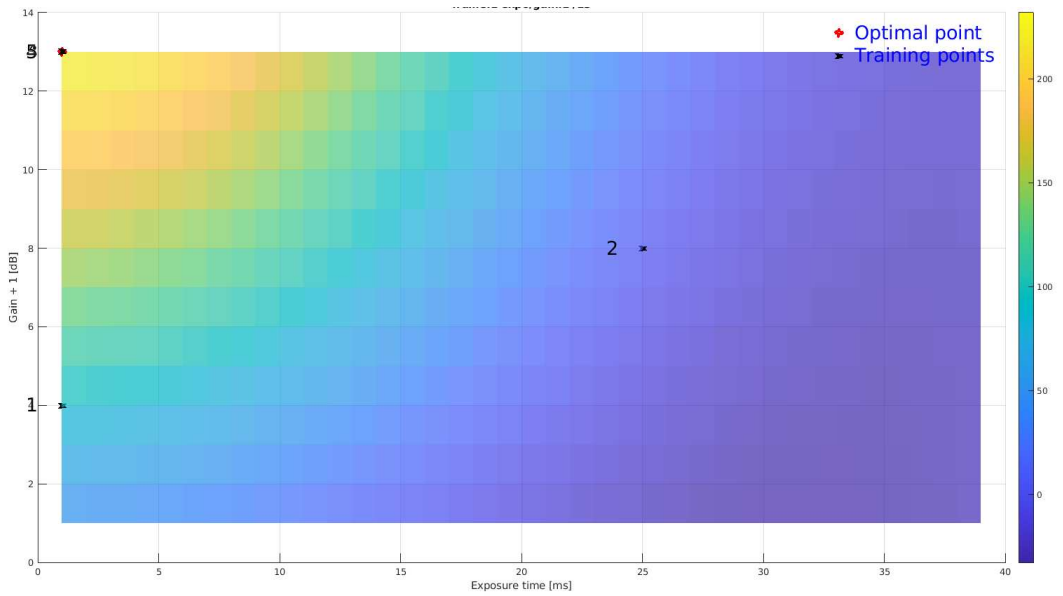


Figure 20: Bayesan optimisation based on NEWG

Figure 21: Comparison between seed (left) and synthetic (right) image with optimal exposure values obtained by the Bayesan optimization process

To better understand the issue discussed in Chapter 7.1.1, two different tests were conducted. The first test, illustrated in Figures 20 and 21, was performed using a seed image with exposure time and ISO values set to $4\ ms$ and 200 (3 dB gain), respectively. These values correspond to optimal conditions; despite the seed image being underexposed, it presents much information, as seen in Figure 21 (left). The result shown in Figure 21 (right) demonstrates proper functioning under optimal conditions, as the image can be considered correctly exposed.

Conversely, the test depicted in Figures 22 and 23 was conducted using a more underexposed seed image corresponding to an exposure time and ISO of $4\ ms$ and 100 (0 dB gain). This test demonstrates how, in conditions of low light, the use of an exposure time and ISO that are too low for the environmental conditions means that the image synthesis algorithm is unable to produce an image consistent with the scene. This impacts the calculation of the *NEWG* evaluation parameter and, consequently, the entire optimization process.
To address this issue, it is necessary to ensure that the seed image has exposure values that result in an underexposed image suitable for this exposure process, but not too low for the brightness conditions present in the scene. Alternatively, this algorithm can be chosen under conditions where there is a minimum level of brightness, for example, from a light source, which would allow for the calibration of the minimum exposure values so that the seed image contains enough information to create the synthesized images.

As explained in Chapter 6.2.2, due to the problem of control delay, no further

investigations were conducted to improve this aspect of the proposed algorithm. Moreover, theoretically, this issue might have been overcome with a different approach in the paper [7], which, as previously mentioned, was not possible to implement. Despite this, software has been developed that allows for the rapid calibration of the Bayesian optimization process, which is explained in the following chapter.

To conclude, the graphs in Figures 20 and 22 have the Y-axis with values ranging from 1 to 13. These values were set in this way due to software requirements, as described in the Y-axis label, the values refer to the gain + 1. This implies that the actual gain values range from 0 to 12 dB, corresponding to ISO from 100 to 1600.

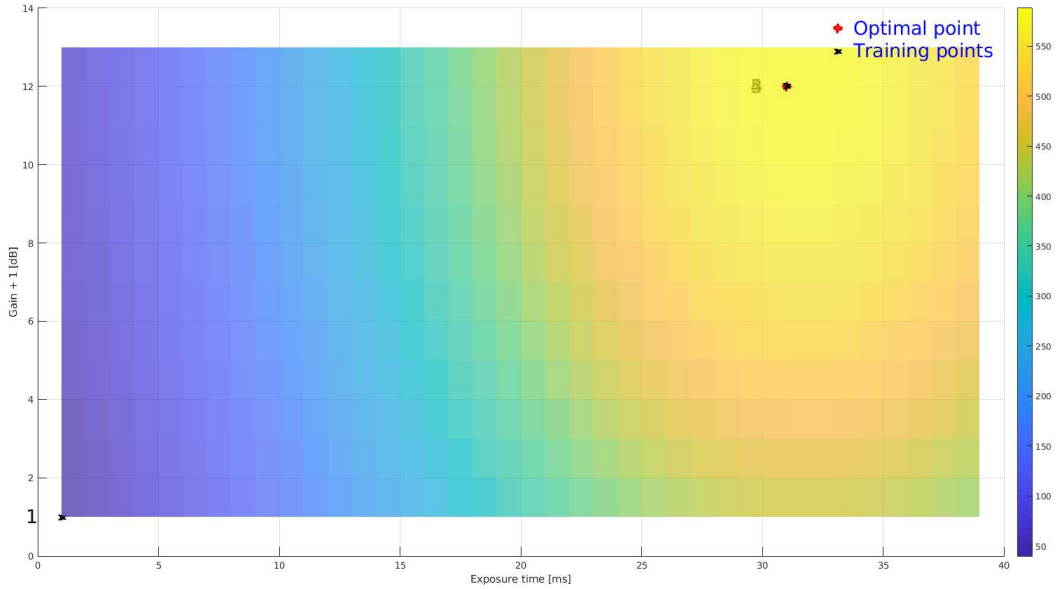


Figure 22: Bayesan optimisation based on NEWG

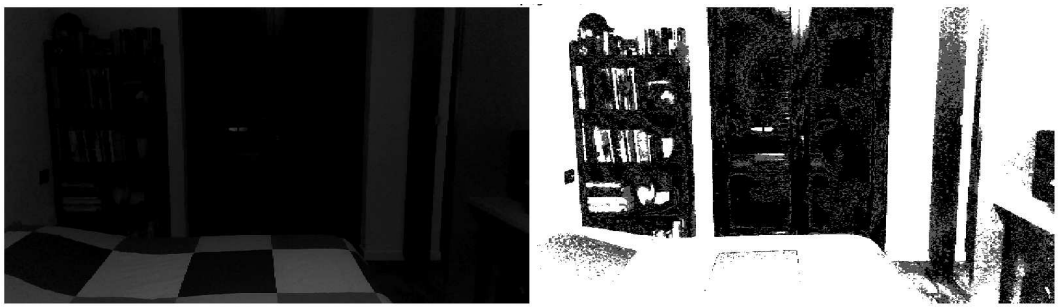


Figure 23: Comparison between seed (left) and synthetic (right) image with optimal exposure values obtained by the Bayesan optimization process

7.1.3 Tuning software

As explained in the previous chapter 7.1.2, two pieces of software have been developed to calibrate and demonstrate the functioning of the Bayesian optimization process. This chapter illustrates the calibration algorithm developed by the authors of the paper in C++. The underlying idea of the software is quite straightforward: it involves extracting data related to exposure and the corresponding *NEWG* evaluation parameter from a *.csv* file and applying the optimization algorithm to ensure that the optimal value achieved is the expected one, namely the maximum value of *NEWG*. If not, it is possible to calibrate the system parameters and restart the program. This simple application allows focusing solely on the search for exposure parameters linked to the optimal *NEWG* value evaluating speed and accuracy, abstracting everything from the context, especially from the synthesis of images and the calculation of the evaluation parameter, thus reducing time and potential bugs.
To ensure the proper functioning of the software, a specific C++ application was developed for this purpose. It is designed to sequentially acquire frames from the Luxonis camera, incrementing exposure time and ISO to cover the entire spectrum available from the camera. This ranges from exposure values of 10 us to 33000 μs in steps of 500 μs, and ISO values from 100 to 1600 in steps of 50. For each obtained frame, the corresponding *NEWG* evaluation parameter was calculated, to do so it was necessary the implement the libraries containing the specific functions for this purpose. The data obtained were saved in a *.csv* file, making it available for use in the calibration program mentioned above.

In Figure 24, we observe the behavior of the Bayesian optimization process. The optimization steps are marked with numbered crosses, indicating the sequence of iterations. A green star marks the final chosen point as the optimal value. In this instance, the number of iterations was set to 6 steps; hence, the optimal value is obtained at the seventh step. Conversely, a red star indicates the maximum value of *NEWG*, which corresponds to the optimal exposure value. The colored bands plotted in the graph represent the *NEWG* value calculated for each exposure, and acquired based on the algorithm previously explained. The test was done in a context with constant ambient light conditions.

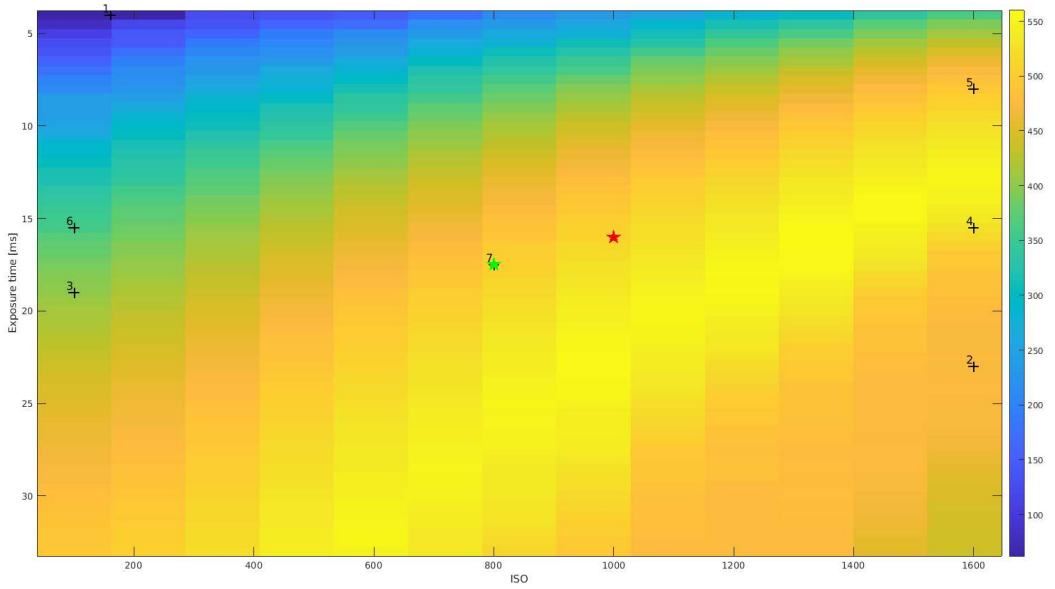


Figure 24: Bayes optimization test

The optimization algorithm, as can be seen by the trend of the numbered crosses in Figure 24, requires calibration to ensure its effectiveness and reliability. However, this calibration has not been performed due to encountered issues already discussed and because the calibration should be conducted in the actual context of the camera's usage.

7.2 Active exposure control for robust visual odometry in HDR environments

The Figure 25 illustrates the exposure time calculated by the algorithm developed in the paper [30] and implemented in this case study. The graph is divided into three distinct areas: an initial blue area representing the initialization phase, followed by red and green areas which correspond to the 'light off' and 'light on' phases, respectively. The experiment was conducted in an environment where the luminosity can be considered constant during these two phases. This constancy allowed for an assessment of the algorithm's performance. However, due to issues encountered as referenced in Chapter 6.2.2, it was not possible to evaluate the execution speed. The software developed for this case study, written in C++, is engineered to wait between the acquisition of one frame and the next. This interval allows the necessary

time for the desired exposure settings to be correctly established. Without this measure, the controller would lack the correct references, rendering it non-functional. This careful design ensures that the exposure settings are accurately synchronized with the frame capture process.

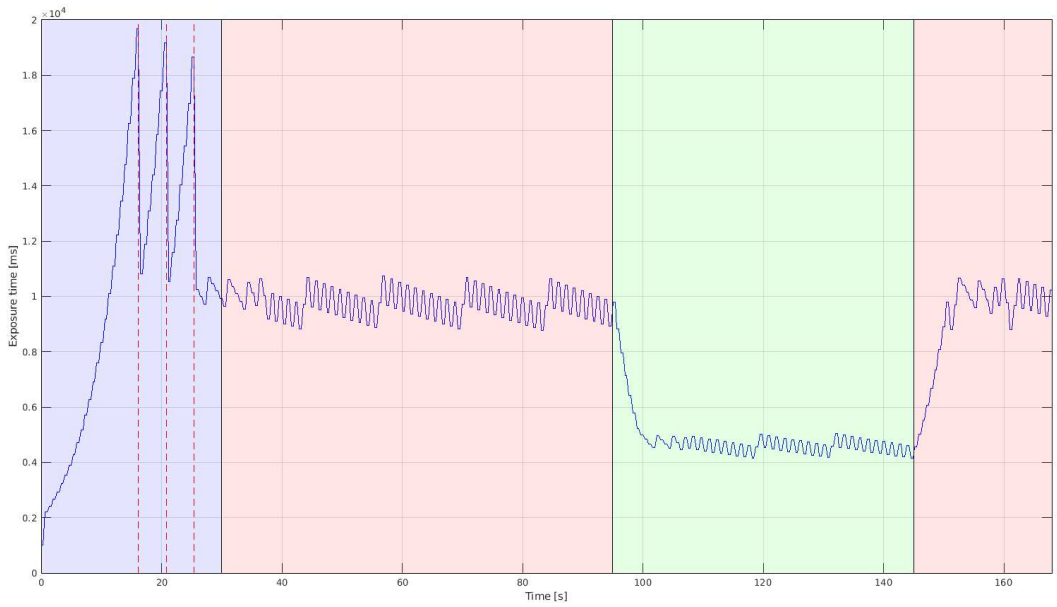


Figure 25: Exposure time set by the algorithm developed by [30], the red and greed areas are respectively light off and on, whereas the blue one is the initialisation phase. The red segmented lines correspond to an increase in ISO

Aside from the issue of the delay between the command to set the exposure parameters and the actual frame with the updated parameters, the algorithm requires calibration. This calibration can be performed once the installation on the conveyor belt is completed. In this case study, it was not possible to calibrate the algorithm, partly because the integrated camera control, which does not have this delay, would still outperform it.

To calibrate the system, given the simplicity of the control algorithm, it is sufficient to adjust the parameter γ in the gradient descent method equation 13.

Given the oscillatory behavior of the exposure time, it is possible that the value of γ is too high, which may prevent the controller from stably reaching the optimal value. This could lead to fluctuations in the exposure time, indicating the need for a finer adjustment of the γ parameter to achieve a more stable control over the

exposure, as can be seen in the red and green areas in Figure 25.

In addition to the parameter γ, it is possible to calibrate the threshold values for the increment and decrement of ISO values. The latter is modified based on two parameters that act as the upper and lower limits of a hysteresis control. Specifically, in this case, the ISO is increased if it exceeds a value of $20000\ ms$ and is decreased when it falls below $200\ ms$. In the graph shown in Figure 6.2.2, the changes in ISO are represented by dashed red vertical lines, corresponding to an increase from 200, to 400, to 800, and finally to 1600 ISO.
As expected, an increase in ISO corresponds to an immediate decrease in exposure time, which allows for maintaining a constant overall exposure. The relationship between ISO increment and exposure time decrement is crucial for achieving the desired exposure balance.

7.3 Luxonis integrated auto-exposure control

In this chapter, the automatic exposure control within the camera has been tested and implemented. To complete the system, a library was implemented that allows the calculation of the *NEWG* evaluation parameter introduced in Chapter 5.2. Furthermore, calculations were made to control the speed of the conveyor belt and ROS was implemented to communicate the information acquired from the camera to possible other devices.

In this case, it was not necessary to implement any particular algorithm to control the exposure and sensitivity of the camera. It was sufficient to acquire the exposure data through the *getSensitivity()* and *getExposureTime()* functions on the previously created *dai::ImgFrame* object.

The camera's auto-exposure system, by default, prioritizes increasing exposure over ISO, up to around frame-time (subject to further limits imposed by anti-banding) [11]. This conflicts with the purpose of the thesis, but it was necessary to implement it to test the rest of the code due to the problems encountered regarding the delay explained in Chapter 6.2.2.

The test was carried out in a room with five identical light sources, which were turned on in sequence to verify the variations of the control parameters, the speed of

the controller, and its behavior. The first graph at the top in Figure 26, named ”Light condition”, shows the trend of the light conditions of the environment where the ”light level” indicates the number of light sources switched on, where 0 corresponds to a slight light source to still have information to capture.

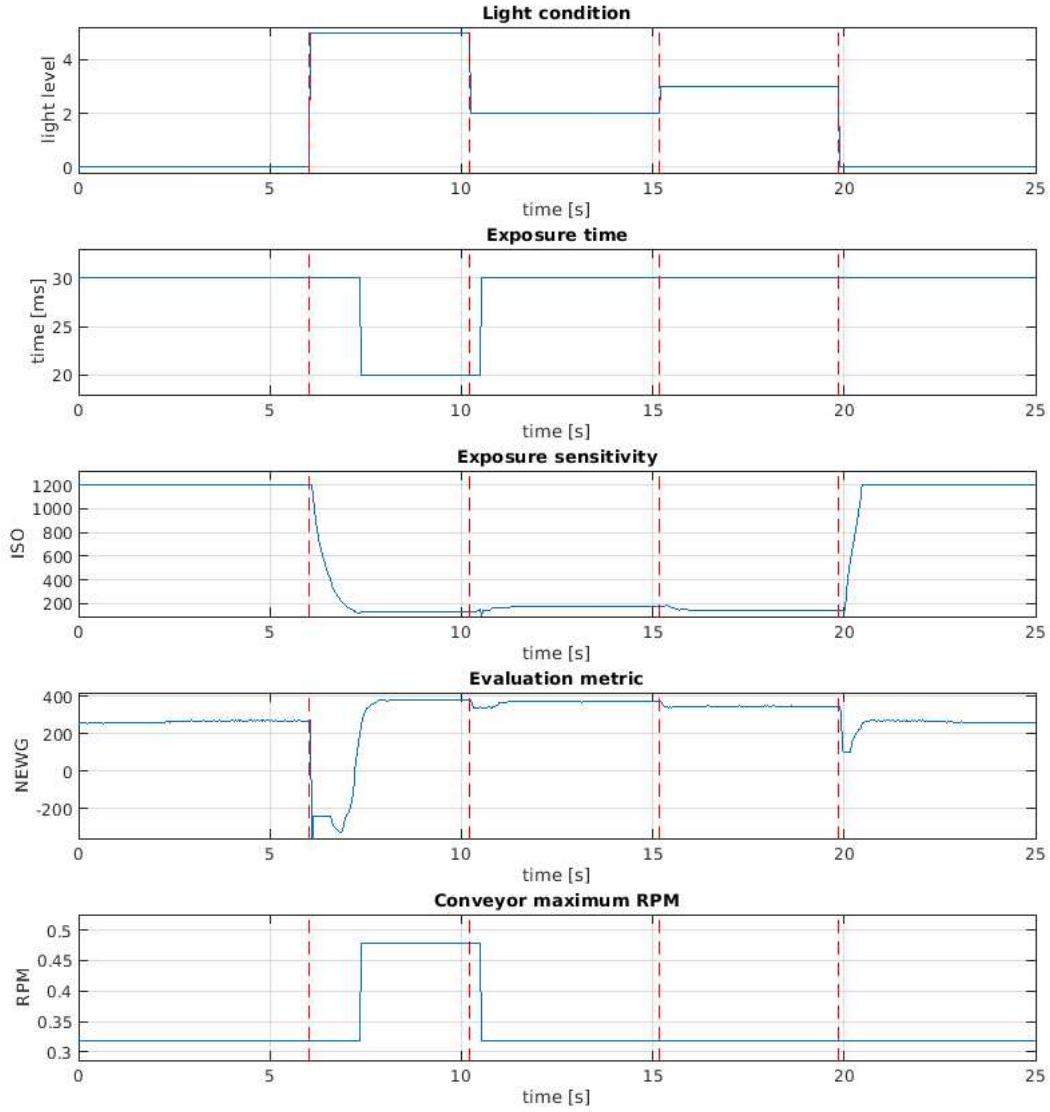


Figure 26: Performance of Luxonis auto-exposure control and related calculations of NEWG and RPM of the conveyor motor

The segmented red lines in the graphs in Figure 26 correspond to the moments when there is a change in brightness in the environment. This allows us to analyze the response time of the algorithm which, as mentioned above, prefers to prioritize the variation of sensitivity and maximize the exposure time. This behavior is visi-

ble in the ”Exposure sensitivity” and ”Exposure time” graphs in Figure 26. At the moment when there is a change in brightness, the first parameter that is modified is the ISO with a response time that varies between 100 and 120 ms, corresponding to 3/4 frames. This results in improved performance, from this point of view, by more than 100%. The variation of the exposure time occurs more slowly, but more abruptly, varying only when the ISO value has reached a lower limit. As soon as the conditions allow it, such as at the level two of illumination, the exposure time tends to remain as high as possible, modifying the ISO if necessary.
The trend of the *NEWG* evaluation parameter of the frames behaves in a way that corresponds to the behavior of the controller. In particular, it can be seen in Figure 26 in the ”Evaluation metric” graph that there is a constant behavior when the controller has reached optimal values. On the contrary, during moments of brightness variation, during the optimization process, there are oscillations that manifest as undershoots proportional to the variation in brightness in the environment, given by the contribution of the entropy calculation which suppresses and exceeds the contribution of the gradient calculation.

7.3.1 Conveyor belt speed calculation

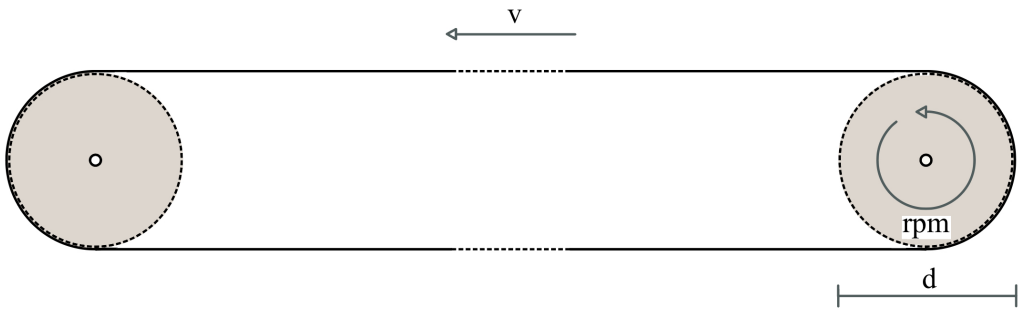


Figure 27: Illustrative diagram of the conveyor belt, d is the diameter of the driving rollers, rpm are the numbers of revolutions per minute done by the driver and v is the resulting belt speed

In Figure 27, an illustrative diagram is presented to understand the parameters to consider when calculating the conveyor belt’s speed. It is assumed that the motor’s revolutions per minute (rpm) are calculated at the drive roller, thereby bypassing

any potential gear reducers. The C++ code is developed to input as model parameters the maximum error associated with motion blur, denoted as e, the diameter of the rollers d, and optionally the maximum and minimum rpm values. This approach ensures that values incompatible with the used conveyor belt are not transmitted. The maximum speed v to achieve an error e less than the set threshold is calculated using the formula 17, and the corresponding rpm is determined using the formula 18. The variable *exp_time* represents the exposure time of the frame that has been captured. Specifically, by defining an object of the class *depthai.ImgFrame*, it is possible to acquire frames and their corresponding data from the *ColorCamera* node via the *xlinkout* messages, as explained in Chapter 6.1.

$$\mathrm{v} = \frac{\mathrm{e}}{\mathrm{exp\_time}} \quad \text{(m/s)} \tag{17}$$

$$\mathrm{rpm} = \frac{(\mathrm{v} \times 60)}{\pi \times \mathrm{d}} \quad \text{(rpm)} \tag{18}$$

Based on the specific application where the system is used, it is possible to calculate the parameters that best fit the type of control required. In the case study, it was not possible to implement automatic control of the conveyor belt's speed, as it is manually controlled with a potentiometer that acts on the inverter controlling the motor. Therefore, the maximum and minimum speeds were set, and the optimal speed was transmitted as a percentage value.

Due to the fact that the maximum and minimum speed values of the conveyor belt are not reported in the BMTec documents, it was necessary to manually calculate the speed of the belt using the cameras that are the subject of this study in order to obtain consistent data and a possible percentage value of the speed. Specifically, a reference was set on the moving part of the belt and two fixed references were set in the system frame at a distance of $41\ cm$. The machine was started and set first to the minimum speed and then to the maximum. With simple calculations, the minimum and maximum speed were calculated considering the time taken to move from the first to the second reference. The values obtained are approximately $0.03\ m/s$ and $1.8\ m/s$, as already mentioned in the introduction.

In Figure 26, in the graph named "conveyor maximum RPM", it is possible to see the trend of the expected rpm on the conveyor belt, considering an error $e = 0.1$ mm and a roller diameter of 0.08 m.

7.3.2 Determining the e error parameter in the context of motion blur

In the context of motion blur reduction, the selection of the error parameter e is critical as it represents the maximum permissible error that can be set to limit motion blur. This choice has a direct consequence on the calculation of the conveyor belt's speed as explained in the previous chapter, necessitating a careful selection based on the specific application. For instance, in applications involving the calculation of density, defined as the number of objects on the conveyor belt, a larger error e can be tolerated. This is because the presence of motion blur does not significantly affect the final result, allowing for a higher value of e which can enhance system performance. Conversely, when precise measurements of objects' dimensions on the belt are required, such as in quality control scenarios, the value of e must be chosen in accordance with the tolerances specified by ISO standards.

ISO 4759, particularly, outlines a set of tolerances for fasteners, which can be analogous to setting tolerances for motion blur in imaging systems. According to ISO 4759, tolerances are categorized into product grades A, B, and C, with grade A being the most precise and grade C the least. These tolerances are crucial for ensuring that components fit together correctly without excessive looseness or tightness, which in the context of motion blur, translates to ensuring that the image quality is maintained within acceptable limits. The standard was last reviewed and confirmed in 2022, indicating its relevance and applicability to current technologies.

To better understand the problem of motion blur in the context of study, the system was tested using washers. Specifically, the OAK-D lite camera was fixed on a structure set up above the conveyor belt, and the system was started with washers placed on the plane. Mainly two tests were carried out, the first with the maximum speed of the belt and the second by manually setting a speed low enough not to cause motion blur. Due to the manual control of the system, it was not possible to precisely set the belt speed based on the error e, but it was possible to analyze the

consequences of incorrect system calibration. In fact, as can be seen in Figure 28, setting a speed too high and/or an exposure time too long results in an effect like the one in the Figure on the left where the washers are blurred and inaccurate. On the contrary, with correct calibration, the outcome is as depicted in the Figure on the right. In this instance, the washers are notably precise and sharp, thereby enabling for example an accurate calculation of the dimensions.

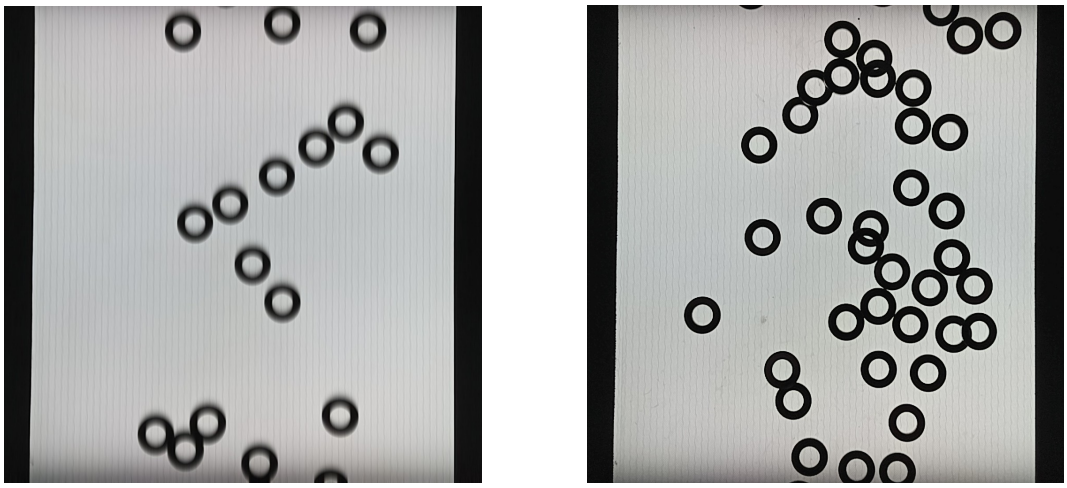


Figure 28: Effect of motion blur in the conveyor belt system

Therefore when applying these principles to the selection of e, one must consider the application's tolerance for error against the desired precision of the imaging results. For quality control purposes where dimensions are critical, a lower e corresponding to a higher product grade, such as A, would be necessary to meet the stringent ISO tolerances. This ensures that the dimensions of objects captured on the conveyor belt are within the acceptable limits set by the ISO 4759 standard, which is essential for maintaining quality assurance and meeting regulatory compliance. Thus, the selection of e is not merely a technical decision but also a strategic one that aligns with industry standards and quality benchmarks.

7.3.3 Implementation with ROS environment

The integration of ROS, or Robot Operating System, into the C++ codebase enhances the communication capabilities with other nodes within a robotic system. ROS is a middleware that provides a structured communications layer above the

host operating systems, offering a set of software frameworks for robot software development. It includes tools and libraries that aid in creating complex and robust robot behavior across various robotic platforms [28].

In the case study, ROS Noetic was chosen in conjunction with Ubuntu 20.04, reflecting the version used in University laboratories. The use of ROS brings several benefits, such as modularity, allowing for software to be written in nodes that can be reused and adapted for different applications. Its flexibility supports a wide range of hardware and software, making it adaptable to diverse robotic platforms. The open-source nature of ROS fosters a large community that contributes to its development and provides extensive support. Additionally, ROS comes equipped with numerous tools and libraries that facilitate the development, testing, and deployment of robot software.

The C++ code's ROS integration allowed for the setting of the conveyor belt's speeds, the exposure time and ISO. In particular the camera is set as a node that has three topics, through which the messages are sent.

To test locally the code, a ROS network was initialize using *roscore* in the terminal, then in another page *rosnode list* was used to check the correct set up of the node and *rostopic list* to check the topics. Then using *rostopic echo* followed by the topic was checked the correct sending of the messages.

Although the case study did not implement automatic control of the conveyor belt's speed due to the manual control, the inclusion of ROS opens up possibilities for future automation and data exchange with other systems, enhancing functionality and adaptability.

8 Future prospects

This chapter presents potential projects, highlighting their advantages and features. The aim is to inspire further research and innovation in these areas.

8.1 ROS and Gazebo for conveyor belt control and simulation

One potential implementation involves leveraging simulation software such as RViz and Gazebo to simulate the behavior of the conveyor belt. These simulation algorithms have already been tested and proven feasible using ROS, thus demonstrating their viability. One key aspect of this possible implementation is to integrate cameras into the simulation system, this allows the evaluation and optimization of positioning along the production line to streamline the process. Furthermore, within the same simulation environment, other devices like robotic manipulators or packaging and distribution systems can be incorporated, simulating the entire production process. Moreover, it may be possible to simulate the feasibility of implemented control algorithms by simulating objects on the conveyor belt and analyzing the performance of the algorithms and their repercussions on the overall system.
Thanks to the flexibility of ROS, real physical systems can be integrated simultaneously with simulated ones by associating the node of such a device not only with simulation but also with the chosen algorithm and hardware for the application.
In conclusion, this type of implementation, albeit complex, allows for cost reduction during the design phase by having a fully simulated system communicating within a ROS framework, where each component can be managed as an individual node, leading to significant cost savings and enhanced problem-solving capabilities.

8.2 Integration of a centralized exposure control system

The implementation of a centralized system for camera control and potential post-processing algorithms, integrating a series of cameras into a ROS network, could be possible. This is due to the previously explained algorithms that allow simultaneous control of various cameras, creating and initializing different objects for each camera on which various optimization operations are then implemented. This

allows comprehensive control of the various integrated systems, reducing costs and hardware implementation. However, this system could have gaps that stem from problems already encountered in this thesis. In fact, everything should be tested to verify that the chosen cameras are fast enough in both the execution of commands and the transmission of information in order to reduce possible delays. Furthermore, the computer managing everything must have sufficient computing power to handle the workload. Once these issues are overcome, it could be possible to have a single control system that exchanges information between the various devices integrated into the system through a ROS network.

In addition to the advantages already mentioned, it could be possible to make the various production lines collaborate, optimizing the workload based on the information acquired from the cameras. The possible disadvantages, on the other hand, could be related to the maintenance and failures of such a system as they would involve the interruption of the entire production chain since the control is centralized for several production lines.

The potential implementation just described could bring advantages in an industrial context, but it is crucial to carefully evaluate the needs and potential issues that may arise.

9 Conclusions

In conclusion, this thesis has offered an in-depth analysis of the multifaceted aspects related to the incorporation of cameras into industrial systems. The algorithms studied have shown a wide range of potential applications, and by addressing latency issues in exposure control, they can achieve superior performance compared to the original control algorithm.
In particular, among the two implemented algorithms, the one developed in the paper "Proactive camera attribute control using Bayesian optimization for illumination-Resilient visual navigation" [8] demonstrates better performance compared to the algorithm proposed in the paper "Active exposure control for robust visual odometry in HDR environments" [30]. However, it encounters problems under low-light conditions as described in Chapter 7.1.2. Despite this, the complexity of the first algorithm is higher, and therefore its integration might be inefficient in systems with low computational power. On the contrary, the algorithm proposed in the paper "Camera Attributes Control for Visual Odometry With Motion Blur Awareness" [7] combines the advantages of the two systems by proposing a less computationally complex algorithm that integrates the key features of the first algorithm. Unfortunately, it was not possible to implement and test its actual performances.

The choice of cameras is crucial for the desired application. Unfortunately, with the current software releases available for the Luxonis cameras, it was not possible to implement the algorithms as desired. Moreover, despite the various challenges described throughout the thesis, which prevented the physical implementation of the system, the control algorithm still holds significant potential for integration.
This research highlights the importance of optimizing camera integration within industrial environments, paving the way for improved efficiency and effectiveness in industrial processes. Even though the full potential of the system could not be realized due to the limitations of the current software releases, the theoretical groundwork laid in this thesis provides a solid foundation for future advancements in this field. The potential for the integration of the control algorithm into industrial systems remains promising, despite the current challenges.

References

[1] Ahmed Nabil Belbachir. *Smart cameras*. Springer, 2010. ISBN: 978-1-4419-0952-7. DOI: 10.1007/978-1-4419-0953-4.

[2] Paul Bergmann, Rui Wang, and Daniel Cremers. "Online Photometric Calibration of Auto Exposure Video for Realtime Visual Odometry and SLAM". In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 627–634. DOI: 10.1109/LRA.2017.2777002.

[3] BMTec. *BMTec homepage*. Accessed: March 2024.

[4] Paul E. Debevec and Jitendra Malik. "Recovering High Dynamic Range Radiance Maps from Photographs". In: *SIGGRAPH 97* (1997).

[5] Lorenzo Finesso. *Lezioni di probabilità seconda edizione*. Edizione libreria progetto Padova, 2018. ISBN: 978-88-96477-91-5.

[6] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing fourth edition*. Pearson Education Limited, 2018. ISBN: 978-1-292-22304-9.

[7] Bin Han et al. "Camera Attributes Control for Visual Odometry With Motion Blur Awareness". In: *IEEE/ASME Transactions on mechatronics, VOL. 28, NO. 4*. IEEE. 2023, pp. 2225–2235.

[8] J. Kim, Y. Cho, and A. Kim. "Proactive Camera Attribute Control Using Bayesian Optimization for Illumination-Resilient Visual Navigation". In: *IEEE Transactions on Robotics* (2020).

[9] Luxonis. *API documentation*. Accessed: January 2024.

[10] Luxonis. *C++ API reference*. Accessed: January 2024.

[11] Luxonis. *Color camera node*. Accessed: January 2024.

[12] Luxonis. *Depth accuracy*. Accessed: November 2023.

[13] Luxonis. *DepthAI Hardware Documentation*. Accessed: November 2023.

[14] Luxonis. *Device comparison*. Accessed: November 2023.

[15] Luxonis. *Device queue event*. Accessed: January 2024.

[16] Luxonis. *home page*. Accessed: November 2023.

[17] Luxonis. *OAK CM models*. Accessed: November 2023.

[18] Luxonis. *OAK Device Features Comparison*. Accessed: November 2023.

[19] Luxonis. *OAK Series 2*. Accessed: November 2023.

[20] Luxonis. *OAK-D Baseboard Specifications*. Accessed: November 2023.

[21] Luxonis. *OAK-D Pro Advanced Features*. Accessed: November 2023.

[22] Luxonis. *OAK-D S2 Series 2 Version*. Accessed: November 2023.

[23] Luxonis. *OAK-D-Lite Specifications*. Accessed: November 2023.

[24] Luxonis. *Robotics Vision Core 2 (RVC2)*. Accessed: November 2023.

[25] Luxonis. *Standalone mode*. Accessed: November 2023.

[26] Luxonis. *xlinkIn node*. Accessed: January 2024.

[27] Luxonis. *xlinkOut node*. Accessed: January 2024.

[28] ROS. *ROS noetic homepage*. Accessed: February 2024.

[29] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. DOI: 10.1017/CBO9781107298019.

[30] Zichao Zhang, Christian Forster, and Davide Scaramuzza. "Active exposure control for robust visual odometry in HDR environments". In: *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2017, pp. 3894–3901.